Kofax RPA

User's Guide

Version: 11.1.0

Date: 2020-09-16

Table of Contents

Preface

This guide is a PDF version of *Help for Kofax RPA*.

Related Documentation

The documentation set for Kofax RPA is available here:[1]

https://docshield.kofax.com/Portal/Products/RPA/11.1.0_vwsnqu4c9o/RPA.htm

In addition to this guide, the documentation set includes the following items:

Kofax RPA Release Notes

Contains late-breaking details and other information that is not available in your other Kofax RPA documentation.

Kofax RPA Technical Specifications

Contains information on supported operating systems and other system requirements.

Kofax RPA Installation Guide

Contains instructions on installing Kofax RPA and its components in a development environment.

Kofax RPA Upgrade Guide

Contains instructions on upgrading Kofax RPA and its components to a newer version.

Kofax RPA Administrator's Guide

Describes administrative and management tasks in Kofax RPA.

Kofax RPA Best Practices Guide for Robot Lifecycle Management

Offers recommended methods and techniques to help you optimize performance and ensure success while using Robot Lifecycle Management in your Kofax RPA environment.

Kofax RPA Getting Started with Desktop Automation Guide

Provides a tutorial that walks you through the process of using Kofax RPA Desktop Automation to build a robot.

[1] You must be connected to the Internet to access the full documentation set online. For access without an Internet connection, see the *Installation Guide*.

Kofax RPA Getting Started with Document Transformation Guide

Provides a tutorial that explains how to use Document Transformation functionality in a Kofax RPA environment, including OCR, extraction, field formatting, and validation.

Kofax RPA Desktop Automation Service Configuration Guide

Describes how to configure the Desktop Automation Service required to use Desktop Automation on a remote computer.

Kofax RPA Developer's Guide

Contains information on the API that is used to execute robots on RoboServer.

Kofax RPA Integration API documentation

Contains information about the Kofax RPA Java API and the Kofax RPA .NET API, which provide programmatic access to the Kofax RPA product. The Java API documentation is available from both the online and offline Kofax RPA documentation, while the .NET API documentation is available only offline.

> **Note** The Kofax RPA APIs include extensive references to RoboSuite, the original product name. The RoboSuite name is preserved in the APIs to ensure backward compatibility. In the context of the API documentation, the term RoboSuite has the same meaning as Kofax RPA.

Training

Kofax offers both classroom and computer-based training to help you make the most of your Kofax RPA solution. Visit the Kofax Education Portal at https://learn.kofax.com/ for details about the available training options and schedules.

Also, you can visit the Kofax Intelligent Automation SmartHub at https://smarthub.kofax.com/ to explore additional solutions, robots, connectors, and more.

Getting help with Kofax products

The Kofax Knowledge Base repository contains articles that are updated on a regular basis to keep you informed about Kofax products. We encourage you to use the Knowledge Base to obtain answers to your product questions.

To access the Kofax Knowledge Base, go to the Kofax website and select **Support** on the home page.

> **Note** The Kofax Knowledge Base is optimized for use with Google Chrome, Mozilla Firefox or Microsoft Edge.

The Kofax Knowledge Base provides:

- Powerful search capabilities to help you quickly locate the information you need.

  Type your search terms or phrase into the **Search** box, and then click the search icon.

- Product information, configuration details and documentation, including release news.

  Scroll through the Kofax Knowledge Base home page to locate a product family. Then click a product family name to view a list of related articles. Please note that some product families require a valid Kofax Portal login to view related articles.
- Access to the Kofax Customer Portal (for eligible customers).

  Click the **Customer Support** link at the top of the page, and then click **Log in to the Customer Portal**.
- Access to the Kofax Partner Portal (for eligible partners).

  Click the **Partner Support** link at the top of the page, and then click **Log in to the Partner Portal**.
- Access to Kofax support commitments, lifecycle policies, electronic fulfillment details, and self-service tools.

  Scroll to the **General Support** section, click **Support Details**, and then select the appropriate tab.

Chapter 1

Introduction

Kofax RPA is a platform for application integration and Robotic Process Automation (RPA). It can integrate applications that were not built to be connected and automate processes across such heterogeneous systems; cloud/SaaS applications with premise systems, legacy systems with modern web applications, back office systems with partner websites.

With our visual editor Design Studio, you can click through the applications and data sources you want to integrate and create an automated workflow using standard steps.

In Kofax RPA, these workflows are known as robots. As you build a robot, you are free to navigate through the applications as you integrate them. You can login to applications, extract data parts of a page, enter data into forms or search boxes, make menu selections, and scroll through multiple pages. Your robot can also access databases, files, APIs, web services, and other robots, exporting data from one application and loading it into another; recognize data using an OCR engine in the Document Transformation Service, transforming data as necessary along the way.

The Document Transformation functionality in a Kofax RPA environment covers functionality such as OCR, extraction, field formatting, and validation.

Desktop Automation in Kofax RPA helps you automate Windows and Java applications on your network computers. Desktop Automation replaces manual processes by controlling an application on a desktop or a terminal. See Desktop Automation for details.

Once built, robots are uploaded to a repository in the Management Console. From here, they can be scheduled for batch-execution on the RoboServer or executed on-demand via Java and C# APIs, or tailored REST services. The REST services are instantaneously available once the robots are added to the repository or exposed as special-purpose end-user web applications called Kofax RPA Kapplets.

The Management Console is also responsible for load balancing, failover, monitoring of RoboServer health and management of user roles and permissions.

Naming Policy

Kofax RPA imposes the following naming policy, which applies to project names, schedule names, folder names (paths), and folder items including robots, types, snippets, and resources uploaded through the API.

- Illegal system characters are allowed, but they generate a warning message.
- Empty names are not allowed, and they generate an error message.
- Folder names must be unique.
- Names that exceed 243 characters are not allowed, and they generate an error message.

- Names with special HTML formatting are allowed, but they generate a warning. The HTML formatting is removed.
- Do not use system reserved words, because you cannot save a file with a system reserved name to a disk. For system reserved words check your operating system documentation.
- Name `local` is a reserved name and cannot be used in Desktop Automation.
- Do not use the following reserved keywords in type and variable names to avoid database and other errors. Check your database documentation for a full list of reserved keywords.
  - integer
  - int
  - bigint
  - varchar
  - float
  - null

The use of period "." in file names

You cannot use or create files and folders that start with period "." in Kofax RPA including robots, snippets, types, texts, database (under cluster), and database mapping files.

Name Conversion Algorithm

Typed name	Result
'<t> aaa </tag>' **Note** ' ' denotes the start and end of the string.	'aaa' **Note** The HTML tags <t> and </tag> are removed. The space before and after the actual folder/project/ schedule name is also removed.
'< t> hello '	'< t> hello' **Note** The tag < t> is kept because it is not a valid HTML tag. The space after the actual name is removed.
'com1 /*** /&#38; &#64;'	'com1/***/& @' **Note** Because "com1" is a reserved word for some operating systems, it generates a warning; however, it is a valid name for a folder, project, or schedule. An asterisk "*" is an illegal character. Kofax RPA accepts it as a valid name but generates a warning. Note that entering a name with reserved words and illegal characters can result in errors when other users download the project to local computers where the operating system does not support the names as valid file names. HTML entity numbers such as &#38; are converted to actual characters.

Chapter 2

Tutorials

The topics in this section contain links to video tutorials that help you perform different tasks in Kofax RPA.

Note You need Internet connection to view the video tutorials.

Beginner Tutorials

This section contains several tutorials that provide an overview of Kofax RPA as well as guide you through your first project in this product. Make sure to install and set up Kofax RPA correctly before proceeding with these tutorials. Click the links to videos to play them.

Introduction

Introduction Tutorial Video.

Robot Beginner's Tutorial

Robot Beginner's Tutorial Video.

Kapplet Beginner's Tutorial

Kapplet Beginner's Tutorial Video.

Type Beginner's Tutorial

Type Beginner's Tutorial Video.

Chapter 3

Design Studio

Design Studio is the application for creating robots and types. In Design Studio, you can also debug your robots and create database tables for types that need to be stored in databases.

Design Studio, an integrated development environment (IDE) for robot development, is all you need to design robots and types.

Robots are programmed in an easy-to-understand visual programming language with its own syntax (structure) and semantics (meaning). To support you in the construction of robots, Design Studio provides powerful programming features, including interactive visual programming, full debugging capabilities, an overview of the program state, and easy access to context-sensitive online help.

Design Studio also lets you create the types that are used by robot variables for data extraction and input. With Design Studio Type Editor, you can design types that are modeled after real-world data. In the most common case, a type is designed to hold the data that a robot extracts from a data source.

Before You Read On

Before you proceed, we recommend that you read Introduction to Design Studio, which will introduce you to Design Studio and the context it is used in.

Introduction to Design Studio

Design Studio is a programming environment for creating robots and designing types. Robots are created using a special-purpose programming language with its own syntax and semantics. Like other programming environments, Design Studio uses several concepts that you, as a robot designer, must understand to fully comprehend the workings of Design Studio. The purpose of the introduction is to define the most important concepts and we recommend that you refer back to this section whenever necessary. The Design Studio concepts become clearer as you explore Design Studio and start creating robots.

Robots

The most important concept in Design Studio is a robot. A robot is a program designed to accomplish some task involving a data source, usually a web site or other robot, but it could also be an Excel document or a database. Typically, one robot is written per task per data source. For example, you would create one robot for extracting news from https://cnn.com, another robot for extracting news from https://yahoo.com and yet another robot for extracting product information from an online product catalog.

Basically, a robot can be programmed to do (automatically) everything you can do in a browser, and to extract data from a database or an Excel document to combine with data stored in a database or file.

In Kofax RPA, we distinguish between two types of robots: Web Automation robots (usually referred to as just "robots") and Desktop Automation robots. For more information, see the next section.

Tip Visit the Kofax Intelligent Automation SmartHub at https://smarthub.kofax.com/ to explore additional solutions, robots, connectors, and more.

Web Automation & Desktop Automation

Web Automation Robot

Kofax RPA robots mimic human behavior; they do what you do when you are looking for content on the Internet using a browser. You start by searching for the content. Once found, you read and process it. Similarly, most robots can be divided into two parts: a navigation part and an extraction part.

Navigation is concerned with "getting to where the content is." Navigation mainly includes loading pages and submitting forms. When navigating in Design Studio, you typically use the Click action to navigate through and among web pages.

Extraction is concerned with "getting the right content." Extraction mainly includes selecting, copying, and normalizing content from a web page that you navigate to. When extracting in Design Studio, you typically use the Test Tag action to skip uninteresting ("noisy") content, the Extract action to copy content into variables, and the data converters for normalizing the content so that it gets the format you want, such as the right date and number format. Once extracted, you output the value with the Store in Database or Return Value action.

A typical robot starts with one or more steps, each containing a Load Page or Click action to navigate to the interesting content on a web site. It proceeds with one or more steps, each containing an Extract action, and ends with a step storing or returning the extracted value.

Note that in many robots, the navigation and extraction parts overlap because the content to extract is located on several pages. Again, this is similar to looking for content yourself; often, you have to visit several pages to get the content you want.

Most robots include other actions than the ones mentioned above, such as a For Each Tag action for loading several similar looking pages or extracting values from several similar looking table rows. Because robots have different tasks, they have different needs. For this reason, we have included a considerable number of step actions and data converters in Design Studio. Start by familiarizing yourself with the basic and most commonly used step actions and data converters, and then begin to explore. Experience shows that you can create most robots using only a handful of step actions and data converters. So, find your own favorite step actions and data converters and stick to them until you feel a need to explore others. See Web Automation for details.

Desktop Automation Robot

Kofax RPA was originally designed for accessing HTML at a time when HTML pages were mostly static. In those cases, the state of the application (web page) can be tracked internally in the robot. By contrast, the Desktop Automation functionality is designed to automate remote applications where the state resides in the application. In this case, the state is external to the robot.

In Kofax RPA, you can create Desktop Automation robots that can automate work processes involving Windows and Java applications on your networked computers. The workflow of a Desktop Automation robot is a sequence of steps that are executed one after the other. The steps model how a user would interact with the application that is being automated. See Desktop Automation and Introduction to Desktop Automation for details.

Document Transformation

The Document Transformation feature helps you extract and use information from images and text documents in a Desktop Automation robot. The Document Transformation action processes your graphical or PDF documents using a selected project. A project is a module that processes and transforms your documents using OCR and other specified operations. See Document Transformation for details.

Robot Execution Mode

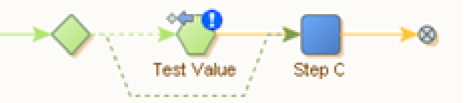
Kofax RPA Design Studio supports two design-time robot execution modes: Minimal Execution (Direct) and Smart Re-execution (Full). This topic provides details about the two modes.

When creating a new Web Automation robot, you can select execution mode in the new robot wizard. Use the Design Mode tab of the robot configuration to view or change the execution mode. Note that the choice of robot execution mode only impacts the execution in Design Mode, and not in Debug Mode or at runtime in RoboServer.

Smart Re-Execution (Full)

In Smart Re-execution mode, the way that the robot is executed in Design mode is similar to the way it is executed at runtime or in Debug mode. This is the default mode for new Web Automation robots.

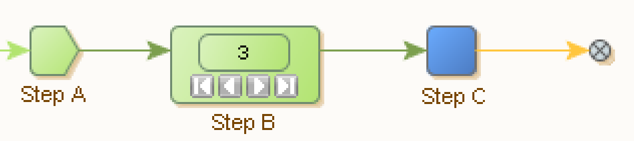
As an example, when you click step C in the following robot, it automatically executes through the bottom branch of the Try construct when the test fails in the Test Value step:



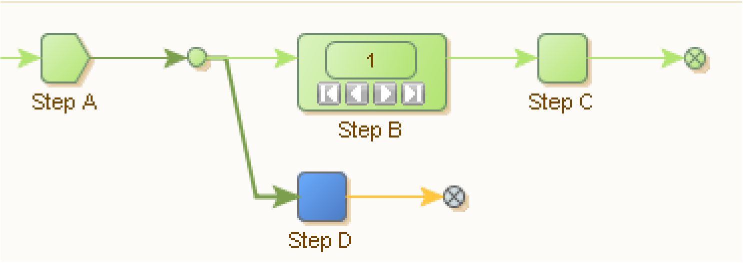
The blue exclamation mark icon on the Test Value step indicates that the error handling to Try Next Alternative was triggered.

With loop steps, all iterations up to and including the selected iteration are executed when clicking a step inside the loop.

In the following example, clicking step C causes execution of three iterations of the loop.



Further, if clicking a step in a branch below a loop, all iterations of the loop are executed. As an example, see the following robot.



Clicking step D causes execution of step A and repeated execution of steps B and C (as many times as there are iterations in loop B) before finally stopping at step D.

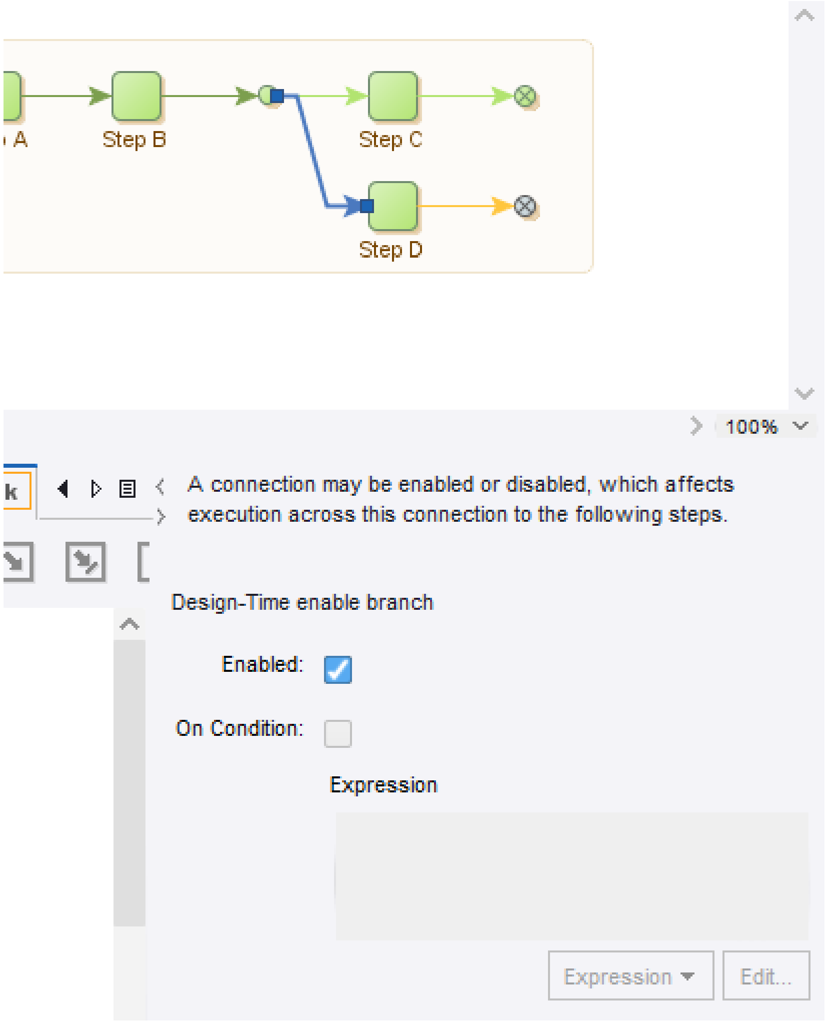
The Smart Re-execution mode is particularly useful when working with global variables, and when you have subsequent steps in the robot that depend on accurate variables. This mode may be useful for building a payload for a web service (REST or SOAP) call, or constructing an Excel document. The XML or Excel document that is being populated resides in a global variable, while its content is added during the execution of a loop. In a branch below the loop that populates the document, the robot takes the entire document and posts it to a web service or similar. In this case, the Smart Re-execution mode makes it easier to build the robot, as it ensures that the document is populated when testing the web service call in Design Mode.

In Smart Re-execution, the interaction with the external world in the form of websites, databases, or web services is cached. Caching avoids re-execution of steps unless the prerequisite for storing the execution result has changed (such as a variable that determines which URL to load). Smart Re-execution has a higher memory footprint than the Minimal Execution mode.

The Smart Re-execution mode is the only mode that supports Desktop Automation workflow. The initial execution of the Desktop Automation workflow caches the state of the returned variables, and the cached variables are not updated when the Desktop Automation workflow is updated. Changing the Desktop Automation workflow does not refresh the cached variable state. Re-execute the entire robot to update the variable values returned from the Desktop Automation workflow.

We do not recommend Smart Re-execution mode for large robots with significant interaction with the external world, or for long-running robots. The execution time as well as memory usage are too high in these cases.

To cut down on the execution time while designing the robot, you can right-click a branch and disable it in Design Studio. A similar setting can be applied in Debug Mode. Additionally, you can disable the branch based on a specified condition in select iterations.



A connection may be enabled or disabled, which affects execution across this connection to the following steps.

Design-Time enable branch

Enabled: ☑

On Condition: □

Expression

Connection configuration

The Design Mode tab of the robot configuration also features an option "Avoid External Re-execution". When checked, it is ensured that steps are never re-executed, even when the cached result of the previous execution cannot be used. In this case, you can still edit the robot, but without a current input state to work on. Use this option only to meet requirements for interaction with the external world to avoid re-execution (for example, if re-execution would result in incorrect or duplicate data in a partner's system).

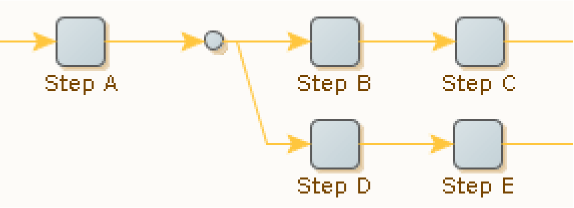
Important Some step actions are not available in the Smart Re-execution mode. For a list of unavailable steps, see the "Execution Mode" section in the Kofax RPA Limitations topic.

Minimal Execution (Direct)

The Minimal Execution (Direct) mode is the traditional Design Studio execution mode. All robots written in versions prior to 9.5 will use this execution mode.

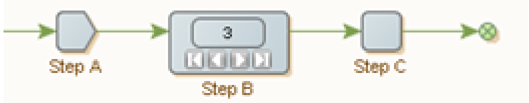
When you click a step in Minimal Execution mode in the robot graph, Design Studio takes the shortest direct path to that step, skipping any previous branches and iterations that are not on the direct path.

Consider the example below:



During runtime execution, the robot would normally execute steps A, B, C and D before reaching step E. But in Design Mode, clicking step E result only in the execution of steps A and D.

Similarly, if the step resides inside a loop, only the selected iteration is executed.

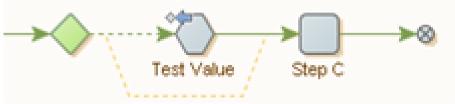


As the iteration counter is set to 3, clicking step C cause only step A, B and C to be executed once, where step B selects the third iteration.

The Minimal Execution mode is optimized towards executing as few steps as possible. This mode is useful when you have large robots and steps that take considerable time to execute, such as steps that interact with complex websites. Generally, we recommend Minimal Execution for most data collection use cases and for robots that perform significant interaction with external sites.

The drawback of Minimal Execution mode is that it requires user assistance to select the path to a given step whenever it cannot execute directly to the step using the default path; for example, try steps in a path may prevent a robot from following the topmost branch.

See the following example.



When clicking step C, Minimal Execution mode is not able to proceed if the test fails in the Test Value step. In this case, the user must explicitly click the bottom branch of the try step first, to guide the execution path towards step C.

The Robot State

When a robot is executed, it works on a robot state, which consists mainly of four elements:

- Windows
- Variables
- Cookies
- Authentications

The Windows element corresponds to the currently open windows, each containing a page. This page could be an HTML page, a spreadsheet, an XML page, and so on. The page has a given Page Type depending on what type of page is loaded into the window and the look of the Page View and the steps you can insert in your robot depend on this type. At least one window is always open, and one window is marked as the current window. The variables element contains the current values of the variables. The cookies and authentications elements are the HTTP cookies and authentications, respectively, received during communication with a web server.

Steps

A robot is made up of steps, which are building blocks in a robot program.

There are five types of steps:

- Action
- Try
- Snippet
- Group
- End

A step works on a robot state and processes it according to the configuration of the step. A step has an input robot state and generates an output robot state. The only exception is the End step. End steps mark the end of a branch in a robot, but not the end of a robot. For example, the robot does not necessarily stop execution after an end step. End steps are the only steps in a robot that do not have outgoing connections.

Steps may have properties, such as a step name, a list of tag finders, a step action and error handling. While Action steps have all these properties, other types of steps only have some.

The step name provides a symbolic name for the step, such as "Extract Headline" and "Load Search Page." In the preceding robot, the step name is "MyStep".

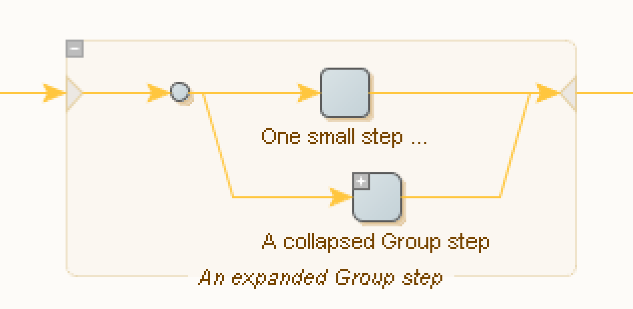
The finders find the elements (HTML/XML tags or Excel cells) in the page that the step action should work on. Some step actions require a single element, whereas others can handle more than one element. Some step actions accept no elements at all.

In a Web Automation robot, there are two kinds of finders: Tag Finders that find tags in HTML or XML pages and Range Finders that find cells in Excel pages.

The step action is the action that the step performs. The action is the "heart and brain" of the step, and it is the selection of the right step action that is the challenge of robot writing. For example, an Extract action can extract the text from a tag in an HTML page and store it in a variable. A Click action can load the URL residing in an <a>-tag and replace the page of the current window in the robot state with the newly loaded HTML page. An action usually changes the robot state. For example, the Extract action changes the variables, and the Click action may change the pages/windows, the cookies and the authentications.

A step can be executed. A step that is executed accepts a robot state as input and, by applying the finders and step action in turn, produces an output robot state. The output robot state is then passed to the following step and becomes its input robot state. Some step actions are "termed loop" actions and steps having such actions are called "loop steps." A loop step may generate zero or more output robot states, and cause the following steps to be executed once for each of them.

You can group steps together in expandable Group Steps. The figure below shows an example of an expanded Group step of a Web Automation robot with a collapsed Group step inside it.



A step is valid if it is properly configured so that execution can be attempted. For example, if a step has no action, it is invalid as execution cannot be attempted.

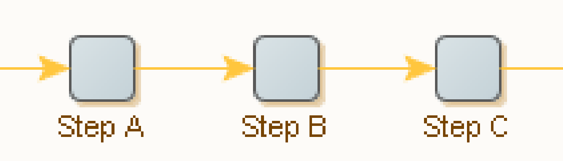
A step definition also specifies error handling.

Connections and Execution Flow

Use connections to determine the execution flow between steps.

Note Examples in this topic are based on the Minimal Execution (Direct) design-time execution mode.

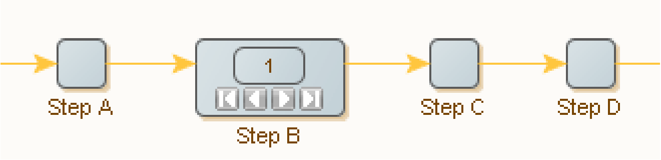
Consider the following simple robot:



This robot consists of three steps: Step A, Step B, and Step C. Assuming that no errors occur, and that each step generates exactly one output robot state, the robot is executed as follows: An initial robot state is generated and used as input to Step A (being the first step). Step A produces an output robot state. This output robot state is the input robot state of Step B. Similarly, Step B produces a robot state, which is the input robot state of Step C. Once Step C has executed and produced an output robot state, execution completes. In short, the execution of steps is described as follows: "A, B, C."

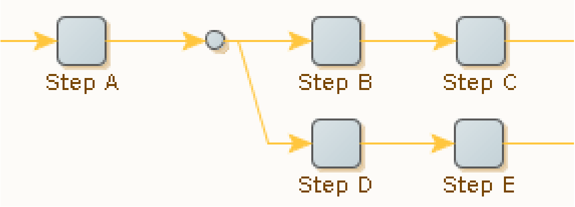
Sometimes, a step generates no output robot state when executed. This happens when an error or a test step causes execution to continue somewhere else in the robot (see Conditions and Error Handling).

Steps containing a loop action may process the input state several times, each time outputting a distinct robot state. Consider the following robot where step B contains a loop action:



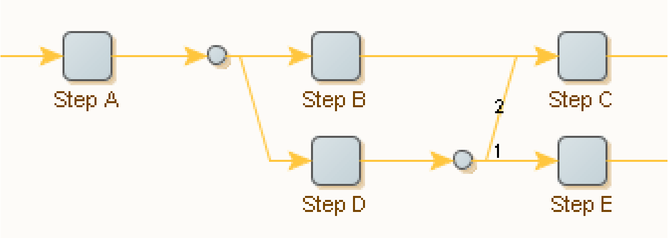
Assuming that there are no errors or test steps, that step B outputs three robot states, and that all other steps output exactly one robot state, the steps are executed in the following order: "A, B[1], C, D, B[2], C, D, B[3], C, D", where B[N] refers to the N th iteration of the loop action contained in step B. Note that the output robot states by step B are different robot states: each iteration will output a new robot state. Hence, step C will receive a new input robot state each time it is executed.

A step can connect to more than one step. This is called "branching". Consider the following robot:



In this robot, step A is followed by a branch point, where the connection splits out in two branches. One branch consists of step B and step C, and another consists of step D and step E. All branches coming out of a branch point are executed, one after another. Therefore, assuming that no errors or test steps change the control flow and that each step generates exactly one output robot state, the preceding robot is executed as follows: A, B, C, D, E. However, it is important to note that step B and step D each receives a copy of the same output robot state produced by step A.

Branches can merge, and in complicated ways. Consider the following robot:



This robot illustrates how connections can be explicitly ordered. In this robot, the branches of step D are executed in the order specified by the numbers: step E is executed before step C. If an order is not specified (by numbers), connections are executed top-down. Thus, assuming that there are no test steps, that no errors occur, and that each step generates exactly one output robot state, the robot is executed as follows: A, B, C, D, E, C. The first time step C is executed, it receives the output robot state produced by step B; the second time step C is executed, it receives the output robot state produced by step D.

Sometimes you want to select (execute) only one of several branches, depending on circumstances. The Conditions and Error Handling topic shows how to do this.

Conditions and Error Handling

A robot may use different approaches in different cases. The cases may be distinguished either based on explicit tests; by evaluation of conditions, or because errors occur and need to be handled.

Note Examples in this topic are based on the Minimal Execution (Direct) design-time execution mode.

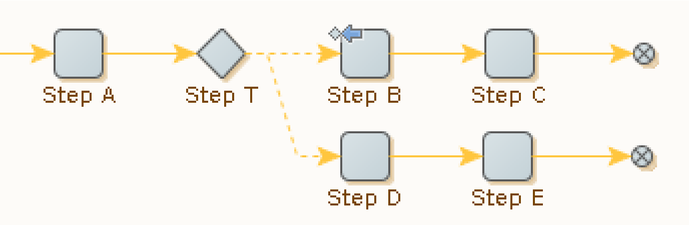
Conditions change the flow of execution based on the content of the input robot state (such as the presence of a particular tag in an HTML page). Error handling is about changing the flow of execution when particular errors occur (for example, some anchor tag is not found on the HTML page as expected and cannot be clicked). Often a situation can be seen both ways: An anchor tag should be clicked if found (this is a condition), or the robot can try and click it to handle the error (if it is not found). In some cases, what is commonly thought of as a condition is too complex to be written up as such (for example, a condition saying "if this particular page can be loaded without error"). In such a case, try and load the page and treat any error as an indication that the condition failed.

Other errors are signs of genuine problems with the robot or the web site being accessed. For example, the web site may be down and cause a page loading error, or a tag finder might fail to find a needed tag due to a dramatic page layout change of an HTML page. A particular error may be considered a failed condition in some circumstances, and a genuine error in other circumstances. The interpretation depends on the robot.

Because of this blurred boundary between conditional execution and error handling, Design Studio provides both features in a unified way. For every step, you can configure what to do in case an error occurs. Furthermore, steps with a test action (based on a condition of some sort) reuse the same approach, meaning that if the condition is not met, the (default) action is applied as if an error occurred.

For each step in the robot, you can configure the desired reaction to errors. Two useful error handling options are described here; see How to Handle Errors for information on the other options. The first option is closely linked to the Try step.

The Try step is similar to a branch point because it may have several branches going out from it. It differs from a branch point because branches beyond the first one are executed only if a step on the preceding branch encounters an error which it handles based on the Try Next Alternative option. Consider the following robot and assume that each ordinary step is expected to output exactly one robot state:



The ◇⬅ icon indicates that step B is configured to handle errors by "Trying Next Alternative."

If Step B executes successfully, step execution is as follows: "A, T, B, C." Because the first branch going out from T executes without error, the second branch is not executed at all.

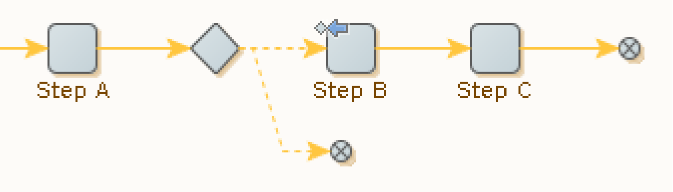
If, on the other hand, Step B encounters an error, then the execution of steps is as follows: "A, T, B, T, D, E." After the error in Step B is handled, execution does not continue at the following step, but instead at the beginning of the next branch going out from the Try step.

Each branch from a Try step represents one possible way to proceed from that point. Steps near the beginning of each branch probe if execution along the branch is a viable approach (and otherwise effect a "Try Next Alternative"), while later steps do the actual work when the branch turns out to be the right one for the case at hand. The probing steps near the beginning of a branch may be either test steps, or any kind of steps that, if they encounter an error, indicate that this branch is not the way to proceed. There may be any number of such branches going out from a Try step.

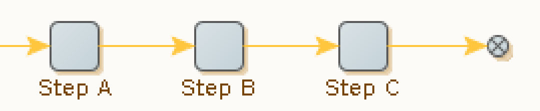
As with ordinary programming languages such as Java, JavaScript, C# or similar, the preceding robot is similar to an "if-then-else" construct: The first branch after the Try step contains the condition (the "if" part) and the "then" part, while the last branch contains the "else" part. Should there be more than two branches, then the ones between the first and the last ones are like "else-if" parts.

If the first branch attempts to do some action that may error, the example can also be likened to a "try-catch" construct: The first branch is the "try" part, while the second branch is like the "catch" part.

Another error handling option, Skip Following Steps, provides a more compact way of expressing a common special case, which is exemplified by the following robot. The step that can encounter an error is the first one on the first branch, and the second branch does nothing.



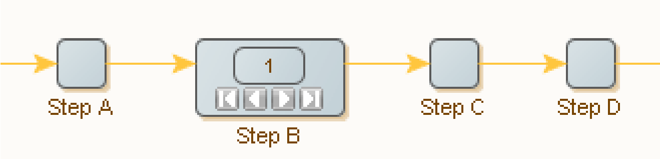
The effect is to skip execution of the steps after step B if it encounters an error. The same effect can be achieved without the Try step by using the error handling option "Skip Following Steps" (which is the default), in the following way.



Location and Location Code

When an error is handled, it is possible to report it back to the caller of the robot, or to log it. In both cases, a message is included that briefly describes the error, together with a location and location code for the step that encountered the error.

The location of the step that encountered the error is the list of steps (including iteration numbers) necessary to execute to reach that step from the first step. Consider the following robot.



If Step C reports an error on the second iteration of Step B, the location is written as: "step A - step B[2] - step C." Note that the location contains the step names and iteration numbers, separated by hyphens. Branch points are omitted.

The location code is similar to the location, but the name of each step is replaced by a unique identifier for that step, thereby avoiding name clashes. For the preceding location example, the location code may be: `{a-i1-a}`. Use the location code in Design Studio to go directly to the step that reported the error (using **Go To Location** on the **Edit** menu).

Important The iteration number in the location and location code is 0 indexed, so the first iteration is: `{a-i0-a}`

Snippets

A snippet is a group of steps that can be reused in several robots. A snippet is maintained in a file separate from the robot. Whenever the contents of a snippet is changed in one robot, it is automatically updated in other robots that uses the same snippet. A snippet is inserted into a robot using the Snippet step, and edited in-line. Snippets contents cannot be edited without being inserted into a robot.

The Snippet step inside a robot is in many ways similar to a Group step. Although, the steps inside a Group step are part of the robot, the steps inside a Snippet step are maintained in a separate file and can be reused in other robots inside the same project. A robot is incomplete and cannot execute if a snippet that it references is not present in the project.

After selecting a group of steps to convert to a reusable snippet, click "Create Snippet from Selection." If only a single group step is selected, it can be converted to a reusable snippet by clicking "Convert Group to Snippet." A snippet can be easily embedded into a robot by clicking the "Convert Snippet to Group" icon after selecting a snippet step.

A snippet can also define a set of variables included in the set of variables of any robot that uses the snippet.

A snippet can have a description. It is edited in the Snippet Editor and is shown on every occurrence of that snippet in robots.

Variables and Types

Variables and Types are important concepts in Design Studio.

Every variable can be associated with a default initial value that it retains unless the robot explicitly reassigns it, which it often will as values are extracted and manipulated during the execution. Most robots output the values of variables, by returning them to the caller or inserting them in a database. Robots can also take input values which are assigned to specific variables marked as receiving their values from input. These are called input variables.

Libraries and Robot Projects

Robots and types are organized in libraries. A library is a collection of robot definitions, type definitions and other files needed to execute the contained robots. A library serves as the deployment unit for robots. Use a library to bundle robots and their required files when you want to distribute and deploy the robots in a runtime environment, such as RoboServer.

In Design Studio, you can work on one or more robot projects at any time. The purpose of a robot project is to develop a robot library. A robot project contains the robot library that you are developing a given set of robots in, as well as other files that are useful for your work on the robot library. Files placed in the library may also be accessed by robots using a special library protocol.

Thus, a robot project is what you work on when you are developing robots, and a robot library is how you distribute and deploy your work.

Shared projects are deployed on a Management Console and connected to a project on your local Design Studio computer. Management Console projects can be shared between several Design Studios. The Management Console section in the My Projects pane provides visual indication of the status of the shared project files as well as tips with descriptions.

Secure Password Handling

Design Studio can be set up not to store license server and Management Console passwords on disk.

If you clear the **Remember Password** option for the license server, Design Studio does not store license server password and you have to type the password every time you start Design Studio with a license server. To open the Licence Server Information window, go to **File** > **Configure License Server**.

If you clear the **Remember Password** option for a Management Console on the **Management Console** tab of the **Design Studio Settings** window, Design Studio does not store the Management Console password. You have to enter the password every time you connect to the Management Console from Design Studio.

Note By default, the check for number of login attempts and wait time before the next attempt is disabled. To enable and configure this functionality, see "Check for login attempts" in Users & groups.

Password encryption in Kofax RPA

Kofax RPA implements the "bcrypt" password hashing function to use and store passwords in the program.

Design Studio User Interface

This topic introduces the Design Studio user interface and begins the tour to the following elements (or others):

- Menu Bar
- Toolbar

- My Projects
- Editors View
- Robot Editor
- Type Editor
- Text Editor
- Windows in Design Studio
- Status Bar

To view the Design Studio main window, you need a valid, activated license. See the *Kofax RPA Installation Guide* for details on licensing.

Menu Bar

The menu bar is located at the top of the Design Studio window.

The available menus and included items are based on the type of file that is open in the Editor view. The following menus are always available even if no file is open (some items may be disabled):

- The File menu includes items for manipulating files, robots, projects, and so on.
- The Tools menu lets you perform tasks related to the type of the file, such as generation of database table (for types), or deployment of robots to the Management Console (for robots).
- The Settings menu includes items for changing default settings and defining proxy servers or database connections.
- The Window menu includes items to change the layout of the user interface, such as Reset Layout and Save Layout.
- The Help menu includes links to the online reference help, documentation, and technical support information.

As soon as you open a file, such as a Web Automation robot, the Edit, View, Debug menus are to the available menus:

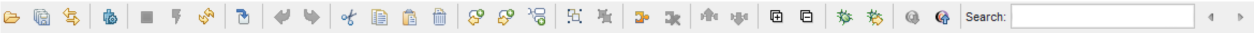
- The Edit menu offers a range of edit actions that you can perform on the opened file. The available actions depend on the type of the file, but it always contains the Undo and Redo actions.
- The View menu lets you perform actions on the view or open additional views that are not open by default.
- The Debug menu contains actions related to the debugger.

If you open a Web Automation robot file, change the mode to Debug to see the Breakpoints menu. The Breakpoints menu contains actions related to the breakpoints in the debugger, such as adding and removing breakpoints.

Most menu items become available when you click "Prepare Execution" to allow execution for a Web Automation robot. By clicking this action, you put the robot into execution mode, which enables you to execute it while editing. You can execute action steps right after you insert them in the robot workflow and immediately see the result. When a Web Automation robot is not prepared for execution, you can still perform some basic editing, such as add action steps and change their properties, configure variables, but you will not be able to see the result of step execution and to obtain and use data from websites that you interact with to design the robot. Also, a Web Automation robot cannot have the privilege to execute in both Design and Debug mode at the same time.

Toolbar

The Toolbar buttons let you perform many actions that are also available on the menus.



The available buttons change, depending on whether "Prepare Execution" is clicked for a robot, and on which editor is active in the Editors view. The execution privilege is intended only for Web Automation robots. Only one Web Automation robot at a time can have the execution privilege, so to take the execution privilege from one robot to another, open the tab with the required robot and click **Prepare Execution**.

Icon	Description
📂	Open Project
💾	Save All Files
⇆	Refresh All
⚙	Configure Robot
🟥	Stop - Escape
⚡	Prepare Execution
↻	Refresh
⤵	Step Into DA Robot
↩	Undo
↪	Redo
✂	Cut
📋	Copy

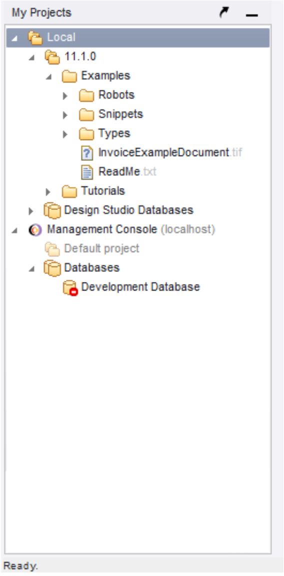
Icon	Description
	Paste Before
	Delete
	Insert step before selected step
	Insert step after selected step
	Add branch from selected step
	Group
	Ungroup
	Create snippet from selection
	Convert snippet to group
	Move step or connection up
	Move step or connection down
	Expand All
	Collapse All
	Switch to Debug Mode
	Start Debug from current location

Icon	Description
	Download robot from Management Console
	Upload robot to Management Console

For a list of buttons associated with Desktop Automation, see Edit Desktop Automation Robot .

My Projects

The **My Projects** pane is located under the toolbar in the Design Studio main window.



```
My Projects
◢ Local
  ◢ 11.1.0
    ◢ Examples
      ▸ Robots
      ▸ Snippets
      ▸ Types
        InvoiceExampleDocument.tif
        ReadMe.txt
    ▸ Tutorials
  ▸ Design Studio Databases
◢ Management Console (localhost)
    Default project
  ◢ Databases
      Development Database
Ready.
```

The **My Projects** pane consists of two main parts: **Local** folder and all running Management Consoles.

Local Folder

The **Local** folder shows an expanding/collapsing tree structure representing the local projects and databases that are open in Design Studio. Click ▸ or ◢ in this tree to expand or collapse the corresponding subtree. This folder can contain as many opened projects as you like. In the **Local** folder, you can right-click to open a context menu to perform various actions, such as creating a new robot in a folder or opening a previously saved robot.

When you right-click a file in the **Local** folder, the context menu shows the following actions.

Action	Description
Open	Opens the file in the editor.
Refresh	Refreshes all opened files.

Action	Description
Rename	Opens a Rename File dialog box for renaming the file.
Move	Opens a Move Files dialog box for changing the file location.
Copy	Opens a Copy File(s) dialog box for copying the file to a different location.
Delete	Prompts to delete the file.
Show in System Explorer	Opens the containing folder on your computer.
Create Robot Library File	Opens a dialog box for creating a robot library file. This option is available only for project folders. For more information, see the "Robot Libraries" section in the *Kofax RPA Developer's Guide*.
Upload	Opens a Upload to Management Console dialog box with available options for uploading the file. For more information, see Upload to Management Console.
Export Desktop Automation Robots	Opens an **Export Desktop Automation Robots** dialog box with available options for converting the action step to a robot. For more information, see Convert old Desktop Automation action step.
Migrate	Opens a Migrate dialog box for migrating a robot to a different browser engine. For more information, see Migrate a Robot to a Different Browser Engine.

The Design Studio Databases folder shows local databases for your Design Studio.

Management Console

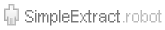
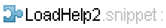
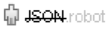
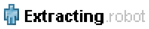
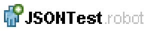
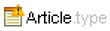
The **Management Console** section shows an expanding/collapsing tree structure representing the robot projects and databases for the Management Consoles that you are connected to.

When you right-click a file in the **Management Console** section, the context menu similar to that in the **Local** folder opens with a few additional actions.

Action	Description
Upload	Opens a Synchronize dialog box for uploading newer files to Management Console.
Download	Opens a Synchronize dialog box for downloading newer files from Management Console.
Synchronize	Opens a Synchronize dialog box for synchronizing with Management Console.

Action	Description
Exclude all from refresh / Exclude from refresh	Excludes either all projects of the selected Management Console from refresh or just one of them. Refreshing of all projects may take time, so we recommend that you exclude no longer required projects from refresh to experience best possible performance. By default, all projects are excluded from refresh and marked (excluded from refresh) in the **Management Console** section.
Include all in refresh / Include in refresh	Includes either all projects of the selected Management Console in refresh or just one of them.

If any projects in the **Management Console** section are shared with the Design Studio on your computer, only **Management Console** section contains those projects. Depending on the status of the files in the shared project, downloaded, updated, and deleted files have a different appearance. You can synchronize your local project with the Management Console project using different strategies. The following table shows project files with different statuses. The Management Console section also provides tips and explain synchronization problems.

Icon	Description	Meaning
 SimpleExtract.robot	Dimmed robot icon and name	An object in a shared project exists in the connected Management Console, but has not been downloaded to your Design Studio.
 LoadHelp2.snippet	Normal icon and object name	An object in a shared project is in sync with a remote Management Console.
 ~~JSON~~.robot	Normal icon with object name crossed out	The object is deleted in the project on your computer.
 **Extracting**.robot	Normal icon with name in bold	The file has been changed locally and needs to be synchronized.
 **JSONTest**.robot	Icon with a plus sign and name in bold	A new file in your local project is not yet uploaded to the Management Console.
 Article.type	Object icon has a yellow sign with an exclamation mark	Conflict exists between your local copy and remote project. For example, the object was deleted in a remote Management Console. When synchronizing, select how to resolve the conflict.

The Databases folder shows databases for the connected Management Console.

To configure connections to the Management Console, navigate to **Settings** > **Design Studio Settings** > **Management Consoles**

The databases are fetched via database mappings in the Management Console. To have the databases displayed in the Design Studio **Management Console** section, database mappings must exist for the cluster databases you want to share with Design Studio users. Unmapped cluster databases are not displayed in Design Studio.

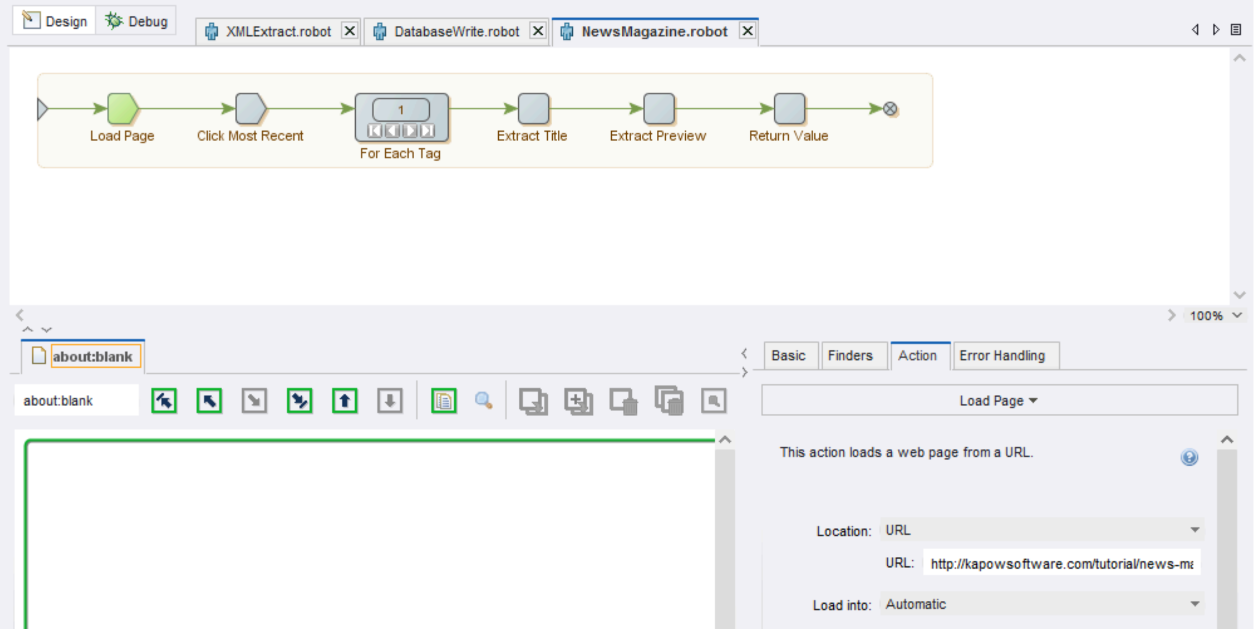
Important The database mappings, types, and drivers are fetched from a Management Console only when the connection between the Management Console and your copy of Design Studio is established or refreshed during the following events:

- Adding a Management Console connection to Design Studio
- Starting Design Studio with an existing Management Console connection
- Refreshing the Management Console connection (to refresh, select the Management Console node in the Management Consoles section tree and click **Refresh**).

Editors View

Use the Editors view to edit your robots and types. You can have many editors open at the same time, but only one editor is shown. The editors are shown as tabs at the top of the Editors view and you can click a tab to switch to another editor. There are three kinds of editors:

- The Robot Editor in which you edit a robot.
- The Type Editor in which you edit a complex type containing one or more attributes.
- The Text Editor in which you edit a plain text file.



This action loads a web page from a URL.

Location: URL

URL: http://kapowsoftware.com/tutorial/news-ma

Load into: Automatic

Robot Editor

Use the Robot Editor to edit Web Automation and Desktop Automation robots. When you open a robot, it appears in a new Robot Editor placed in a new tab on the Editors view. The Robot Editor has two modes: design (default mode) and debug. Select a mode by clicking a mode button in the left corner of the Robot Editor. Depending on which mode you select, the appearance and availability of options may vary.

For information on editing Desktop Automation robots, see Edit Desktop Automation Robot .

Each view consists of several subviews. For the design mode, the subviews are as follows:

- Robot View
- Applications View
- Step View
- Variables View
- Frames View

Also, when you work on a robot, various status messages can appear in the bottom left corner in Design Studio. For the meaning of these messages, see Status Bar.

Robot View

The Robot View is located at the top of the Robot Editor under the tabs. The Robot view shows you the robot program: the steps and connections that make up the robot. In this view you navigate the robot steps. Select a step to edit its structure such as delete, move, change, or connect steps.

Current Step

In the Robot View there is a notion called a "current step". The basic idea is that the partial robot that you are building is actually executed while you are building it. The current step marks the position in this execution and Design Studio shows the state in the Page view and the Variables view.

The current step is marked in green. Click a step to execute the robot up to that step. The selected step becomes the current step. While the robot is executing, the step you clicked is shown in yellow. When execution reaches the step, it becomes the new current step and appears in green. If execution cannot reach the clicked step (for example, an HTML page does not load), the execution stops at the valid step, which becomes the new current step. If you click a step the robot has already executed, no execution occurs, but the new step becomes the new current step. You always configure the current step in the Step View.

Current Execution Path

The Current Execution Path is the path in the robot that the execution performed to get to the current step. The robot continues on this path until it reaches a branch or an end node. The current execution path is marked by a darker color on the connections. You can change the current execution path by clicking a connection, which will result in the connection being included in the path.

Execution with No Path Restrictions

No Path Restrictions mode makes it possible to reliably reach any required step in a workflow when the execution cannot continue from the current step to the required one. This most frequently happens because of the restrictions imposed by other steps.

Select Items

To select a series of steps or connections, press and hold the Ctrl key and click the items. You can also hold down the left mouse button and drag it over the steps to select. Click anywhere outside the robot to deselect currently selected steps and connections.

When steps or connections are selected, you can apply actions to them. For example, to insert a new step, select a step and click the Insert Step After on the toolbar. You can also right-click a step or connection to select an action from a list.

> **Note** When you right-click a step or connection, it is automatically selected.

Edit Actions

The Robot Editor lets you perform a long range of actions on steps and connections. These include standard editor actions such as copy, paste, cut and delete, and actions that affect the execution of the robot in the design view, such as changing the iteration of a loop. You can perform actions on either the current step (if no other step is selected), selected steps or selected connections. Perform an action by clicking the corresponding toolbar button or by using the context menus on the selected elements.

You can configure another step by selecting it (by Ctrl-clicking it or dragging a selection box around it) and pressing the F2 key or selecting Configure Step on the context menu.

For more information, see General Editing.

Applications View

The Applications view is located under the Robot view in the Robot Editor. In the Applications view, you can see a part of the current robot state: the part of the robot state that has to do with loaded pages. The state shown is the input state to the current step.

In the Applications view, you see the Page views of the windows in the current robot state. When loading from a URL, several windows may be opened, each containing a page. The current window is marked with an orange box. If the opened page contains non-HTML content, you can preview the page depending on the type of the content. Use the Preview button to change the type of the content. You can preview CSV, JSON, text, Excel, XML, and binary content and apply step actions to them.

If you use the Classic browser engine, for each window, the Page view is split into several sub views depending on the type of the page. For example, if the loaded page is an HTML page, the Page view has sub views. There are five types of pages: HTML, XML, JSON, Excel and Binary. HTML and Binary use the same view and the other page types use their own specialized Page views.

Note To view XML content with the applied XSLT transformation in the Applications view, select **Configure Robot** > **Basic** tab. Click **Configure** next to **Default Options**, navigate to the **Legacy** tab, select **Classic loading** in the **Format Handling** setting and clear the **Convert XML to HTML** option.

To see the Cookies view of the state of the current step, you can open the Cookies window from the View menu. Cookies are added to this list as the robot loads web pages that use cookies.

Similarly, you can open the Authentications window from the View menu to see the authentications of the current state.

Step View

The Step View shows the configuration of the current step. Click the tabs to view and edit the following properties.

Basic Tab

This tab contains the name of the step and any associated comments. Steps with an attached comment are shown with a name in bold in the Robot View. You can rest the mouse pointer on a step to view the comment.

Finders Tab

This tab contains the tag or range finders used by the step to find the tags/ranges that the action of the step should use. You can configure the finders by right-clicking an element in the Page view. For more information, see Using the Tag Finders.

Action Tab

On this tab, you can view and configure the action of the step. For a description of the available actions, see Step Actions and Data Converters.

Error Handling Tab

This tab contains properties that control how to handle errors that occur during execution of this step.

Variables View

The Variables View includes a list of variables. When you select a variable from the list, the associated details appear on the right-hand side of the view. The view shows the variable values for the current step of robot execution, and cannot be edited.

- Right-click the variables list to access a list of variable types. You can add or remove variable types using this list. You can also remove the selected variable using this list.
- Click **Edit** to modify the initial variable values, or double-click an item in the variable list. A variable view similar to the Variables View window appears. This window displays the values of the variables before any step has been executed, and you can edit them.

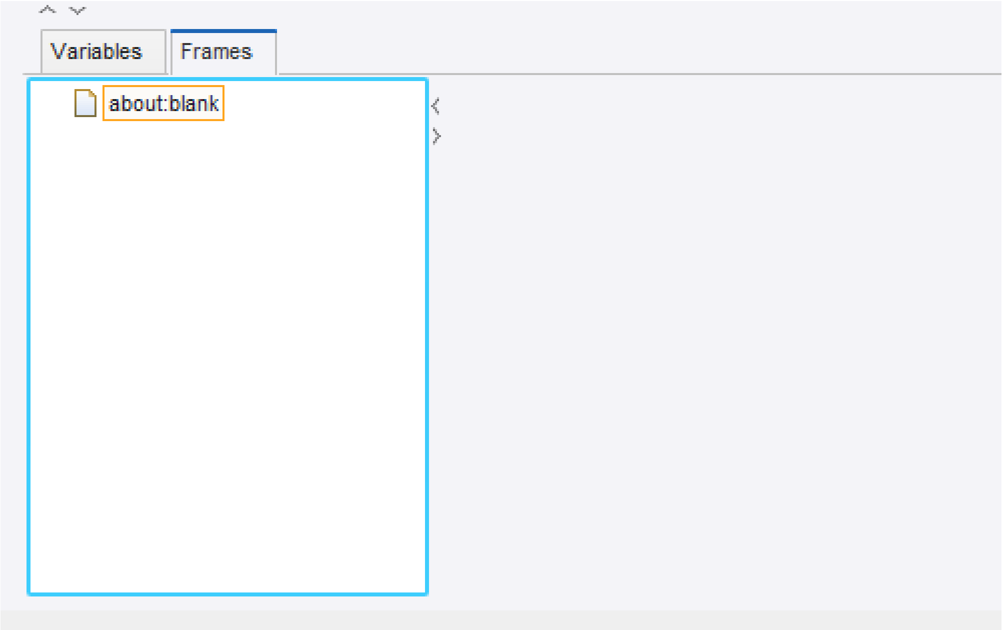
When writing and testing a robot, you use initial input variable values. When a robot runs in production, the input variables are initialized to values determined by the application running the robot.

Note If the application does not provide values, the robot run will fail.

Initial values for variables are the values that they have at the start of the robot (for example, at the first step). The values apply when you are writing, testing, and running the robot in production.

Frames View

The Frame view is located next to the Variables view at the bottom right corner of Design Studio.



The Frames tab shows all top-level browser frames and all their sub-frames in a tree. The view also contains a preview panel showing details about the selected frame. Starting from Design Studio version 9.6, the Frames view is the only place to get an overview of the frames. The labels of the top-level frames in the Frames tree are the same as the tab titles in the Page view. If the HTML page shown in the frame has a title, it is shown, otherwise the URL is shown. The labels of sub-frames are shown by their names (Unnamed (n) if they have no name).

A node in the Frames tree may have various decorations such as:

- An orange box around a label: the frame is the current window.
- A gray box around a label: the frame is currently selected in the page view.
- Light gray background color around the label: the frame is open in the page view.
- The label and the icon are dimmed: the frame has no view (its viewport is zero height or zero width).

The Frames Preview panel next to the Frames tree shows details about the selected in the tree frame. The details shown are the URL and a small rendering of the browser view of the frame with an overlay showing the size of the frame, for example 1263 x 1024. If a frame is blocked by URL blocking, then this is shown with ⦸ both in the preview and in the tree.

Note The Frames Preview panel is only available for robots designed with the Default browser engine.

Frame View Actions

There are a number of actions associated with the nodes of the frame tree.

- **Set as Current Window**: inserts a "Set Current Window" step into the robot which is configured to open the frame with the name of the selected node (the name is shown in the tooltip on the node).
- **Close Window**: inserts a step in the robot to close the frame
- **Open/Close**: opens or closes a frame in the page view (a tab). Only works on non-top level frames as the top-level frames are always open. Note that this command does not insert any step into the robot.
- **Block URL**: opens a dialog box for editing a URL blocking pattern for the frame and add this pattern to the list of Blocked URL Patterns for a robot
- **Select in Browser View**: selects the frame element in the browser view that defined the frame. If the frame containing the element is not open in the *Page* view, then this frame is opened.

Note These actions are also available on the browser view tabs of the page view.

Debug Mode

The Robot Editor contains a specialized mode for debugging robots. Click Debug or Design on the toolbar to switch between design and debug modes. This is also available on the Design Studio Main Window toolbar. Alternatively, to debug from the current step in Design Studio, click Debug.

The top of the Robot Editor in debug mode also contains a Robot view, similar to that of the design mode.

Note The Robot View in debug mode has a current step only when you are actually debugging the robot. This current step is not always the same as the current step in the Robot View in the design mode.

In the main panel, you see the results of the debugging process divided into various tabs.

- **Input/Output**: List of all used variables and all values returned during debugging.
- **API Exceptions**: List of API exceptions reported during debugging.
- **Log**: The processing log generated during debugging. Some actions, particularly those that take a while to execute, such as the Loop Form action, write status information to this log. Step errors are also logged if configured to do so.
- **State**: Whenever the debugging process is temporarily stopped, the State tab shows the robot state that is input to the current step. The State tab contains several sub-tabs.
  - **Variables**: Lists the variables.
  - **Window**, **Cookies**, and **Authentication**: Shows the state with associated dialogs.
  - **Local Storage** and **Session Storage**: Shows the HTML5 objects that have persisted locally.
  - **API Exception**: Generated at the current step. For all API Exceptions (and related errors), you can click the "Go to" button to navigate to the step (in Design mode) that generated the error. The step that generated the error becomes the current step in Design Studio.
- **Summary**: An overview of the number of variables returned or written to a database and generated API exceptions so far during the debugging process.
- **Stop When**: Specify the criteria required to temporarily stop the debugging process.
- **Steps to Skip**: Select steps to skip such as Store in Database, Delete from Database, Execute SQL, Execute Command Line, or Send Email.

For details, see Debugging Robots.

Type Editor

Use the Type Editor to create new types for a robot and configure the currently edited type. You can add new attributes to a type, remove attributes, change their order, and configure attributes using the buttons below the Attribute table.

For more information, see Configure Types.

Text Editor

The Text Editor is a simple editor for plain text files (.txt) such as Readme files. The allowed extensions that the editor can open are .txt, .java, .jsp, .js, .log, .html, .xml, and .csv. The editor does not use the information that these extensions imply about file content. All files are treated as plain text files (no syntax highlighting).

Windows in Design Studio

Windows in Design Studio are dockable: You can move, re-size, or dock them to an edge. This creates a custom layout that can be saved and used later.

- To undock a window, click the Toggle floating ↗ button. Drag the title bar of the floating window to move it. Click the ↙ button to dock a window.
- To save the changes, click **Window** > **Save Layout** on the menu.

All the saved layouts are shown in the list at the top of the **Window** drop-down menu.

- To upload a layout, click one of the layouts and select **Load Layout**.
- To delete an item from the list, click a layout and select **Delete Layout**.
- To reset the default layout of windows in Design Studio, click **Window** > **Reset Layout** on the menu.

Also note that Design Studio automatically saves the current layout on exit and resets when you open Design Studio again.

Status Bar

When you work on a robot, the following status messages can appear in the bottom left corner.

Status Message	Description	Shown in Mode
Ready	Robot is ready to run.	Design and Debug
Preparing execution	Robot is about to be executed.	Debug
Finishing execution	Robot is about to finish executing.	Design and Debug
Executing	Robot is being executed.	Design and Debug
Executing to start location	Robot is going to the currently selected step when started with the Start Debug from Current Location button from Design mode.	Debug

Status Message	Description	Shown in Mode
Start location reached	Robot reaches the step selected with the Start Debug from Current Location button from Design mode.	Debug
Start location could not be reached	Robot could not reach the step selected with the Start Debug from Current Location button from Design mode.	Debug
Execution completed successfully	Robot successfully executed all of the steps.	Debug
Execution completed with errors	Robot execution finished with errors.	Debug
Stopped	Robot is stopped after the Pause button is clicked.	Debug
Stopped after reporting API exceptions	Robot is stopped after an API error is reported.	Design and Debug
Stopped at breakpoint	Robot is stopped at a breakpoint set with the Toggle Breakpoint button.	Debug
Stopped at next visible step	Robot is stopped when the next visible step is encountered after the current position marker. This message is shown when the Single Step button is used.	Debug
Stopped at next step	Robot is stopped at the next step after the current position marker, even if the next step is hidden in a group. This message is shown when the Step Into button is used.	Debug
Stopped at a given step	Robot is stopped at a given step. This message is shown when the Step Out button is used.	Debug

Browser Tracer

The Browser Tracer is available from the Tools menu in Design Studio. The Browser Tracer can trace HTTP traffic in Design Studio as well as JavaScript execution for robots written with the Classic browser that was the default in pre-9.3 robots. The Browser Tracer is useful for performance tuning robots and figuring out workarounds for sites that do not work out-of-the-box.

Note The Browser Tracer is available for Web automation robots only.

Tracing

To start tracing, enable the Record button. While recording, especially if using JavaScript recording, things may run much slower than normal as vast amounts of data are collected. Make sure to disable the Record button once you have traced the traffic you wanted.

HTTP Trace

The HTTP trace shows HTTP traffic. Selecting a trace entry shows the details about that HTTP event in the detail view below the trace. The detail view shows the request and response headers, as well as the request and response data sent. Normally, only POST requests will contain request data.

> **Note** Pending loads are shown in blue.

Blocking URL

You can block certain URLs by right-clicking an item in the HTTP list and selecting **Add URL to block**. The Configure URL Blocking Pattern dialog box opens for you to edit the URL pattern. See URL Blocking for more information.

JavaScript Trace

JavaScript tracing is only available for robots that use the Classic browser that was default in pre-9.3 robots. If you have a robot that uses the Default browser, the JavaScript tab is not available.

Below each JavaScript trace, the JavaScript source code for the currently selected trace entry is shown. When a trace entry is selected, the corresponding source code line is highlighted in the source view. The trace entry is the runtime result of the execution of the highlighted source code line. Each source code line may, of course, be executed multiple times, in which case multiple trace entries are produced, all corresponding to the same source code line.

Stepping through trace entries can help you understand how a piece of JavaScript code works.

It is possible to turn off tracing of JavaScript if you are only interested in the HTTP trace. You do this by clearing the Enable JavaScript Tracing check mark in the Trace menu.

Saving and Loading Trace Sessions

You can save trace sessions to load at a later time. A trace session contains both the Design Studio trace and the proxy trace, and both JavaScript and HTTP traces. Saving a trace session can be useful if it is large and you want to look at it in detail later, or if you want to mail it to someone else.

> **Note** Bug reports submitted from Design Studio will automatically contain the current trace session, if any.

> **Note** If downloaded content is changed by "Page Changes" or "JavaSript changes", the browser tracer adds an extra line where the changed content is shown in the response tab. The URL starts with `Rewritten`.

Saving trace response

You can save trace responses by their path.

Click **File** > **Save** or **File** > **Save As** to open a **Save** dialog box and choose the directory to save all body responses.

All requests are saved by their path in the domain directory of the selected folder.

URL:

```
https://chedlink.chedraui.com.mx/Artus/g9/projects/main.php
```

Disc:

```
c:\somedir\chedlink.chedraui.com.mx\Artus\g9\projects\main.php
```

General Editing

This topic gives a few general hints related to editing robots in Design Studio. These hints apply to when you make changes to a robot in the Step View, to a type in the Type Editor or to a text in the Text Editor.

Copy, Paste, or Cut

Use keyboard shortcuts to cut, copy, and paste items In Design Studio.

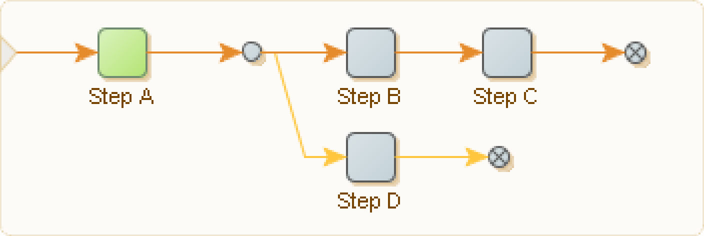
- **Ctrl-C** Copy
- **Ctrl-V** Paste
- **Ctrl-X** Cut

In addition, in most lists, such as the list of finders for a step, you can use **Ctrl-Shift-C** to copy all items in the list.

Group and ungroup steps

To group steps, select multiple steps and click Group on the toolbar. You can also right-click a step and select from the list.

Some selections cannot be grouped. A group step must have exactly one ingoing connection and exactly one outgoing connection, and this must also hold for the selection of steps that you want to group. The only exception is when a selection of steps does not have any outgoing connection. In this case, you can group the selection, but the topmost End step must be connected to the end of the group. Take a look at the following example.



In this robot, the following are examples of steps you can group:

- All the steps.
- Any single action step alone, such as Step A, Step B, and so on.
- The branch point, step B, step C and the end step after step C.

The following are some examples of steps you cannot group:

- the branch point and Step B (more than one ingoing connection)
- steps B, C, D and the two End steps (more than one outgoing connection)

You can add a step to the created group step by either dragging it into the created group step or by using the copy (cut) and paste option.

To ungroup a step or collection of steps, perform one of the following actions: either drag it out of the group step or collapse the group step, click on the collapsed group step and then click the Ungroup button on the toolbar (or select "Ungroup" in the context menu).

- Drag a step out of the group step.
- Collapse the group step, select it, and click the **Ungroup** button on the toolbar.
- Collapse the group step, right-click it, and select **Ungroup** in the context menu.

Note The Group and Ungroup actions are inverse. If you group a selection of steps and immediately ungroup them again, the structure of the robot is unchanged.

Use expand ⊞ and collapse ⊟ from the toolbar to perform the action on all groups.

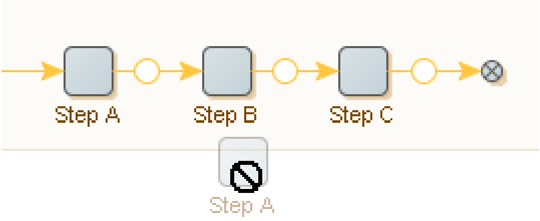
Use expand ⊞ and collapse ⊟ from the context menu to perform the action on the selected group or groups.

The Expand and Collapse options on the context menu on steps do the same, but they are restricted to the Group steps in the selection.

Drag and drop

In addition to actions, you can edit robot elements directly using drag and drop. As soon as you drag a step, special indicators appear showing valid drop locations. You can also select and move multiple steps at one time.

- To move a connection endpoint, select the connection and move the mouse to one of the handles at the end. Next, click the handle and move it to a new location. As soon as you click a handle, special indicators appear, showing where you may connect it.
- To abort a drag and drop action, move the mouse outside the robot and let go of the mouse button as shown in the following figure.



Add a new connection

You can also create new connections using the mouse. Place the cursor near the end of a step so that an indicator appears (an orange circle with a green halo). Click the indicator and a new arrow appears. Keep the left mouse button pressed; move the mouse and a new connection will follow your mouse when you move it. New indicators appear and you can move the mouse to drop the new connection end point by releasing the left mouse button.

Undo and redo changes

While editing a robot, you can undo and redo every action. Click ↶ or select Ctrl-Z to undo an action.

Similarly, click ↷ or select Ctrl-Y to redo an action.

Step validation

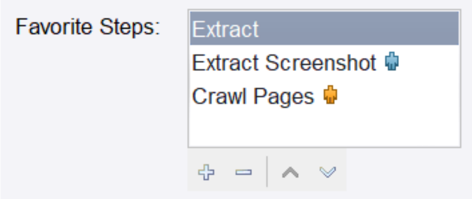
As you edit your robot, the Robot View validates each step. Invalid steps are underlined in red. You can move the mouse to an invalid step to view an explanation of error.

Add favorite steps to the Insert Step menu

While editing a robot, you might use some steps more than others. To minimize the time needed to insert your favorite steps, you can add them directly to the Insert Step menu. To add steps to the **Insert Step** menu, open the Robot Editor tab in the Design Studio Settings dialog box. To add steps to the list, click

+ under **Favorite Steps** and select steps. To remove steps from the list, select one or more steps and click −. To reorder steps, select a step and move it up and down the list using arrows.

Names of steps that work only with a particular browser engine are followed by a robot icon. Steps that work only with the WebKit-based browser have an orange icon, and steps that work only with the classic engine browser have a blue icon. Steps without icons can work with both browsers.



Keyboard help

Single-Line Text Fields

Description	Keyboard Shortcut
Move to prev/next word	Ctrl+Left, Ctrl+Right
Move to start/end of field	Home/End
Select all	Ctrl+A
Deselect all	arrow keys
Extend selection left/right	Shift+Left, Shift+Right
Extend selection to start/end	Shift+Home, Shift+End
Extend selection to prev/next word	Ctrl+Shift+Left, Ctrl+Shift+Right
Copy selection	Ctrl+C
Cut selection	Ctrl+X
Paste from clipboard	Ctrl+V
Delete next character	Delete
Delete previous character	Backspace

Multi-Line Text Fields

Description	Keyboard Shortcut
Move to start/end of line	Home, End
Move to prev/next word	Ctrl+Left/Right
Move to start/end of text	Ctrl+Home/End
Block move up/down	PgUp, PgDn
Block move left	Ctrl+PgUp
Block move right	Ctrl+PgDn
Select all	Ctrl+A

Description	Keyboard Shortcut
Copy selection	Ctrl+C
Cut selection	Ctrl+X
Paste Selected Text	Ctrl+V
Delete next character	Delete
Delete previous character	Backspace
Insert line break	Enter

Buttons

Description	Keyboard Shortcut
Activate	Spacebar

Drop-Down Menus

Description	Keyboard Shortcut
Toggle menu up or down	Alt+Up/Down

Multi-Line Text Fields

Description	Keyboard Shortcut
Expand entry	Right
Collapse entry	Left
Toggle expand/collapse for entry	Enter
Move up/down one entry	Up, Down
Move to first entry	Home
Move to last visible entry	End
Block move vertical	PgUp, PgDn
Block move left	Ctrl+PgUp
Block move right	Ctrl+PgDn
Block extend vertical	Shift+PgUp, Shift+PgDn
Select all	Ctrl+A
Single select	Ctrl+Spacebar

Lists

Description	Keyboard Shortcut
Move within list	Up, Down
Move to beginning of list	Home
Move to end of list	End
Select all entries	Ctrl+A

Description	Keyboard Shortcut
Cut	Ctrl+X
Copy	Ctrl+C
Copy all entries	Ctrl+Shift+C
Paste	Ctrl+V

Tables

Description	Keyboard Shortcut
Move to next cell	Right Arrow
Move to previous cell	Left Arrow
Move to first cell in row	Home
Move to last cell in row	End
Move to first cell in table	Ctrl+Home
Move to last cell in table	Ctrl+End
Select all cells	Ctrl+A

Projects and Libraries

When working in Design Studio, you can have any number of projects open at any time. The purpose of a project is to develop a library containing a collection of robots and the files required by these robots. Typically, you create a project for each separate usage of robots, such as one project for each application in your company that uses robots. Two projects cannot share files; a type always belongs to one project, and the scope of a type is the project it belongs to.

A project is a folder located anywhere in the file system. The project folder can have any name you want, but must contain the Library sub-folder.

Note See Naming Policy for more information about naming.

Library

This folder contains the library of the project.

Place all robot files, type files, and other files used by the robots, such as files that are loaded from the robot library in the Library folder. You can organize the files in the Library folder using subfolders as appropriate.

The following example shows a project folder named NewsAndStocksProject for a project that develops a robot library for extracting news from news sites and stock quotes from stock sites.

```
NewsAndStocksProject/
    Library/
       News/
          CNN.robot
          Reuters.robot
```

```
         News.type
      Stocks/
         Nasdaq.robot
         NYSE.robot
         Stocks.type
```

Note that this project has a Library folder with robot and type files divided into News and Stocks subfolders.

When you close Design Studio, it remembers the projects and files open. The next time you open Design Studio, it will open the same projects and files.

Current Project

In Design Studio you can work with many projects, but the other applications in Kofax RPA, such as RoboServer, always work on a specific project, referred to as the current project. When you install Kofax RPA, a default project is created. This project is selected as current. If you open Design Studio the first time, this current project is the only opened project. If you close all projects before you close Design Studio, the next time you open Design Studio it opens the selected current project.

You can change the current project selection using the Settings application, specifying the path to your new project folder in the Current Project Folder property in the Project tab, and then clicking OK to close Settings. Please see the *Kofax RPA Developer's Guide* for more information.

Shared Project

Shared project is a project that is deployed on a Management Console and connected to a project on your local Design Studio computer. Management Console project can be shared between several Design Studios, thus several people can edit a project. When your shared project is out of sync with the project on the Management Console, the Management Console section in the My Projects pane visualizes the status of each object in the project. You can use different strategies when synchronizing your local copy with one deployed on the Management Console.

Manipulate Robot Projects

Use the following procedures to open, close, and create projects.

- To open an existing project, on the File menu, select **Open Project** and select a project folder. The Open Project window appears.
- To close a project, in the My Projects view, right-click the project. The Project window appears. Click **Close**.

  You can also close all projects from the File menu.
- To create a new project, do the following:

1. On the File menu, select **New Project**.

   The New Project window appears.
2. Enter the name and location for the project.

3. Click **Finish**.

   A new project is created in the location you specified. The project folder name is the same name you assigned to the project.

   Example

   If you entered the name MyProject and the location: C:/KofaxRPAProjects then the following folders are created:

   C:/KofaxRPAProjects/MyProject

   C:/KofaxRPAProjects/MyProject/Library

Organize Robot Files

When you want to distribute and deploy your robot library in a runtime environment, such as RoboServer, you can pack the robot library into a single file called a robot library file.

This will pack together all files contained in the robot library of the file in the current editor and save the result as a single file with a name that you assigned. Before creating the robot library file, save all open files, such as robots and types, to include the most current changes.

You can make the robot library file available to RoboServer and execute robots from the robot library. See the *Kofax RPA Developer's Guide* for more information.

1. In Design Studio, save all open project files such as robots and types.
2. On the Tools menu, select **Create Robot Library File**.

   The Select Robot Library Output File appears.
3. Navigate to the location to use for your library.

   Use the icons on the toolbar to change to Details or List view, move up one level, or create a new folder.
4. In the File Name field, enter a name for the library.
5. Click **OK**.

   The system creates the robot library file.
6. Click **OK**.

Work with Shared Projects

Once you connect to one or more Management Consoles, the Management Console section in the My Projects pane displays all projects deployed on all Management Consoles that you have access to. If projects and objects are not downloaded to your local computer, they are listed but not available. Design Studio does not track such projects.

Upload a project to a Management Console

To upload a project to a Management Console, perform the following steps:

1. Right-click a project in the Local Folder and select **Upload** in the context menu, or select a project and click **Upload to Management Console** in the Tools menu.
2. Select the Management Console and project that the files will be uploaded to in the Upload to Management Console window.

   Click **Remember this (as a shared project)** if you want to keep this project as shared and synchronize it between the Design Studio and the Management Console.
3. Click **Upload** to complete the procedure.

After you upload a project to the Management Console, the project appears in the Admin > Projects tab and all project files appear in the Repository tab of the selected Management Console.

Download a project from a Management Console

To download a project from a Management Console, perform the following steps:

1. Right-click a project under the Management Console section in the My Projects pane and select **Download** in the context menu, or select a project and click **Download from Management Console** in the Tools menu.
2. Select the name for the project and its location in the Select Project Name and Location window.
3. Click **Finish** to download the project.

After you download a project from the Management Console, the project also appears in the Local Folder and you can edit the project files locally.

Synchronize Projects

After you edit the files in the shared project on your computer, you can synchronize your local files with those deployed on the Management Console. Because a shared project can be accessed by several people, you might come across a synchronization conflict. Design Studio provides messages and descriptions for you to understand what the conflict is, and how you can resolve it. Note that changed dependent files such as Types and Snippets can also prevent your robot from functioning properly. If you use Download to synchronize your project, the files are downloaded from the Management Console and your local changes are lost. If you use Upload, your local files are uploaded to the Management Console and any changes made by other people are lost (but those changes might as well be stored on their local computers). In any conflict situation, when changes made by you or other people can be lost, the Design Studio opens the Synchronize window for you to select the synchronization option.

The following table provides synchronization examples.

Status	Synchronization Option	Result
The shared project files have been edited on your computer. No others who have access to the same project on the Management Console edited the files.	Upload	Your changes are uploaded to the shared project in the Management Console. If you select Synchronize, the default option is to upload your changes to the Management Console.
The shared project files are changed in the Management Console. You know who edited the files and what the changes are.	Download	Changed files from the Management Console are downloaded to your local project.

Status	Synchronization Option	Result
You edited the shared project files on your computer. Someone else edited the files and uploaded them to the Management Console while you were editing the same files.	Synchronize	This is a conflict situation and you need to decide which changes to keep. In the Synchronize window you can select to either upload your changes to the Management Console, download the files from the Management Console, or just keep your files without synchronizing them with the Management Console.

Interact with Databases

You can use Design Studio to interact with databases. For details, see the following topics.

- Map Databases
- Types and Databases
- Database Warnings
- Create Database Tables
- Store Data in Databases

Map Databases

Robots may need to access databases through various database accessing steps (such as Store In Database). You must provide a reference to a named database for these steps. The named databases used by a robot must be accessible from RoboServers in order for the robot to be executed successfully on RoboServers.

While designing robots in Design Studio, it is convenient to use local databases that are not available from RoboServers. Rather than having to remember to change the named databases on the various database accessing steps before deploying a robot, Design Studio has an extra layer of abstraction to help overcome this problem: the database mapping. The mapping mechanism maps a named database in a database access step of a robot to a Design Studio database. As long as the robot is executed from within Design Studio, the named databases of the database accessing steps are mapped to the Design Studio databases specified by the mappings. The Design Studio user can use local databases while designing and testing robots without having to change the referenced named databases of the database accessing steps before deploying the robots.

Using database mappings also makes it easy for the Design Studio user to create the robot store values in a different database: it is a matter of reconfiguring the mapping to make it point to a different database.

A database mapping is a small configuration file defining which database to map to and whether Design Studio should display various warnings helping the user correctly configure the mapping and the referenced database. The name of the mapping is the file name of the configuration file. This means that if you create a mapping with the file name "objectdb," the database that the mapping points to will be accessible under the name "objectdb" in robots.

> **Note** The databases may have the same names across different Management Consoles, to distinguish them when creating database mappings in Design Studio a database name in the list includes a Management Consoles name.

The following steps show several ways to create a database mapping in Design Studio.

1. On the **File** menu, select **New Database Mapping**.

   A wizard appears.
2. Select a database and a project and click **Next**.
3. Enter a unique database mapping name and click **Finish**.

   When the wizard is finished, the mapping is created in the selected project and folder and is usable by robots.

Database View

1. In the database view, right-click the database to associate with a project.
2. Select **Add to Project** and select the project to add the database to.
3. Enter a unique name to use for the database mapping. This is the mapping file name and the name the database is accessible under.

   Notice that a name is suggested. This is the default database name, and the name used to access this database in other Kofax RPA applications apart from Design Studio.

Unmapped Database

In Design Studio, when you open a robot using a database you do not have a mapping for, a warning is displayed.

1. Open a robot using an unmapped database.

   A warning appears, recommending a mapping with the name of the database referenced in the robot. This allows you to quickly run robots sent to you from developers who have other databases defined - without modifying the robots.
2. Complete the steps in the wizard.

Types and Databases

If your robot writes the values of variables to a database, the types of these variables need to define which attributes must be part of the key used to store the values in the database. The database key for the value is calculated as a secure hash of the attributes marked to be part of the database key.

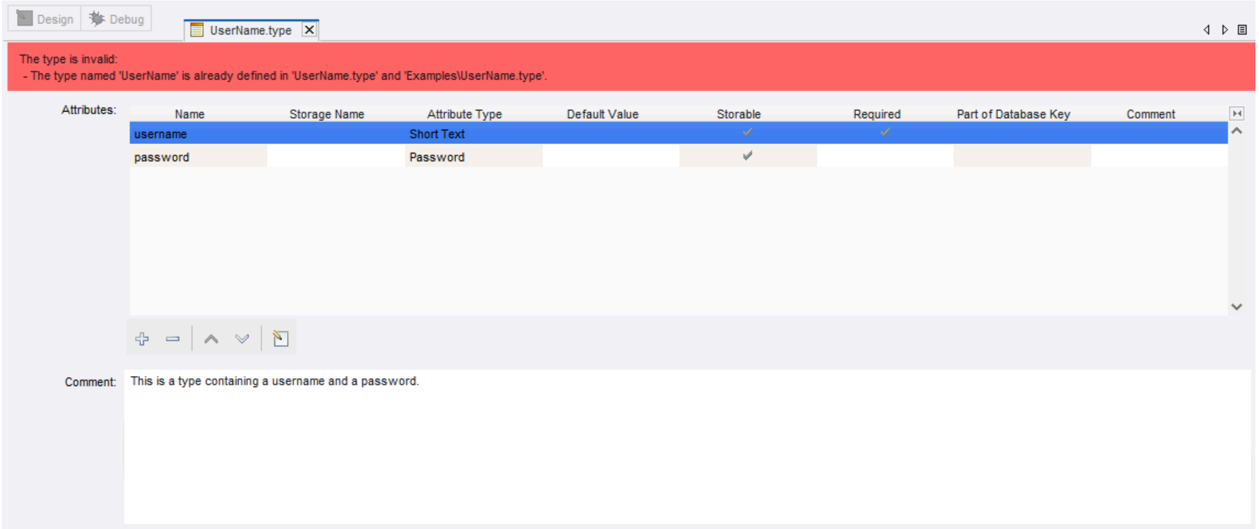
You can also specify a storage name as part of your attribute definition. This is an optional different name to use when storing the attribute.

When saving values to database storage using the Store in Database action, the appropriate database table must exist in an available database. The table is required to contain columns matching the attributes of the type.

See Create Database Tables for more information on how Design Studio can assist you in setting up the appropriate database tables. Consult Settings in Design Studio for more information on setting up database connections in Design Studio.

Database Warnings

Database warnings help you configure the database mappings, the robots and the referenced databases correctly. The warning system automatically monitors for potential problems such as type validation issues, missing tables, or missing database mappings and if an issue arises, a warning message is displayed in a status bar.



The type is invalid:
- The type named 'UserName' is already defined in 'UserName.type' and 'Examples\UserName.type'.

Attributes:

Name	Storage Name	Attribute Type	Default Value	Storable	Required	Part of Database Key	Comment
username		Short Text		✓	✓		
password		Password		✓			

Comment: This is a type containing a username and a password.

Also, the warning system will monitor which databases names are used in robots and assist in creating any missing mappings. The system performs a shallow monitoring of the databases, which means that it does not constantly ping the databases to see if they are online, but rather updates the information as needed. The system caches the table structures of the relevant database tables and uses this cache to compute any warnings to prevent an excessive number of database queries. The cache is recreated when Design Studio knows something has happened that requires it. Any external modifications of the database tables or the database availability are not monitored. There is the option of rebuilding the database cache for a single database or for all databases. This option is available through the database view, and through and through warnings, as applicable. For instance, to recreate the cache for a database, right-click it in the Database view and select **Refresh**.

Create Database Tables

To store extracted variable values in a database, you create matching tables in the database. Design Studio can assist in creating these tables by examining the types you have created, and generating the appropriate SQL. When storing the value from a variable of some type, tables representing that type must be present in the database.

> **Important** Do not use reserved names in your variable names to avoid database errors. See Naming Policy for details.

1. Open a robot or a type and on the **Tools** menu, select **Create Database Table** .

   The Create Database Table window appears.

   > **Note** If you use Kofax RPA built-in development database, start the database before creating database tables. To start the database, right-click the **Development Database** item under **Design Studio Databases** and select **Start Local Development Database**.

2. Select a database from the **Database** list. The list contains created databases. See Map Databases for information on how to create a database.
3. Select the type for which you want to create a database table.
4. Click **Generate SQL**.

   The system suggests a SQL statement for creating the tables.
5. Select **Drop table if exists** if you want to recreate tables in the existing database by including a 'DROP TABLE' statement before each 'CREATE TABLE' statement to make sure the corresponding table is removed if it already exists.
6. Modify, execute, or save the statement.

   The SQL shown is a recommended suggestion: you can change the statement to fit your needs, if required. For example, you might change the column type for a Short Text attribute from "VARCHAR(255)" to "VARCHAR(50)" to conserve database space, or you could add an auto-incrementing primary key. However, under normal circumstances, you should not modify the table name or any column names, or remove any of the columns.

Store Data in Databases

This section explains how Kofax RPA database storage works.

Object Keys

The tables you create for a type in a database have a column for each of the attributes in your type, plus an additional 7 household fields, named: ObjectKey, RobotName, ExecutionId, FirstExtracted, LastExtracted, ExtractedInLastRun, and LastUpdated. The most important field is ObjectKey, as it is the primary key for the table.

> **Note** The reason for the name "ObjectKey" is found in the terminology previously used in Kofax RPA. Previously, types and variables were called "objects." To adhere to the new terminology, "ObjectKey" should be called "ValueKey." Renaming it would cause quite a lot of backward compatibility problems, though, and therefore it has been allowed to keep its old name.

The ObjectKey for a type is what uniquely identifies values extracted from variables of that type when stored in a database. You have to figure out what uniquely identifies values of the type. If you are building a car repository, the VIN number may be enough to provide unique identification of each car. If you are collecting baseball results, you may need the year, team names, ballpark, and date to uniquely identify each match.

As you build the type you can select how the ObjectKey is going to be calculated. This is done by checking the "part of database key" option when creating a new attribute. For our car example, the VIN number would be the only attribute marked as part of the database key, for the baseball match example, the attributes year, team names, ballpark, and date would all be marked as part of database key.

The robot developer may also specify the key directly on the Store in Database action, to override the default algorithm defined on the type.

The attributes that are not part of database key are sometimes referred to as non-key fields. For example, the car might have a price attribute, but even if the price changed we would still consider it the same car.

Store in Database

Kofax RPA provides three actions for managing values in a database: Store in Database, Find in Database, Calculate Key, Delete from Database, Query Database , and Execute SQL. The Find and Delete actions are simple, but Store in Database does more than just store the value.

Store in Database may insert a new value into the table, or update an existing value that was previously stored. Here is a list of exactly what happens.

1. When storing the value of some variable, the ObjectKey is calculated based on the variable's values of the attributes which in the variable's type are marked Part of Database Key. If the robot developer specifies a key on the action, this key is used instead.
2. Using the calculated key, a check is made to see if the value already exists in the database.
3. If the value does not exist, a new row is inserted into the database (under this ObjectKey).
4. If the value already exists, it is updated, and all the non-key attributes are written to the table (under this ObjectKey).

Household fields

Whenever a value is inserted all the 7 household fields are updated. On update, only some fields change. The following table provides an overview.

Field	Description	Changed on
ObjectKey	The primary key for this value	Insert
RobotName	The name of the robot that stored this value.	Insert and Update
ExecutionId	The execution id for the robot execution that stored this value.	Insert and Update
FirstExtracted	The first time the value was stored.	Insert
LastExtracted	The last time the value was stored.	Insert and Update
LastUpdated	The date when the value was last updated.	Update

Field	Description	Changed on
ExtractedInLastRun	If the value was extracted in the latest run (uses 'y' and 'n').	Insert and Update

After each robot execution (in which the robot used Store in Database), all values previously collected by this robot, but not stored during this run, will have ExtractedInLastRun set to "n" and LastUpdated set to "now", indicating that the value was not found on the website during the latest run.

Note If a value was found in the previous run, but no non-key fields have changed, then LastUpdated is not updated. However, if the value was not found in the previous run, but in a run prior to that, LastUpdated is updated even if the non-key fields have not changed. This means that the value was deleted from the site and then reappeared later.

Harvest Tables

The tables created by Kofax RPA are often referred to as harvest tables, as the robots are harvesting data into them.

To find out what information was available on a website the last time the robot was run, you can use the following SQL command:

SELECT * FROM table WHERE ExtractedInLastRun = 'y'

If you are running queries against a table at the same time as a robot is storing data into the table, the result is comprised of data from the previous run, mixed with whatever data the executing robot has stored so far. We recommend that you copy the data out of the harvest tables and into a different set of production tables, so you can run queries against a stable data set.

There are many solutions where robots are used to store data in a database, but most of them fall under one of the three scenarios listed in the following table.

Scenario	Description
Repository matching website (small data sets)	The idea is to have a repository that matches the items on a website 1-to-1. The easiest way to accomplish this is have a truncated production table (deleting all rows) every time the robot is done executing, and then copy every record where `ExtractedInLastRun='y'` from the harvest table into this table. This works well for small data sets.

Scenario	Description

Repository matching website (large data sets)	Same as above, but the data set is too large to copy all data after every robot execution. Instead we want to update the production table after each robot execution, based on the changes that occur.
	This is where the LastUpdated field comes in handy. All values that have been updated have a LastUpdated field value larger than the start time of the robot. You can get the start time from the database logging tables, or you can have the robot store it somewhere.
	To detect deleted values, use the following command:
	`SELECT * FROM table WHERE LastUpdated > 'StartTime' AND ExtractedInLastRun = 'n'`
	To detect new values:
	`SELECT * FROM table WHERE LastUpdated > 'StartTime' AND ExtractedInLastRun = 'y' AND FirstExtracted > 'StartTime'`
	To detect updated values
	`SELECT * FROM table WHERE LastUpdated > 'StartTime' AND ExtractedInLastRun = 'y' AND FirstExtracted < 'StartTime'`
	Then update your Production table accordingly.

Historic data	The default setup allows you to see when a value was first extracted and when it was last updated, but you cannot see which run of the robot the value was found in. In this case, you should copy all the data from your harvest table into another table after the robot run, but in your new table the ObjectKey should not be a primary key. Instead, create an extra column called RUN_ID and use it together with the ObjectKey to create a compound primary key. If you do not need a RUN_ID you could simply create an auto-incremented column and use that as the primary key of your secondary table. Truncate the harvest table before each run.

You do not have to copy all the household fields to your production table; only the ObjectKey is required for you to update your production tables

Concurrency Considerations

If you have multiple robots storing values of the same type to the same database, be aware of the following considerations.

- Every time a value is stored, the RobotName column is updated. If you have two robots storing the same value (as identified by ObjectKey), only the last one will show after the robots are done executing.
- If two robots store the same value at exactly the same time, you get an error. They both find that the value is not in the table and try to insert it, but only one of them will succeed. In most cases, the error can be ignored because it is the same value.
- If you run the same robot twice at the same time and the robot stores data in a database, you break the way the ExtractedInLastRun column is used. When the first robot is done executing, it updates the ExtractedInLastRun to "n" for all values it has not stored. This includes all values stored by the second robot so far. Later when the second robot finishes, it sets ExtractedInLastRun to "n" for all values stored by the first robot, completely negating the first run.

Value Relations

The storage system does not provide an automated way of managing relations between values. If you have a value of type Person and one of type Address, and you want to link them, you have to maintain this link.

The easiest way to create a link is to have the ObjectKey of the Person value be a foreign key in the Address value that should be linked to this person.

If the ObjectKey is calculated automatically from the type, you can use the Calculate ObjectKey action to generate the key and assign it to each of the address values before you store them.

You should be careful when building robots with connections between stored values. If an error occurs when you store the Person value, make sure that no Address values are stored.

ObjetKey Caveats

If you are using MySQL or Oracle, review these important ObjectKeys rules.

- On Oracle empty string is stored as null.
- MySQL does not have millisecond precision in timestamps.

These two cases result in a potential loss of data when the data is stored in the database. The ObjectKey is calculated inside the robot based on the data in the given variable. If you later load the value from the database and try to recalculate the ObjectKey, the ObjectKey is different if data loss occurred in any of the attributes marked as part of database key.

Configure Types

In the main view of the Type Editor, the various properties of a type can be edited. This includes the attributes of the type, a comment attached to the type (optional), and a storage name (optional).

A type must have a valid name. This name is simply the file name of the corresponding type file. This name must contain only letters, digits, and underscores, and must begin with a letter or an underscore. Additionally, the name must be unique within the project, including the name case. For example, a single project cannot contain two types named "MyType" and "mytype". The name of a type is used as storage name unless a storage name has been specifically configured (see below).

A type contains the following properties which can be configured in the Type Editor:

Attributes

The attributes that are added to the type are shown in the table. To add a new attribute, click the **Add and Configure New Attribute** button below the table. To remove an attribute, select the row with the attribute, and click the **Remove Attribute** button. To configure an attribute, click the **Configure Attribute** button. When you add or configure an attribute, the Attribute Configuration dialog will automatically pop up. Note that to conserve space, not all columns in the table are shown by default. To change which columns are shown, right-click on any of the column names.

Comment

Here you can add an optional comment to the type. The comment is only displayed in the Type Editor.

Storage Name

Here you can set a name to use when storing values from variables of this type, e.g. the table name in a database, or the tag name in XML. If you leave this field blank, the name of the type will be used instead for storage. The intended use of the type may put some constraints on the storage name; for example, if values of the type are going to be stored in a database, you must avoid using a storage name which is the same as some keyword in the database you intend to use.

Attribute Configuration

Basic Tab

This tab contains the basic properties of an attribute.

Name

The name of the attribute. The name must be unique within the type, including the name case. For example, a type cannot contain two attributes named "Location" and "location". Also, the name must contain only letters, digits, and underscores, and must begin with a letter or an underscore. Additionally, if no Storage Name is set on the attribute, the Name will be used as Storage Name (row name in a database, column header in a CSV file, or tag name in XML). Depending on the intended use of the type, this may impose additional constraints on the name; for example, you may have to avoid naming the attribute the same as a keyword in the database you intend to use.

Type and Default Value

Choose the attribute type of the attribute from the list of attribute types, and set the default value for the attribute.

Required

If checked, this option serves two purposes:

- Variables of the type will not be stored (in a file or database) if the attribute does not have a value (an exception is thrown). Note that if a value is not supplied for a non-required attribute, nothing is stored in a file either, though there is no exception or error thrown.
- Input variables of the type must have a value for the attribute or the robot execution will not start.

Part of Database Key

When storing values of a type in a database, they need to be stored under a key. The key for the value is calculated as a secure hash of the attributes that are part of the database key. Choosing a good key is important when storing values in a database. You should make sure that the key attributes you select are unique across all the values of the type. Good examples of keys are product numbers and URLs. If you have stored data in the database, you should be very careful when changing this option. Any change will cause all the robots to be unable to refind (update) existing values in the database. If you have the correct setting for most of your robots but a single robot requires a different key calculation then you should change the key field on the steps in the robot.

Comment

In this field, you can add an optional comment for the attribute, which can be helpful for describing the attribute in detail.

Advanced Tab

This tab contains the advanced properties of an attribute.

Storage Name

This is an optional different name to use when storing the attribute, such as the row name in a database, the column header in a CSV file, or the tag name in XML. If you leave this field blank, the value from the Name property will be used automatically for storage. The intended use of the type may put some constraints on the storage name; for example, if values of the type are going to be stored in a database, you must avoid using a storage name which is same as a keyword in the database you intend to use.

Visible

Select this option if the attribute should be visible in robots in Design Studio.

Storable

Select this option if this attribute should be stored when storing values of the type.

Show Separator Before

This option is marked if a separator should be shown before this attribute when the type is used in robots in Design Studio.

Separator Title

The name of the separator.

Type Attributes

The attributes within a type must also be correctly added and configured in order for the type to be valid. You must specify both a name and a type for each attribute. The available attribute types are listed in the following table.

Attribute Type	Description
Binary	Binary data; any sequence of bytes.
Boolean	A boolean value; either "true" or "false".
Character	A single character, such as "A".
Country	A country code, as defined by the ISO-3166 standard, such as "DE" for Germany.
Currency	A currency code, as defined by the ISO-4217 standard, such as "EUR" for Euro.
Date	A date, which must use the form yyyy-mm-dd hh:mm:ss.n, such as "1992-04-25 10:33:06.0".
Excel	An Excel document. This is the same as Binary Data, except that you can preview the document and perform basic Excel operations on it.
HTML	An HTML clip. This is the same as a Long Text, except that you can preview the clip in a browser window.
Image	An image. This is the same as Binary Data, except that you can preview the image.
Integer	An integer, such as 12. The possible range is from -9223372036854775808 to 9223372036854775807, both inclusive.

Attribute Type	Description
JSON	A JSON value is either a JSON text or JSON Simple type where the JSON Simple type is either a JSON literal, a number, or a string.
Language	A language code, as defined by the ISO-639 standard, such as "de" for German.
Long Text	A long text. Displayed in a multi-line text box.
Number	A number, such as 12.345. The possible range is from ±2.2×10−308 to ±1.8×10308 with slightly more than 15 digits of accuracy.
Password	A password. Displayed in a password field that shows asterisks instead of the characters in the password.
PDF	A PDF document. This is the same as Binary Data, except that you can preview the PDF document.
Properties	A text that represents a list of properties where each property is a name/value pair. For more information, see Properties Attribute Type .
Session	A session (containing cookies, authentications, and so on).
Short Text	A short text. Displayed in a one-line text field.
XML	An XML document. This is the same as a Long Text, except that only well-formed XML documents are allowed.

Properties Attribute Type

An attribute of the Properties attribute type contains a text that represents a list of properties, where each property is a name/value pair.

The Properties attribute type is useful for representing a list of properties that may vary dynamically. If you have a fixed set of properties, you should normally represent each property as an attribute instead.

Here is an example of a list of properties that an attribute of the Properties attribute type can contain:

"productName" = "Hydraulic Valve" "productNumber" = "53563-433" "productVendor" = "American Valves Inc." "productWeight" = "3.45" ...

Each property must be on a separate line. A property line must consist of the property name, followed by a '=', followed by the property value. A specific property can occur at most once in the list. The name of a property cannot be empty, but the value can.

The name and value may be specified with or without quotes. When you use quotes, you can use the backslash character (\) to enter special characters:

- \n for line break.
- \r for carriage return.
- \f for form feed.
- \t for horizontal tab.

- \b for backspace.
- \" for double quote.
- \' for single quote.
- \\ for backslash itself.
- \uxxxx for the unicode character with encoding xxxx, where xxxx is four hexadecimal digits.

If you do not use quotes around the name/value, all spaces at the start and end of the name/value will be removed, you cannot specify an empty value, and you cannot use the backslash notation to enter special characters.

The list of properties can contain empty lines and comment lines. A comment line starts with two forward slash characters (//).

Configure Variables

When creating a new Web Automation robot, you usually start by configuring its variables. Of course, you can reconfigure the variables at any time during the lifetime of a robot, such as changing the initial value of a variable.

Desktop Automation robots can take input from a Web Automation robot and use the same variable types with some additions.

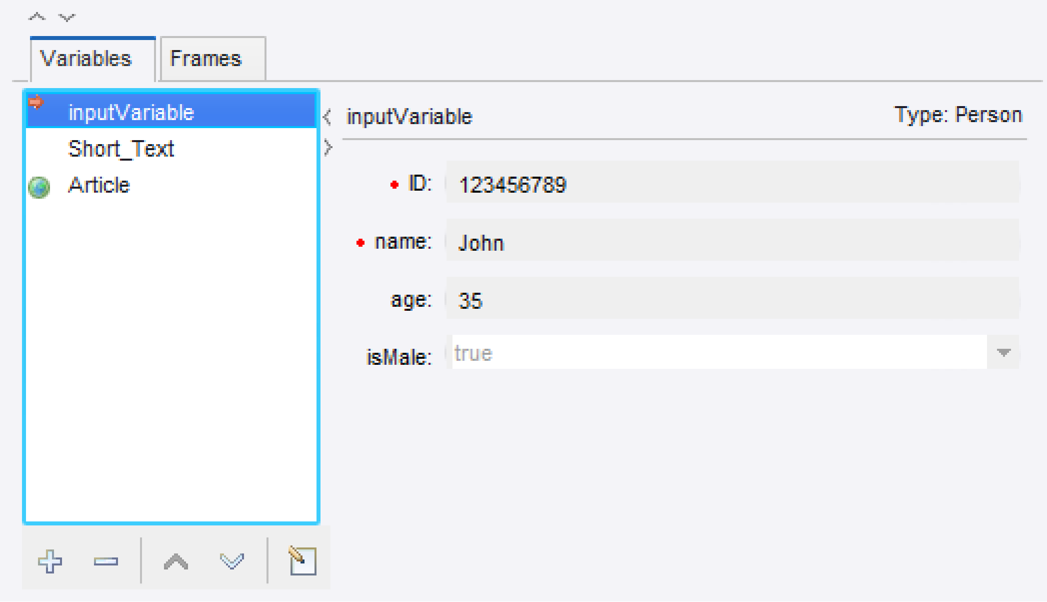
1. In the Robot Editor, below the Step View, select **Variables View**.

   The variables you specify become part of the robot state given as input to the first step of the robot.

   The Variables View shows a list of variables, together with details on the selection. The icon next to the variable indicates the variable type as follows:

   - Input Variable ➡
   - Global Variable 🌐

   The following Variables view contains three variables: one input variable, one normal variable, and one global variable.

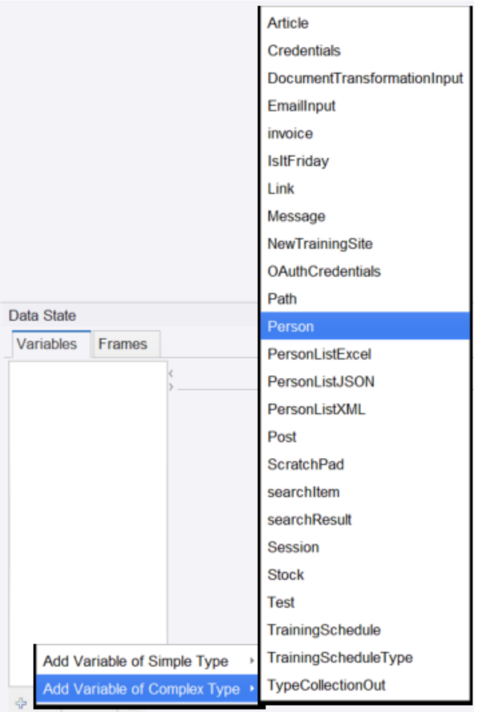
   inputVariable
   Short_Text
   Article

   inputVariable Type: Person

   • ID: 123456789
   • name: John
   age: 35
   isMale: true

   The Variables View shows the values of the variables at the current step.

2. To add a new variable, click **Add** ✚, or right-click the Variable and select a type.



The Edit Variable window appears.

Note If the variable is added using the right-click method, where a type is already selected, the window opens with the pre-selected type.

Note This is also the window used to configure an existing variable, either by double-clicking it or by clicking the ✎ button.

3. In the Edit Variable window, enter a name for the variable.

   The name must adhere to naming standards. For example, spaces are not allowed. When you click OK, you are notified if the name is invalid. Change an invalid name or click **Cancel**.

   Note Use the variable configuration window to edit initial values. In other words, this dialog box does not, as the Variables view, show current values. The values you provide are used at the start of the execution.

4. Select a variable type.

   See Variables and Types for more information about types and their connection to variables.

5. Complete the input fields based on the type.

   Use these fields to provide the variable with initial values. You do not have to manually give the variable a name.

6. Click **OK**.

   If you have not entered a name, you are prompted to generate a name from the type name.

7. Use the **Global** and **Use as Input** check boxes to configure a variable as input to the robot or global.

   If a variable is used as input, it makes it possible to supply the robot with values for that variable when running it on a RoboServer. For input variables, the values entered for attributes should be regarded as test input and used only when you are working with the robot in Design Studio. When the robot is run on RoboServer, the input values are overridden (replaced) by values supplied by the client that runs the robot. Note that variables of simple types cannot be used as input, as their usage is as temporary variables, which are internal to the robot.
8. If you want the variable to keep its value during the entire robot execution, select **Global**.

   Global variables provide a way to create counters and do other kinds of computation across iterations and branches. Global variables can also be used for accumulation of data across iterations or branches, such as accumulating a text consisting of comma-separated values.

   This is different from normal variables, whose values are not kept across loop iterations and branches.

   > **Note** In Design Studio, the values of the global variables depend on which steps you have executed to get to the current step. Unless care is taken to execute the right sequence of steps, the values are different from when the robot is actually executed.
9. To remove a variable, right-click the variable and select **Remove**.

   Alternatively, select a variable and click **Delete** − below the list.

Variables

You define each variable as either complex or simple.

Simple Variable

A simple variable does not define any attributes, but only represents the type of a single value. Thus, a variable of a simple type contains a single value, for example a text string, and is referred to only by its variable name, such as Username. Simple types are built-in and cannot be edited, or created.

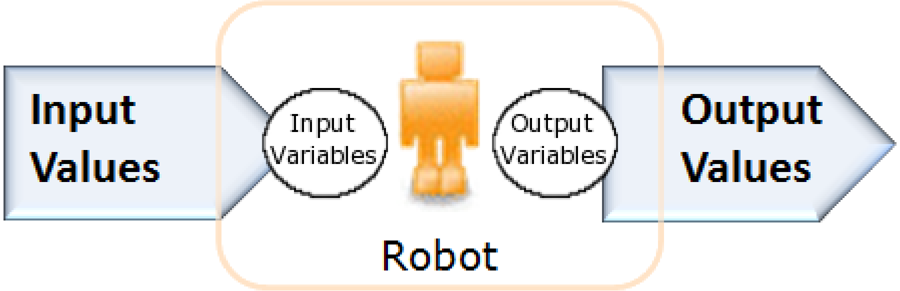
- Useful when extracting temporary data or as global counters.
- Commonly used as temporary variables, internal to the robot.
- You cannot use a simple type for input variables.
- You cannot output the value of a simple type.

Complex Variable

A complex variable defines a set of attributes. Each complex variable denotes several (named) values. We generally refer to each attribute such as "title" as a separate variable such as "Book" and denote its value using the fully qualified attribute name, such as Book.title. You can create complex types within Design Studio to suit your needs.

Complex variable values are output in various ways. For example, a robot extracting news from a web site might output the values of news variables; each news variable would have a complex type with attributes such as headline, bodyText, date, and author; and each news value to output would comprise a possibly unique subvalue for each named attribute.

For robots containing input variables, they must be specified as part of the robot input with values assigned to the input variables. For example, a shopping robot that orders books at http://amazon.com might depend on input values containing user and book information. These might be assigned to two input variables in the robot called "user" and "bookInfo" of type "User" and "BookInfo." The following figure shows how a robot accepts input values and generates output values.



The figure shows robot input-output. Input values are assigned to input variables, and the values of some variables are output. Only variables of complex types can be assigned from input or have their values output.

Variable Validation Errors

Validation errors can occur on variables using types that are changed, renamed, or deleted. This topic explains how to deal with these errors.

If a type used by a variable is missing, a validation error informing that the "type could not be found" is shown. Resolve the error by creating a type with the name of the missing type, or by opening the Configure Variable window and selecting another type for the variable.

A configuration problem with a type used by a variable can also result in a validation error on the variable informing that it has an "invalid type." This problem is solved by rectifying the problem with the type or selecting a different type for the variable from the Configure Variable window.

The validation error informing that some of the variable's initial/test values are "incompatible with its type" is a bit more subtle, and not as self-explanatory as the preceding errors. The error occurs if a variable has some initial/test values assigned to attributes and the variable's type is then changed such that one or more of the values can no longer be assigned to the attributes. This happens if the attribute is deleted from the type, or if the attribute type is changed so that the old assignment value is now incompatible. For example, if an initial value is assigned to an attribute with the attribute type Short Text, but the variable type is changed so that the attribute now has the attribute type "Boolean," the old value can no longer be assigned to it. To fix the problem, you need to clear the non-assigned values. Open the Configure Variable window and then exit it by clicking OK. A prompt appears, informing that the non-assigned values will be discarded. After you click OK again, the validation error is resolved.

Design Studio Settings

To open settings for Design Studio, on the menu, click **Settings**. Use the following tabs in the Design Studio Settings window to set preferences for Design Studio:

- General Settings
- Text Files
- Robot Editor

- Desktop Automation
- Local Databases
- Proxy Servers
- Certificates
- Bug Reporting
- Management Consoles

General

When you open the Design Studio Settings window, the General tab appears by default. Use this tab to set general preferences for Design Studio.

The following table describes options on the General tab.

Option	Description
When switching to Design Studio	Indicates what happens when the user switches to Design Studio.
Default execution mode	Specifies the default robot execution mode for all newly created robots.
Maximum number of recent projects	Lists the maximum number to include in the local history of recent Design Studio projects. Users can access the list by selecting **File** > **Recent Projects**.
Maximum number of recent files	Lists the maximum number to include in the local history of recent Design Studio files. Users can access the list by selecting **File** > **Recent Files**.
Reopen projects on startup	When selected, the most recent project is reopened when Design Studio is started.
Show welcome screen at startup	When selected, the Welcome screen is shown when Design Studio is started.
Create backup files	When selected, backup files are created whenever a saved file is changed. Backup file names end with a tilde (~) character.
Documentation location	When selected, the Kofax RPA documentation set is available for use in offline mode.
Show documentation retrieval notifications	When selected, the "Retrieving help and documentation" warning is displayed if you try to access the online documentation from Kofax RPA without Internet access.

Text Files

Use the "Text files" tab to set preferences for text files used in Design Studio.

The following table describes options on the "Text Files" tab.

Option	Description
Default file encoding	Specifies the default encoding of text files.
Default line separator	Specifies the default line separator in text files.
Default tab size	Specifies the default tab size in text files.

Robot Editor

Use this tab to set preferences for the Robot editor.

The following table describes options on the "Robot editor" tab.

Option	Description
Show tooltips on steps	When this checkbox is selected, tooltips are shown for robot steps.
Show error handling on steps	When selected, robot steps with custom error handling are marked with a red exclamation mark.
Default zoom factor	Assigns the zoom factor to apply when robots are opened in the robot editor. You can manually adjust the zoom factor from the lower right corner of the robot editor.
Source view font	Specifies the point size of the text font in the source view (the bottom section of Design Studio).
Favorite Steps	Lists steps that show up directly on the **Insert Step** menu that is available when you right-click a connection in the Robot view in Design Studio. To add steps to the list, click + and select steps. To remove steps from the list, select one or more steps and click −. To reorder steps, select a step and move it up and down the list using arrows.

Desktop Automation

Use this tab to set preferences for Desktop Automation.

The following options are available.

Option	Description
Command Time-Out (sec.)	Specifies how long the Design Studio must wait for a reply from a command on an automation device. This option applies only to automating terminals and browsing websites in Desktop Automation robots. A command is an instruction sent to Automation Desktop, such as click a mouse button, open an application, add a location found guard, and so forth. If a command cannot be completed in a specified time, the service sends a notification and execution of the robot stops. Note that in case of a Guarded Choice step, this setting applies to invoking the guard in the workflow, but waiting for the guard to be satisfied is not bound to this timeout and can wait forever. Similar situation occurs when using the Move Mouse and Extract steps. The commands must be invoked on the device within the timeout specified in this field, but the robot waits for up to 240 seconds for the commands to complete.
Local Hub TLS Configuration Settings See Use TLS Communication for more information.	

Option	Description
Use Default	Use files provided by Kofax RPA for TLS communication between Design Studio and automation devices.
Private Key File	Path to a private key file used by the local hub that exists on the Design Studio computer.
Public Key File	Path to a public key file signed by the underlying private key.
Trusted Certificates Folder	Folder to store trusted certificates.

Local Databases

Use the "Local Databases" tab to create a database in Design Studio. Note that databases created in Design Studio are only available in Design Studio.

To make the database available both in Design Studio and on the server, the databases must be configured in the Management Console.

A list of created connections is shown in the left pane. You can create new connections, remove connections, or change their order using the buttons below the list. The currently selected connection is configured in the right side of the "Local Databases" window. The field name, host, type, schema, username and password are mandatory and must be specified.

The different database types are defined in the Management Console and automatically distributed to Design Studio at startup. New database types must be created in the Management Console.

Option	Required	Description
Name	Yes	A name for uniquely identifying the database in Kofax RPA. The name is used for internally referencing the database and it may only contain alphanumeric characters and underscores.
Host	Yes	The host name of the database server. This can be an IP address, or the fully qualified domain name, such as myhost.kofax.com.
Type	Yes	The type of database, such as Oracle. The different types of databases are configured in Management Console and provided automatically at Design Studio startup.
Schema	Yes	The name of the database schema (or catalog).
Username	Yes	The user name for the database.
Password	Yes '	The password for the database.
Max active connections	Yes	The maximum number of concurrent connections to the database that Kofax RPA (RoboServer or Design Studio) create. The connections are managed by a connection pool, which means that existing connection is reused before creating new connections.
Max idle connections	Yes	The maximum number of idle connections allowed. If more connections are created due to a heavy load, they are closed automatically when no longer needed.

To test the current connection, click **Test Connection**.

> **Note** This action only tests the connection to the database. It does not test whether you have the proper permissions in the database.

Connecting to Oracle: If you are using an Oracle database, In the Username field, you must enter a username and a role. For example, if the username is "sys" and the role is "sysdba," you should enter "sys as sysdba" in the Username field.

Proxy Servers

Use the Proxy Servers tab to specify the number of proxy servers that can be used by Design Studio.

Option	Description
Use Proxy Server	When selected, the use of a Proxy Server is enabled.
Host	The host name of the proxy server, which can be an IP address, or the fully qualified domain name, such as myproxy.kofax.com.
Port number	The port number on the proxy server. Leave this blank to use the default proxy server port 8080.
Username	The user name to use if the proxy server requires a login.
Password	The password to use if the proxy server requires a password.
Excluded hosts	Here you can specify a list of host names that the proxy server should not be used for. You can specify either one host name per line or a host name pattern per line using wildcards (*). Each host name can be an IP address or a fully qualified domain name, such as www.kofax.com.

Use the Import button under the proxy server list to import a list of proxy servers. The file may hold an arbitrary number of proxy server definitions, each of which must comply with the following format:

```
proxyName.proxyServerUse = true
proxyName.proxyServerHost = host name or IP address
proxyName.proxyServerPort = port number
proxyName.proxyServerUserName = user name
proxyName.proxyServerPassword = password
proxyName.proxyServerExcludedHostNames = list of hosts
```

Where proxyName is a name to identify a particular proxy server. Each proxy server must have its own unique proxyName.

When multiple proxy servers are specified, a new proxy server is selected every time a robot is run.

You can also specify a proxy server for an individual robot. This is done when you configure the robot in the Robot Configuration window in Design Studio. Such a proxy server overrides the proxy servers specified here. See Configuring Robots for more information. Furthermore, the proxy server is changed during robot execution with the Change Proxy action.

See Use Proxy Services for more details.

Certificates

Use the Certificates tab to specify whether a robot should verify the identity of a web server that it accesses via HTTPS. Such a verification is routinely (and invisibly) done by ordinary browsers to detect phishing attacks. However, the verification is often not necessary when robots collect information, because the robots only access the web sites that they have specifically been written for. Thus the verification it is not enabled by default.

Verification is done in the same way a browser performs verification.

The web server's certificate is checked based on an installed set of trusted HTTPS certificates similar to those you can configure in a browser. See the *Kofax RPA Developer's Guide* for more information about HTTP certificates.

The following table describes the options on the Certificates tab.

Option	Description
Verify HTTPS Certificates	When selected, a robot verifies a web site's certificate when accessing it over HTTPS. Verification is done based two sets of trusted certificates: the set of root certificates and an additional set of server certificates.
HTTPS Client Certificates	A list of client certificates that the robots can use. Use the buttons under the list to add or remove certificates.

Note that root certificates are installed with Design Studio just as root certificates are installed with your browser. They are found in the Certificates/Root folder in the application data folder. See the "Certificates" section of the *Kofax RPA Administrator's Guide* for more information.

Some HTTPS sites may use certificate authorities that are not included by default. In this case, you need to install the appropriate certificates for Design Studio to load from these sites. Most often, these would be installed in the Certificates/Server folder in the application data folder.

For the purpose of handling HTTPS sites, it does not matter whether you add certificates to the set of root certificates or the set of server certificates.

To install a certificate, you need to obtain the certificate as a PKCS#7 certificate chain, a Netscape certificate chain, or a DER-encoded certificate. You install the certificate by copying it to one of the folders mentioned above. The name of the file containing the certificate does not matter.

Bug Reporting

Use the Bug Reporting tab to set preferences for emailing bug reports related to internal Design Studio errors. When an internal error occurs in Design Studio, the user can send a bug report with details about the error. The user can also send a bug report manually, by selecting the Report Bug action from the Help menu.

Users are generally encouraged not to change the default settings on this tab unless absolutely necessary.

Option	Description
Mail server	Lists the mail server to use when sending bug reports.
Send email to	Lists the email address to which bug reports should be sent.

Management Consoles

Use the Management Consoles tab to configure connection settings to Management Consoles. The URL must be unique, but you can configure several connections to the same Management Console with different protocols, user names and passwords. The first time you start Design Studio and specify a license server, that server is automatically added to the list of Management Consoles.

Name

Name of the Management Console.

URL

A URL to connect to the Management Console. Specify the protocol by typing either `HTTP` or `HTTPS` and a port number. For example, `http://localhost:50080/`. You can also use an IP address in this field.

User Name

Specify a user name to access the Management Console. The following built-in users can connect to the Management Console.

- admin
- Administrator
- Project Administrator
- Developer

Remember Password
If selected, Design Studio stores the password entered for a Management Console. If this option is cleared, you need to enter the required password every time you connect to a Management Console from Design Studio. You are prompted to enter the password to the Management Console when the Design Studio is being started.

Password

Specify the user's password.

Use Proxy Server
Select to use a proxy server through which Management Console connects to external servers.

- Host Name: The proxy server host name
- Port: The proxy server port
- User Name: User name to connect to the proxy server if it requires authentication
- Proxy Password: Password to connect to the proxy server if it requires authentication

> **Note** See the Proxy Servers section for more information.

Show DB warnings

When selected, database warnings, such as missing tables, are shown at the top of the robot editor.

Accept JDBC drivers

When selected, JDBC drivers are distributed from Management Console to Design Studio. Users should rarely need to disable this option.

Use as Primary
Select to use the current Management Console as the primary Management Console. This option affects which Management Console to connect to when accessing the password store, Robot File System, and Kofax TotalAgility configuration. Robots in projects that only exist locally and are not synchronized with a Management Console use the Management Console marked as primary. If a robot is in a shared project, it connects to the Management Console that the project is synchronized with.

Upload to Management Console

To publish your robot as well as the types and snippets it uses to a Management Console, right-click the robot in the Projects view and click **Upload**. This dialog box appears when you upload your files from a local non-shared project to a shared project on a Management Console. See Work with Shared Projects for details.

Management Console

Select one of the Management Consoles from the list. The list contains Management Consoles that your Design Studio is connected to.

Project

Select the project to upload to.

Remember this (as shared project)

Select this option to link your project to the selected Management Console project.

Chapter 4

Web Automation

This chapter describes the different tasks involved in creating Web Automation robots and the types used by these robots. Step actions help you login to applications, extract data from web pages, enter data into forms or search boxes, make menu selections, and scroll through multiple pages. Your robot can also access databases, files, APIs, web services, and other robots, exporting data from one application and loading it into another; transforming data as necessary along the way.

Get Started

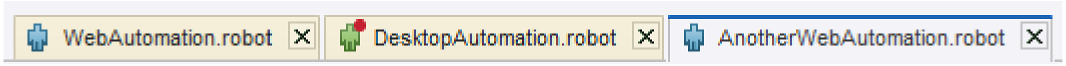
The following procedure shows how to create a Web Automation robot.

1. Open Design Studio.
2. Click **File** > **New Web Automation Robot**.
3. Specify a name for the robot, select a project to store the robot in, and then click **Next**.
4. Optionally, enter a URL that the robot should start from. When you do so, a Load Page step with this URL is automatically inserted in the robot.
5. Select a browser engine to use: Classic or Default (WebKit-based).
6. Select a design-time execution mode to use: Minimal Execution (Direct) and Smart Re-execution (Full). Smart Re-execution is the default mode.
7. Click **Finish**.

To start editing and executing your robot, you need to prepare it for execution by clicking **Prepare Execution** ⚡ in the Applications view or on the toolbar. By clicking this action, you put the robot into execution mode, which enables you to execute it while editing. You can execute action steps right after you insert them in the robot workflow and immediately see the result. When a Web Automation robot is not prepared for execution, you can still perform some basic editing, such as add action steps and change their properties, configure variables, but you will not be able to see the result of step execution and to obtain and use data from websites that you interact with to design the robot.

The execution privilege is intended only for Web Automation robots. Only one Web Automation robot at a time can have the execution privilege, so to take the execution privilege from one robot to another, open the tab with the required robot and click **Prepare Execution**. Also, a robot cannot have the privilege to execute in both Design and Debug mode at the same time.

When a Web Automation robot has the execution privilege, the editor tab of this robot is highlighted. When a Web Automation robot is calling a Desktop Automation robot, the tabs of both robots are highlighted for convenience as shown below. The robot where execution is currently located is marked with a red dot.



Also, you can open the Desktop Automation robot that the Web Automation robot is calling by clicking **Open** in the properties of the respective Call Desktop Automation Robot step. The robot is opened on a new tab.

Tip Visit the Kofax Intelligent Automation SmartHub at https://smarthub.kofax.com/ to explore additional solutions, robots, connectors, and more.

Choose your browser

Kofax RPA uses a browser to interact with web sites or web applications. Web Automation part of Kofax RPA currently comes with two different browsers, each optimized for different purposes: the Classic engine and the Default (WebKit) engine. Desktop Automation part of Kofax RPA contains a built-in browser based on the Chromium engine.

Classic engine browser
This browser is intended for backward compatibility with legacy Kofax Kapow products.

WebKit-based browser
This browser saves the state of the application (web page) internally in the robot and it is intended for legacy low resource-dependent web sites that do not keep server-side state and do not implement the most recent web standards and technologies.

Chromium-based browser (CEF)
This is the most current browser engine implemented in Kofax RPA. This browser must be used with modern web sites where the state resides on the site, that is the state is external to the robot. Execution of steps in this browser move forward only and you must use the browser tools (such as back and forward buttons) to navigate the web pages. This browser is updated regularly and includes the latest web technologies. For more information on how to use this browser, see Access Websites.

Make Robust Web Automation Robots

Web sites often change without notice. Robustness is the term used to describe how well robots cope with web site changes. The more changes the robot can deal with and still work correctly, the more robust it is.

Making robust robots involves analyzing the web site in question and understanding how it responds in various situations, such as when a registration form is filled out incorrectly. In a sense, writing robust robots involves a kind of reverse engineering of the web site logic, and usually the only way to do this is through exploration.

The two different approaches to robustness each serves a different purpose:

- Succeed as much as possible.
- Fail if not perfect.

Succeeding as much as possible might, for a robot extracting news type variables, mean that it should extract as many news items as possible. In Web Automation robots, you use conditional actions, Try steps, and data converters to deal with different layouts, missing information, and unusually formatted content.

Failing when things are not perfect might, for an order submission robot, mean that it should fail immediately if it cannot figure out how to enter a field correctly, or the order result page does not match an exact layout. In this sense, failing does not mean to generate an API exception. Instead, it means that the robot should return a value dedicated to describing errors and failure causes. Robots taking input variables often fail, rather than succeed as much as possible. In Web Automation robots, you can use dedicated error type variables, error handling, and conditional actions to detect and handle unexpected situations.

For more information on Design Studio techniques to make robots more robust, consult the following sections:

- Write Well-Structured Robots
- Handle Errors
- Debug Robots
- Web Automation Steps
- Data Converters

Write Well-Structured Robots

Writing well-structured Web Automation robots is essential because each robot is a program. Writing well-structured robots is important because:

- It helps document the robots.
- It makes it easier to maintain the robots.
- It makes it easier to find your way around the robots.

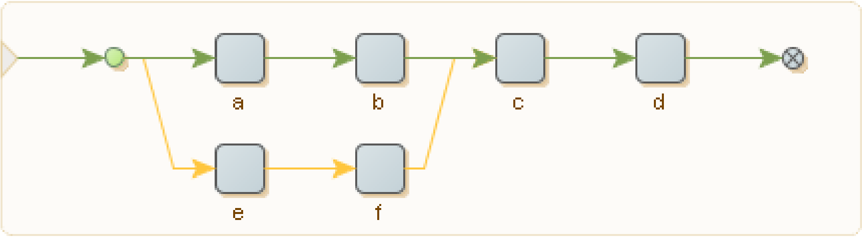
A side-effect of writing well-structured robots is that it can also make them load faster in Design Studio. As a result, robots are generally more responsive when they are edited in the Robot View.

The two main tools for writing well-structured Web Automation robots are Snippet steps and Group steps. Both step types are a way to take a part of a robot, give it a descriptive name, and pack it up in a single step. This way you can forget what the part of the robot does in detail and concentrate on the overall structure of the robot. This concept is similar to those in other programming languages, such as methods, functions, and procedures.

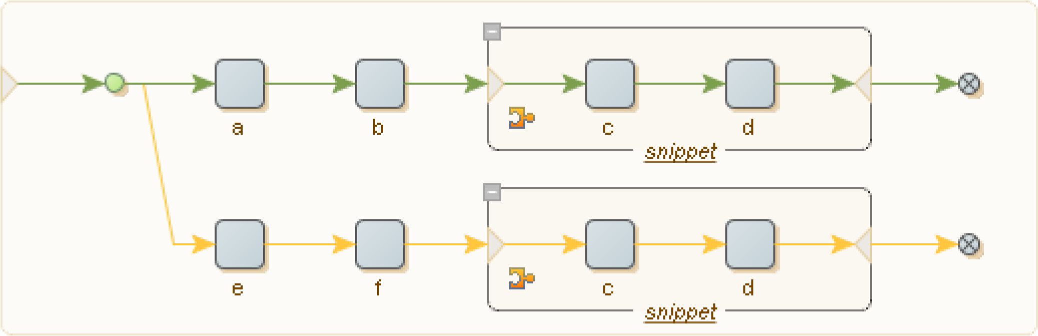
You use a group step to pack up and hide steps that perform a well-defined task. Give the step a descriptive name, such as Login to site X, Report error. It is important to give a relatively short descriptive name to the group step that describes what the steps inside the group do. If you cannot provide a good name, then it may be because the group does not perform a well-defined task. By introducing a group step you help document your robot, because the name describes what this part of the robot does.

Although snippets are mainly introduced to share functionality between robots, they can also be used inside a single robot to help structure it. If you have a collection of steps in a robot used in several branches, such as connections from different parts of the robot joining at the start of the steps, you can replace such steps sharing by introducing a snippet containing the steps.

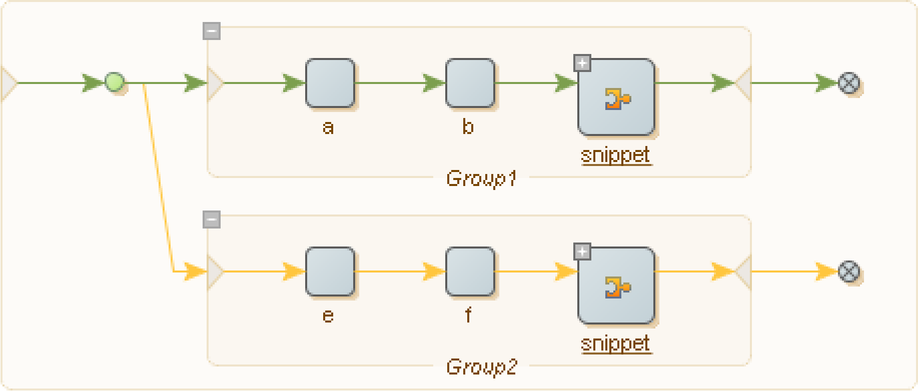
The following robot structure uses snippets and groups instead of joining connections.



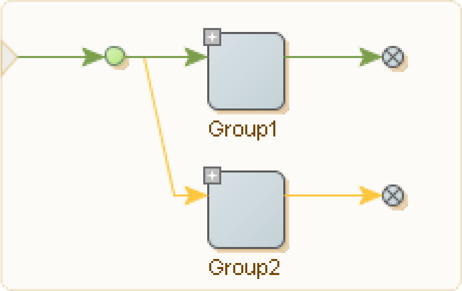
The last two steps **c** and **d** are shared by the two branches starting with the steps **a** and **e**. In real life, you probably have a much larger robot and more than two branches sharing steps in this way, and the steps involved may be far apart. As a result, it may be difficult to get an overview of the robot. As a first step towards getting a better structured robot, you can introduce a snippet step containing the steps **c** and **d** as follows.



You can edit the steps inside the snippet steps and still be sure that the changes are shared in the two branches. You can structure the robot further by putting both branches into a group step:



Finally, you can use the two group steps and get the following simple robot.



This resulting robot does two tasks, one performed by Group1 and the other performed by Group2. By giving these two groups descriptive names, the robot has a more logical structure than the original robot.

Admittedly, this is a very simple example, but when robots get beyond a certain size and contain connections crisscrossing the Robot View, they can become overly complex. Restructuring the robot in the manner described above may help ensure that the robot overview is manageable.

Handle Errors

A step in a robot may generate an error when it is executed. For example, this happens if the tag finders cannot find the tag to work on, or if the step action generates an error. You can configure test steps to act as if an error occurred if the test fails. The default behavior of a robot is to report and log the error immediately, and to omit execution of the steps beyond the one that failed. However, by configuring the error handling properties of the steps in the robot, you can change this behavior. For example, you can make the robot skip a step that generates an error, or you can make it try alternative branches.

Note The error handling behavior described in this help system applies to runtime execution of a robot (such as execution in RoboServer or in debug mode), not to the execution in design mode in Design Studio. In design mode, an error is normally reported immediately, and the execution of the subsequent steps is aborted. One exception is when the step is configured to "Ignore and Continue" in case of errors, in which case Design Studio does ignore the error and executes the next step, just as it would during runtime execution.

The following shows how to handle API Exceptions and Logging errors:

1. In the Step View, Error Handling tab, select an error handling option.

   a. Select **API Exception** to report the error back to the caller of the robot. This is most useful when the robot is executed by a client via one of the APIs and runs in RoboServer. In this case the error is sent back to the caller via the API as a RobotErrorResponse, which causes an exception at the caller side, at least when using the default RQLHandler. See Error Handling for more details when the robot is executed in other ways.

   b. Select **Log as Error** to log the error. Logging happens in different ways depending on whether the robot is run in the Design Studio or in RoboServer.

   Note You can select or clear the check boxes, which may be marked with an asterisk * to indicate they are set to a non-default value. For details, see Showing Changes from Default, which explains how to remove the asterisk and revert to the default value. When the default value applies (that is, when no asterisk is present), be aware that the default varies according to how the error is handled.

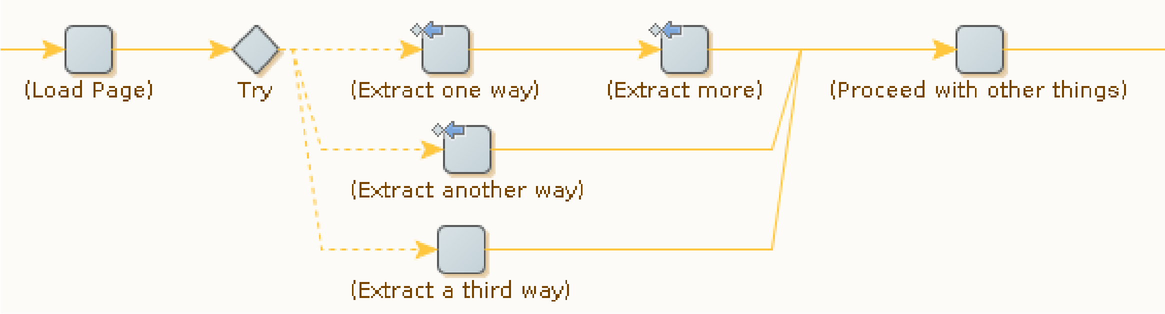
2. In the **Then** field, select an option from the list.

   This value defines how and where robot execution continues after an error is encountered. The possible options are described with examples in the following sections. For detailed descriptions, see the Error Handling section.

Error Handling Alternatives

You can select several alternative methods to handle errors. See Conditions and Error Handling for an error handling overview.

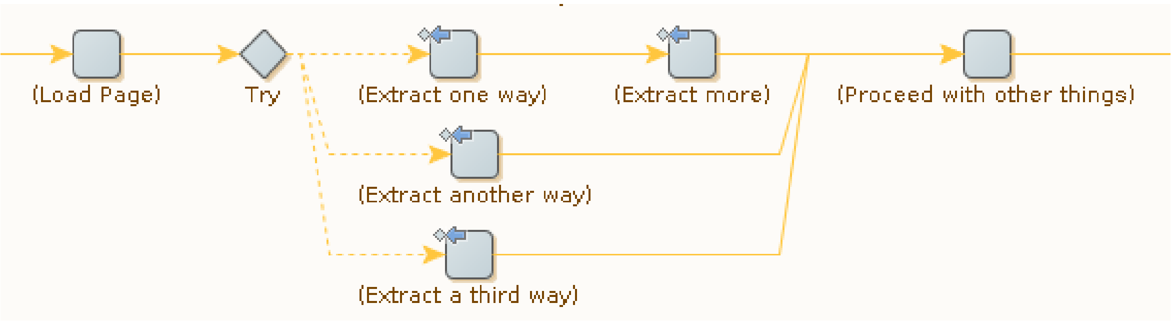
Suppose that some part of a web page has varying structure and layout, but always comes down to one of three cases. In each case there is information to extract. It can be done by attempting the extraction one case at a time. If it fails, the next case is tried, until the third one, which we assume will succeed.



Note the ◇⬅ (Try Next Alternative) icons on the Extract steps. If extraction fails, the next branch from the Try step is executed (if a branch coming out of a Try step succeeds, the next branch is not executed). The Extract steps do two things at the same time: They do extraction from the web page, and if it cannot be done, they ensure that the next approach is tried. Note that if either of the two steps on the first branch fails, the second branch is executed; this is an example of how the "success condition" for a branch can be expressed by a combination of steps.

This approach works best if the "third way" of extracting is bound to work (for example, by applying a fixed set of default values rather than actually extracting data from the web page). If the third branch accesses the web page as the first two branches do, it may not be wise to assume that it will succeed. The next time the robot is run, the web site may have changed so much that none of the three strategies succeeds, and the robot should be able to respond in a reasonable manner.

The easiest way to respond is to report the problem back to the caller and log it, giving up on doing the extraction and whatever would follow. This can be achieved by making the third branch inform the Try step if it fails to do its work, similar to the first two branches.



(For a Try step, **Skip Following Steps** means that no additional action is taken beyond reporting and logging.)

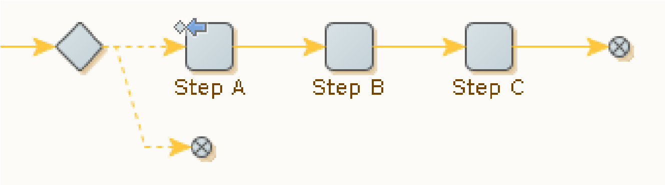
Alternatively, it is possible to make the Try step propagate the problem back to an earlier Try step for handling. For more information, see Try-Catch.

Shortcuts for Common Cases

Try steps and Try Next Alternative error handling are very flexible tools. Used in the proper way, they support many different ways to handle errors, and in this topic, we will show a couple of simple and common cases. In fact, these cases are so common that they are also supported by specialized error handling options.

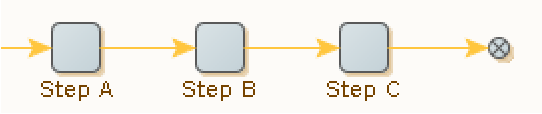
Skip Following Steps

In many cases, a robot must be able to handle optional elements on a web page. That is, if the elements are present, they must be handled (for example, data must be extracted), but if they are absent, the handling of elements can be skipped. Their absence is not an error, but an expected situation. This can be expressed as follows in a robot. Step A tests if the elements are present (by trying to extract something), while steps B and C do further processing that depends on the success of step A.

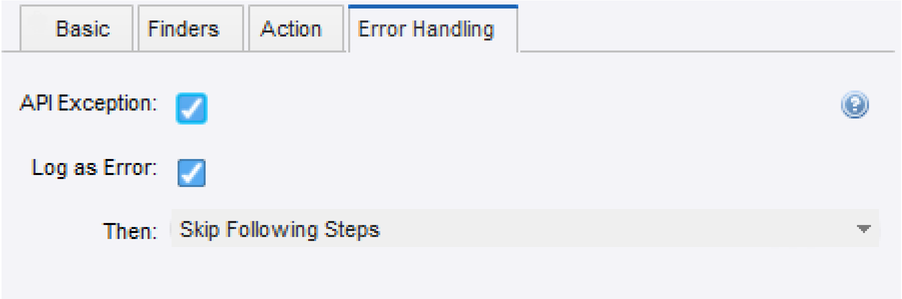


If step A is not successful (because elements are missing on the web page), its Try Next Alternative (◇⬅) error handling sends notice to the Try step (which is unnamed in this example). This causes the second and empty branch to be executed, after which execution of the whole branch that starts in the Try step is done. Thus steps B and C are not executed if step A is not successful.

This situation happens so often that a specific error handling option, Skip Following Steps, is introduced as a shortcut. It makes it possible to simplify the example as follows.



The error handling for step A is configured as follows. This is the default configuration for all new steps.



API Exception: ☑

Log as Error: ☑

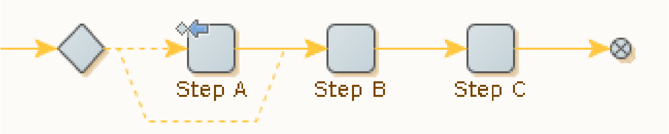
Then: Skip Following Steps

Strictly speaking, you need to clear the **API Exception** and **Log as Error** check boxes to get exactly the same behavior as shown with the Try step. This is because the default values for these check boxes are different for the two ways to do error handling.

Note that if step B had been similar to step A (that is, if step B had also had Try Next Alternative error handling), this same shortcut could be used.

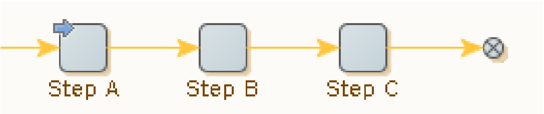
Ignore and Continue

Sometimes, some action (such as extraction) needs to be done if some condition is met, and otherwise it can be skipped. Subsequent steps do not depend on the result (or a proper default for the result has been set up in advance). This case can be expressed as follows.

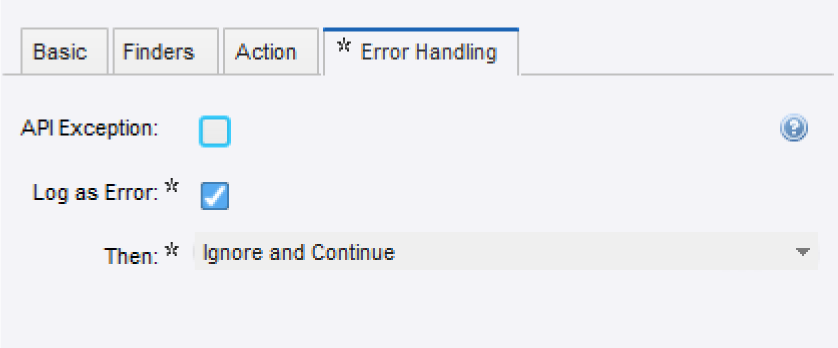


If step A is not successful, its Try Next Alternative (◈⬅) error handling causes the second and empty branch from the (unnamed) Try step to be executed. After this, execution continues at step B with the same robot state that was input to step A, thus step A is effectively skipped.

This can also be done without the Try step by using the error handling option **Ignore and Continue** (➡) on step A.



One interesting possibility is to have the situation logged even though it is otherwise ignored. This can be achieved by configuring error handling on step A as follows.



API Exception: □

Log as Error: * ☑

Then: * Ignore and Continue

The same can be done if you prefer to use the method with a Try step.

At Target

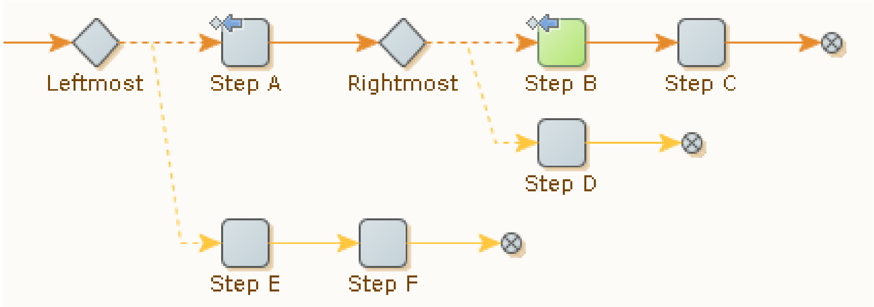
Try Next Alternative

By its very nature, Try Next Alternative error handling refers to a Try step. This Try step must be a prior step on the current execution path, that is, it must be one of the steps whose execution led up to the current step. Otherwise the "next alternative" part of Try Next Alternative does not make much sense.

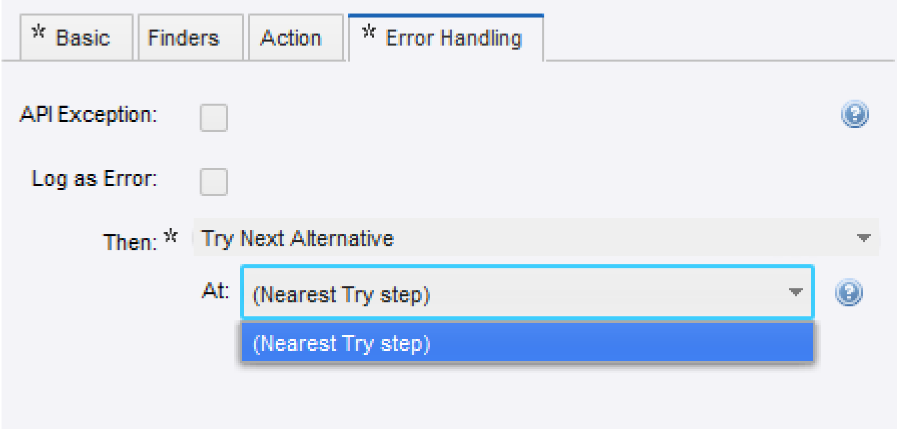
Consider a robot with several Try steps on the current execution path. In this situation, the robot will be in the process of executing a "current branch" relative to each of these Try steps, and for each of them the "next branch" is different. For example, in the following robot, if an error occurs in step B, Try Next Alternative continues execution using one of the following approaches.

- The next branch at the rightmost Try step, so that execution skips step C and continues at step D

- The next branch at the leftmost Try step, causing execution to skip both steps C and D and continues at step E



You can define the option on the Error Handling tab. The following example shows the error handling configuration for step B.



API Exception: □

Log as Error: □

Then: Try Next Alternative

At: (Nearest Try step)

(Nearest Try step)

The default is the Nearest Try step, but you can select other Try steps on the execution path. The reference to the Try step is by name; if several Try steps have the same name, the nearest (rightmost) one with that name is implied. This can be used to advantage as described in Try-Catch.

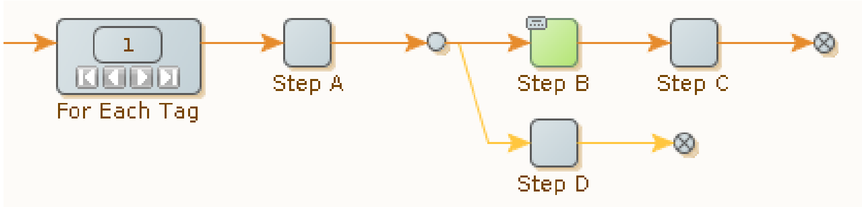
While this example only describes At targets in relation to Try Next Alternative error handling, you can use such references with the Next Iteration and Break Loop error handling options described in Looping. In those cases, the references go to loop steps rather than Try steps.

Looping

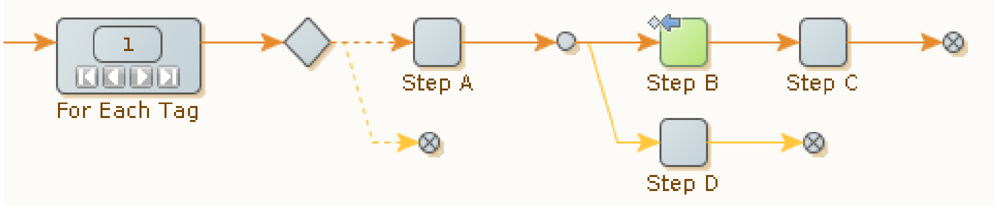
Sometimes when an error occurs or a test fails, the proper reaction is to abandon execution of the current iteration of a loop, or the whole loop. This is supported by two specialized error handling options.

Next Iteration

In the following robot, step B has error handling for Next Iteration. If an error occurs during execution of this step, execution of the current loop iteration is stopped. Steps C and D are not executed; instead execution continues at step A with a robot state that reflects the next tag among those that the loop step iterates over.



This error handling option is a shortcut, as you can achieve the same effect with the aid of **Try Next Alternative** and a Try step.



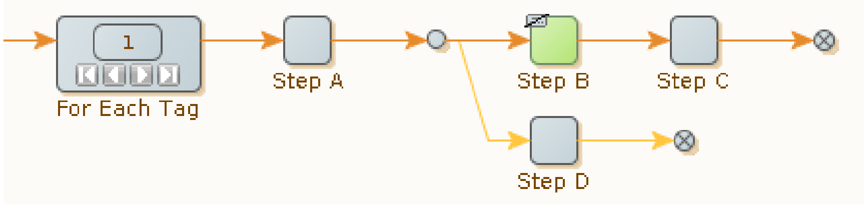
Note that this transformation in general requires use of At Targets because other Try steps may interfere.

If the robot contains several loops steps after each other, it is possible to select the one in which to go to the next iteration.

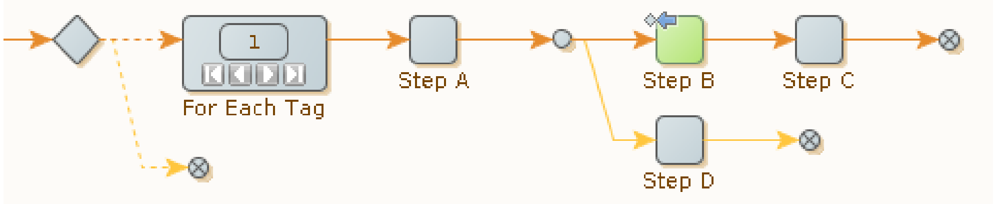
Next Iteration does not work with Repeat-Next loops. The word "next" has very different implications for the browser state in these two cases.

Break Loop

Instead of completing a single iteration of the loop with Next Iteration, you can use Break Loop to abort the whole loop.



This error handling option is a shortcut. The following robot will have the same effect:



Note that unlike Next Iteration, Break Loop does not work with Repeat-Next loops.

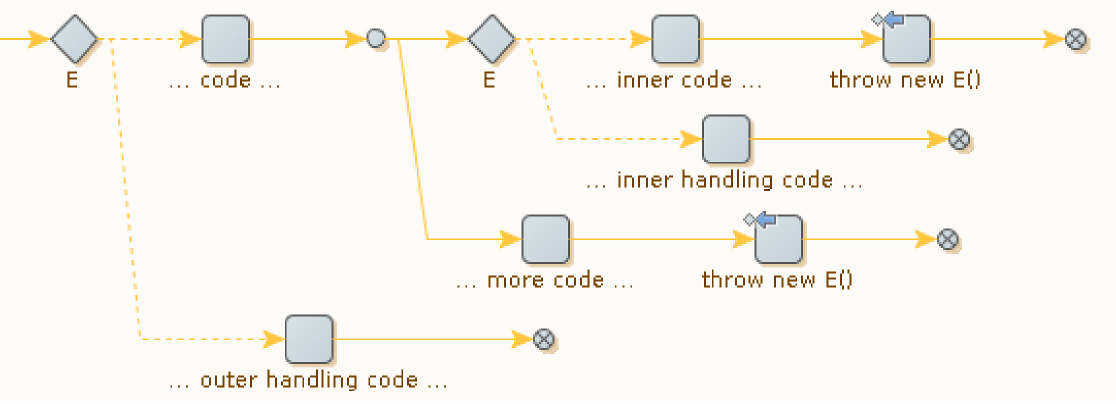
Try Catch

When Try Next Alternative error handling is used with an explicit **At** reference to a target Try step, the step is identified by its name. Most often, the fine distinction between the target step and its name is not important, but it can be exploited to provide exception handling functionality similar to the try-catch constructs in Java or C#.

In those programming languages, a section of code between "try" and "catch" has special error handling. If a specific error is signaled within this section (by "throwing" a named "exception"), the piece of code following the similarly named "catch" is executed. Try-catch constructs can be nested, and a named "exception" is always handled by the innermost enclosing "catch" with a matching name. For example:

```
try {
   ... code ...
   try {
     ... inner code ...
     throw new E(); // caught by innermost "catch"
   }
   catch (E e) {
     ... inner handling code ...
   }
   ... more code ...
   throw new E(); // caught by outermost "catch"
 }
 catch (E e) {
   ... outer handling code ...
 }
```

In robots, something similar can be done with Try steps. Remember that an "At" reference to a Try step with a given name always means the nearest prior step with that name (along the current execution path). It is permitted to use the same name for several Try steps, even on the same execution path. Thus each try-catch construct is modeled with a Try step having the same name as the "exception." The Try step has two branches: one for the code part of the "try" construct, and one for the code part of the "catch" construct.



The correspondence between the Java/C# syntax and the Design Studio terms is described in the table.

Java / C# Syntax	What to use in Design Studio
try { ...code... }	The first branch of a Try step (the steps correspond to code)

Java / C# Syntax	What to use in Design Studio
Name of an exception	Name of a Try step
throw new E() within the code of a try	Handling an error with "Try Next Alternative at E"
catch E { ...code... }	The second branch of a Try step named "E" (the steps correspond to code)

Thus, the core idea is: When Try steps are used for error handling, name the Try steps after the error situations they handle. The advantages are:

- The naming helps make the purpose of each Try step clear.
- When errors are handled on a general level (with a Try step more to the left in the robot), it is still easy to do specialized handling in some cases (with the aid of a second Try step with the same name).

Identify Error Handling in Robot View

Robot steps containing special error handling are marked with a small symbol in Robot View. The symbol is based on the type of error handling defined for the step to help users visually identify steps containing custom error handling. If you do not want to have steps with custom error handling marked, you can disable this feature on the **Design Studio Settings** > **Robot Editor** menu in Design Studio.

See Robot Editor for more information.

Error Handling

When an error is encountered during execution of a step, it is handled as specified on the Error Handling tab for the step's configuration. Test actions may also apply the same handling when a condition fails. There are two aspects of handling an error or failed test: How and where to report or log the incident, and how and where to proceed with execution of the Web Automation robots.

Properties

Error Handling is configured using the following properties.

API Exception

This property specifies whether to report the incident back to the caller of the robot. Reporting works differently depending on how the robot is executed:

- The property has no effect when the robot is executed in Design Mode in Design Studio because the incident always is reported in this mode (except when it is handled using Ignore and Continue as described below).
- When the robot is run in the Management Console, API exceptions are just counted and logged as errors and do not stop the execution. In this scenario, we recommend using the same value (checked or unchecked) as for Log as Error.
- When the robot is executed by a client via one of the APIs and runs on a RoboServer, the report is sent back to the caller via the API as follows:

  ```
  RobotErrorResponse
  ```

  This causes an exception at the caller side, at least when using the default RQLHandler. (This case explains the name of the property.)

Note The property may be unchecked or checked, but may also be marked with an asterisk * meaning that it has been explicitly set to a non-default value. This is explained in Show Changes from Default, where it is also shown how to remove the asterisk and revert to the default value. When the default value applies (that is, when no asterisk is present), be aware that the default varies according to the selection made under the Then property.

Log as Error

This property specifies whether to log the incident. This differs from reporting (see the preceding).

- When the robot is executed in Design Mode in Design Studio, the incident is added to the log that is displayed when you select View Log in the View menu.
- When the robot is executed in Debug Mode, the incident is added to the Log tab.
- When the robot is run in a RoboServer (from the Management Console or via the API), the incident is logged in the place defined in the cluster settings for the given RoboServer. This is most likely the same database that the Management Console uses.

Note The property may be unchecked or checked, but may also be marked with an asterisk * exactly as explained earlier for API Exception.

Then

This property specifies how and where to continue execution of the robot after the incident. The following options are available.

Skip Following Steps (Default)

The steps following the one with the error (or failed test condition) are not executed. Execution otherwise proceeds as if they had been executed successfully.

Ignore and Continue

The step with the error (or failed test condition) is skipped, and execution continues with the following step as usual.

Try Next Alternative

This can be used in a branch coming out of a Try step. Execution of the current branch from the Try step is abandoned and execution continues with the first step on the next branch from that Try step. Execution will continue there with the same robot state as for the first branch.

If the current branch is the last one from that Try step, Try Next Alternative is not illegal, but results in an error ("all alternatives failed") which is handled according to the way error handling is configured for the Try step.

When Try steps are nested, it is possible to select the relevant Try step among those that are on the execution path to the current step.

Send Backwards (legacy)

This option is present only to provide backwards compatibility with robots created by Design Studio versions prior to 8.0. It is described separately.

Next Iteration

This can be used within a loop, that is, following a loop step of some kind (except a Repeat-Next loop). Execution of the current loop iteration is abandoned and execution continues with the next iteration. That is, execution continues with the first step after the loop step, and with a robot state that corresponds to the next looped-over item.

When loops are nested, it is possible to select the relevant loop in which to go to the next iteration.

Break Loop

This can be used within a loop, that is, following a loop step of some kind (including a Repeat-Next loop). Execution of the loop is abandoned. Thus the current iteration is not finished, and subsequent iterations are not executed at all. This means that execution continues with the same step as after normal termination of the loop.

When loops are nested, it is possible to select the relevant loop among those that are on the execution path to the current step.

Step References in "At..."

Under error handling you can select various ways to proceed in case an error occurs. The following three ways listed require that you identify a previous step on the execution path (either a Try step or a loop step) where execution should proceed. Execution proceeds at or after the selected step depending on which one you select:

- Try Next Alternative
- Next Iteration
- Break Loop

When identifying the preferred step, you can select the "nearest step" (of the appropriate kind) or a specific step by name. Sometimes, however, the preferred step does not appear in the selection list for a number of reasons:

- Only named steps can be selected. If the step is unnamed (the default for Try steps), the step must be given a name before it will appear in the selection list. If the step is the nearest one of the appropriate kind, you can also reach it by selecting "nearest step."
- Remember that Next Iteration cannot be used with Repeat steps. Thus Repeat steps do not appear in the selection list for Next Iteration.
- A step name never appears more than once in the selection list, because any reference always goes to the nearest step (of the appropriate kind) with the selected name. Keep that in mind when modifying the robot!
- If more than one path through the robot leads to the current step, an (appropriate) step with the selected name must be present on every path to the current step. Otherwise, the name does not appear on the selection list.

The last two rules support using error handling in an advanced way that is very similar to the try-catch constructs in Java or C#. See the Try-Catch section for more on this. In those programming languages, you "throw" a named "exception" at the point of error, and "catch" it (by name) somewhere in surrounding code. In robots, you can do something similar with Try steps. The correspondence between the Java/C# terms and the Design Studio terms is described in the table.

Java/C# term	What to use in Design Studio
try	The first branch of a Try step
Name of an exception	Name of a Try step
throw E	Handling an error with "Try Next Alternative at E"
catch E	The second branch of a Try step named "E"

This way of thinking about error handling probably is most useful in large robots. When using error handling this way, the names of the Try steps signal what kind of exceptional circumstances each Try step's second branch handles.

Send Backwards (legacy)

Send Backwards (legacy) is one of the options for error handling. It is present only to provide backwards compatibility with robots created by Design Studio versions prior to 8.0 and should not be used in new robots.

Send Backwards is similar to another error handling option, Try Next Alternative, because it affects execution of the branches that go out from a Try step. However, it does so in a quite different way. Its immediate effect is that the steps following the current one are not executed (similar to Skip Following Steps). Aside from that, execution of the current branch from the Try step continues. However, the error is also "sent backwards" to the Try step and remembered there, which causes execution to continue with the next branch of the Try step once execution of the current branch has finished successfully. This part is similar to the effect of Try Next Alternative (although delayed), and the same comments regarding the robot state apply.

If the current branch is the last one from that Try step, Send Backwards is not illegal, but results in an error ("all alternatives failed") which is handled according to the way error handling is configured for the Try step.

If there is no prior Try step to be found, the error is "sent backwards" to the beginning of the robot and is reported via the API and logged just before execution of the robot finishes.

Send Backwards is quite difficult to work with because of the delayed effect, and we recommend Try Next Alternative instead. When robots created using Design Studio versions prior to 8.0 are opened, they are automatically modified to use Try Next Alternative instead of Send Backwards (whenever this can be done without changing the results from the robot).

Debug Robots

This section explains how to debug a robot using the debug mode built into Design Studio. To confirm that the robot does what you expect, use debug mode to execute a robot the same way it will be executed by RoboServer.

> **Important** It is not possible to execute robots in Design and Debug modes at the same time.

Basic Debugging

1. To switch to debug mode, in Design Studio, click **Debug** > **Switch to Debug Mode** on the toolbar or click the **Debug** button.
2. To start debugging the robot, click **Run**.
3. In the Robot view, you can watch the robot execute in debug mode.

   You can also watch the results in the main panel.

   In the Input/Output tab:

   - The Input panel shows the input variables.

     > **Note** If the robot has no input variables, the Input panel is not shown.

   - The Output panel shows all values returned so far during the execution.
   - The API Exceptions tab shows all API exceptions generated during the execution.
   - The Log tab shows what has been written to the log during the execution.
   - The State tab shows the robot state, if any.
   - The Summary panel (to the right of the main panel), shows a summary of the execution. This summary contains the number of returned values, the number of API exceptions generated, statistics on the number of HTTP requests, the amount of sent and received data, and the number of JavaScript instructions executed.

   > **Note** It is important to understand that execution in debug mode is independent of the execution done in design mode in Design Studio. Therefore, debug mode has its own current step and its own current robot state, independent of the current step and current robot state in design mode. In debug mode, the current step is the step that is about to be executed, or is being executed in the debugging process, and the current robot state is the input to that step.

4. Click **Stop** to stop debugging.

   You can stop the debugging at any time.

5. To stop debugging when a certain event occurs, enter a **Stop When** action.

   Here, you can select whether the debugging should stop when values are returned, when API exceptions are reported, and when breakpoints are reached.

   Of course, debugging always stops when the execution of the robot has completed.

   When debugging has stopped, you can see the reason for the stop in the status bar at the bottom of the Robot Editor.

   If the debugging has stopped before the execution of the robot is complete, you can watch the current robot state in the State tab. The Variables, Windows, Cookies, and Authentications sub-tabs show the robot state in the same way as in the State View in Design Studio. The API Exception sub-tab shows the API exception, if the execution stopped because an API exception was reported.

6. If debugging has stopped before the execution of the robot is complete, click **Run** ▶ to resume debugging.

   You can also click **Restart Debug** ■ to restart debugging. This cancels the current debugging process and makes the debugger ready to start a new debug operation from the start of the robot.

   > **Note** The debugging is also restarted automatically whenever the current robot is modified or replaced by another robot in Design Studio.

7. If the robot has input variables, you can edit the variables in the Input panel. Click **Stop** ⏸, change the values, and then press Enter to restart debugging with the new input values.

   You cannot edit the input values while a debug is running. To change the input values, you must first restart the debugging.

Debug from the Current Location in Design Mode

1. To start debugging from the current location, in design mode, click **Start Debug From Current Location** .

   The Robot Editor switches to debug mode and executes, as directly as possible, to the location that is current in design mode.

   When that location is reached, debugging is stopped.

2. Click **Run** ▶ to continue debugging from this location.

   This feature is useful to debug a part of the robot, such as a specific branch or a specific iteration of a loop action.

   If Design Studio is already debugging the robot when you click **Start Debug From Location** , it must restart the debugging before it executes to the location, and prompts you for permission.

3. Click **OK**.

Return to Design Mode from a Debugging Location

During debugging, you can return to a location using Go to this Location in Design Studio Window, or Go to. The following steps demonstrate returning to design mode for both options.

1. When debugging has stopped at some location in the robot, in debug mode, click **Go to** to switch back to design mode at the same location.

   This allows you to closely examine that location in design mode, and perhaps modify the steps around that location, or modify some other part of the robot.

   Alternatively, you can switch to design mode and go to the location where a value was returned.

2. On the Input/Output tab, select the value in the Output panel and click **Go to** in the lower right corner.

   This is useful if a value has not been extracted correctly, and you want to find out why.

   You can also switch to design mode and go to the location where an API exception was reported, or where an error occurred.
3. When you view an API exception in the API Exceptions tab or the API Exception sub-tab in the State tab, click **Go to** next to the location code to go to the location where the API exception was generated. Click **Go to** to the right of a specific error, to go to the location where that error occurred.

   This is useful when you want to find the reason for the error and fix the problem.
4. When finished with design mode, you can resume the debugging. If you have not modified the robot, you can click **Run** .

   If you have modified the robot, the debugging is automatically restarted, so you cannot resume it directly. Instead, you can click **Start Debug From Current Location** to start a new debugging session from the current location in design mode.

Use Breakpoints

While debugging, you can set a breakpoint to make Design Studio stop at a specific step.

1. In the Robot View, right-click a step and select **Toggle Breakpoint**.

   The breakpoint is indicated by the Breakpoint icon in the step.

   During debugging, Design Studio stops at the breakpoint.
2. To stop debugging when a certain action takes place, enter **Stop When** parameters.
3. Click **Run** to resume debugging.
4. To remove a breakpoint, right-click the step and select **Toggle Breakpoints** .
5. To remove all breakpoints from multiple steps, select one or more steps and click **Remove Breakpoints** .

   All breakpoints are removed from the selected steps.
6. To remove all breakpoints in a robot, click **Remove All Breakpoints**.

   All breakpoints are removed from the robot.

Single Stepping

You can use single stepping to execute one step at a time in debug mode. Single Stepping is useful if you want to examine the execution very closely.

Note You can single-step when Design Studio is ready to start a new debug, or when it has stopped during a debug.

1. To execute the next step, click **Single Step** .

   When that step has been executed, execution is stopped.
2. Click **Single Step** again to execute the next step, and so on.
3. To resume normal execution, click **Run** at any step.

Step Into

1. If you used the Group step to create a group containing several steps, create a breakpoint at the group step in the same way a step was created at any other step.
   - If the group is collapsed using Single Stepping, Design Studio treats the group as a single step and will not enter any of the grouped steps.
   - If the group is expanded, the steps in the group are visited when using single stepping.
   - If a group is collapsed, using the Step Into icon, instead of the Single Step icon, it results in the debugger stepping into the group and visiting each step.
2. To leave the group and advance the debugger to the step following the Group step, click **Step Out**.

Configure Robots

A robot is configured using the properties described below. The **Robot Configuration** window appears when you click **Configure Robot** (for Default browser engine or for Classic browser engine) on the toolbar. You can also select **Configure Robot** from the **File** menu.

Basic Tab

Default Options

Configure the default options for the step actions of the robot. See Default Options in Basic Robot Configuration for more information.

Robot Comment

Enter a comment about a robot.

Robot Tags

Create one or more tags for a robot. The tags are displayed in the **Tags** column under the **Repository** > **Robots** page in Management Console. You can use the tags to filter the list of robots in the Management Console. Tags can contain letters, numbers, and underscore. You can use 255 characters in tags. If you type more than that number, only the first 255 of them are saved.

Human Processing Time

This option helps you specify the time in minutes required for a person to perform the same task that the selected robot performs during its run. The difference between the specified value and the actual robot run time is displayed in the "Manual processing time saved" table of the Overview report in the Kofax Analytics for RPA.

Advanced Tab

HTTP Client (Classic browser only)

The client used to make HTTP requests to remote sites.

NTLM Authentication (Classic browser only)

Authentication type	Description
Standard	Kofax RPA has built-in support for the NTLM authentication scheme over HTTP (both for proxies and target systems).
JCIFS	For Classic browser robots, in case Kofax RPA cannot authenticate with your system, an alternative NTLM authentication engine named JCIFS is available. To use JCIFS, download the JCIFS library version 1.3.16 JAR file from http://jcifs.samba.org and place it in the lib folder of your Kofax RPA installation directory and select "JCIFS" as the NTLM authentication to use in the configuration of the robot.

Default Waiting (Default browser only)

Specifies the default wait criteria for the robot. For information about wait criteria and its options, see Use Wait Criteria.

Enable Private HTTP Cache

Select this option to enable private HTTP caching. Pages received from a server marked with Cache-Control: private contain information specific to a particular client and are not stored in the global HTTP cache. To never cache such pages you should disable this option. To store such pages in a robot specific cache you should enable this option. The downside to enabling private HTTP caching is using more memory per robot. If you are running a large amount of robots on the same server, you can disable this option to decrease their memory footprint.

Private HTTP Cache Size

This property specifies the maximum amount of memory to use for the private HTTP cache. The size is specified in kilobytes. You should beware of setting this number high because each and every robot instance running could potentially use this amount of memory in addition to its other state. All pages stored in the HTTP cache are compressed, so simple pages with text content will require very little memory. Also note, that only pages with Cache-Control: private or similiar will ever be stored in the private HTTP cache. Pages marked for non-private caching will go into the global HTTP cache shared by all robots.

Proxy Server

Use Proxy Server

Specify an optional proxy server to use for all page and data loading done by this particular robot. You should use this property only rarely. We recommend to specify one or more proxy servers for the entire Kofax RPA installation. This can be done in the Design Studio Settings window. See Proxy Servers for more information. The proxy server specified for a particular robot will override proxy servers specified in the Design Studio settings.

Design Mode

Select the Design Mode Execution for your robot that use the Default (WebKit) browser engine. Available options are:

- Minimal Execution (Direct)
- Smart Re-execution (Full)

The **Avoid External Re-execution** option, which is available for **Smart Re-execution**, ensures that steps are never re-executed, even when the cached result of the previous execution cannot be used. This option should only be used when there are requirements from the interaction with the external world to avoid re-execution, for example if this would result in incorrect or duplicate data in a partner's system.

Version

Shows the version of the saved robot and the version of the Design Studio the robot was last edited in.

Default Options in Basic Robot Configuration

In the Configure Robots -> Basic -> Default Options dialog box, you can configure default options of the action.

All Loading Tab

This tab contains general loading properties, used for both page loading and other types of loading.

Credentials

As credentials, you can either use standard username/password credentials or OAuth credentials. If you select Standard, the following properties are available:

Username

This property specifies which username to use for login. The value can be specified in several ways using the Value Selector. Note that this username is used only for HTTP and FTP based login. These types of login normally cause a pop-up window prompt in a browser, and are different from the typical and more commonly used form based method for login.

Password

This property specifies which password to use for login. The value can be specified in several ways using the Value Selector. Note that this password is used only for HTTP and FTP based login. These types of login normally cause a pop-up window prompt in a browser, and are different from the typical and more commonly used form based method for login.

See Web Authentication for more information.

Alternatively, you can use OAuth credentials. OAuth is the preferred authentication mechanism for a number of popular REST APIs. See OAuth on how to use OAuth in Design Studio and Management Console.

Client Certificate

This property defines where to get a client certificate when loading from HTTPS URLs. The client certificate may be given directly, or you may refer to one of those that have been installed as described in HTTPS Client Certificates. The options are:

- **Automatic** Selects that one of the installed certificates which is marked as "default". If no certificates have been installed, or none of the installed ones are marked as "default", no client certificate will be used for the connection.
- **Installed Certificate** Selects one of the installed certificates by giving its ID, which was defined when it was installed.
- **Certificate From Variable** The certificate is given as the value of a binary variable. The certificate's password must be given as well, as the value of another variable.
- **ID From Variable** Selects one of the installed certificates by giving its ID as the value of a variable.

SSL/TLS

This property specifies the version of SSL/TLS to use when loading from HTTPS URLs. This is configurable, because some sites give different results depending on the SSL/TLS version that is used, for example because they do not work with TLS or because they do not accept the weaker security that SSL provides. It is possible to select between SSLv3 or its successor TLS, or to accept both in negotiation with the site. In either case, it is possible to specify that the protocol negotiation should be initiated with SSL Hello or TLS Hello.

Verify SSL Certificates (Default browser only)

If this option is selected, the robot verifies the presented SSL certificate.

Browser to Emulate (Classic browser only)

This property specifies which browser you want the action to appear as when it loads something. Appearing as an older browser can sometimes provide you with a simpler page. However, we generally recommend that you stay with the default because this will normally cause the remote web server to serve up JavaScript and so on, that is compatible with Kofax RPA's built-in browser.

For anonymization purposes, you should rather change the "HTTP User Agent" property which is described below.

Authentication Method (Default browser only)

Select authentication protocol to use. You can select from NTLM and Negotiate. If you select Negotiate, you can add specific Negotiate protocol parameters. See Web Authentication for more information.

HTTP User Agent

This property specifies the exact text to send as the value of the HTTP User-Agent header. For the Classic browser by default, the user agent header value is derived from the "Browser to Emulate". Changing the User-Agent header - perhaps in a random manner, obtaining the value from a variable - can be useful to better blend in with other requests to a remote web server.

Language

This property specifies which browser language to appear to have, both when queried by JavaScript and when loading something.

Screen Size

This property specifies which screen size to appear to have, if queried by JavaScript.

Flash Version (Classic browser only)

This property specifies which flash version to appear to support, if queried by JavaScript.

Referred from this URL

This property specifies which URL you want the action to appear to have been referred from when loading something. If you do not specify a URL, the action will appear to be referred from the current page in the robot.

Enable Cookies

This property specifies whether you want cookies to be enabled.

HTTP Cache

This property specifies how you want the robot to use HTTP caching.

- **Default browser engine**

  The default setting **Enabled** enables the HTTP Cache and caches HTTP responses based on the rules of HTTP caching. **Disabled** option disables HTTP caching. **Aggressive** option overrides cache directives and enables caching of the resources that are otherwise not cached. The Aggressive option may be helpful to boost performance of high latency sites.
- **Classic browser engine**

  The default setting is **Standard**. Standard HTTP Cache mode enables the HTTP Cache and transparently caches HTTP responses based on the rules of HTTP caching. Selecting **Force Caching of JS and CSS** will override the rules of caching and force the robot to cache JavaScripts and style-sheets, in some cases this will boost the performance of high latency sites. Selecting **Disabled** will disable all HTTP caching.

Max. Number of Attempts

This property specifies how many times you want to attempt execution of the action if a load error occurs. The minimum value is "1".

Time Between Attempts (s)

This property specifies how many seconds to wait between each attempt to execute the action.

Timeout for Each Attempt (s)

This property specifies how many seconds each attempt to execute the action is allowed to take before timing out. The value must be greater than zero.

Additional Headers to Send

This property specifies an optional variable or a list that contains additional HTTP headers to send. The headers must be represented as a text, in the same format as in an HTTP message.

Store Received Status Code Here

This property specifies an optional variable in which to store received HTTP responses status code. The code will be an integer, and will correspond to the same response for which the received headers were obtained.

Store Received Headers Here

This property specifies an optional variable in which to store received HTTP headers. The headers will be represented as a text, in the same format as in an HTTP message.

Ignore Load Errors

This property specifies whether to ignore errors when the loading of a page or resource fails.

Page Loading Tab

This tab contains properties used specifically for loading pages.

Page Content Type

This property specifies the content type of the loaded pages. Usually, the "Automatic" setting should suffice, but you may also directly specify a content type, either for all pages loaded in the action or for only some of them, depending on their URL.

Page Content Encoding (Classic browser only)

This property specifies the character encoding of the loaded pages. The "Automatic" setting should work in most circumstances, but it may be necessary to specifically set the encoding of the pages, either for all pages and text resources (e.g. external JavaScript files) loaded in the action or for only some of them, depending on their URL.

Form Parameter Encoding (Classic browser only)

This property specifies which character encoding to use for encoding field values when submitting a form. Usually, the "Automatic" setting will be sufficient, but if you encounter problems with incorrect characters in the submitted data, you can try to set a specific encoding here.

Follow Meta Redirections

This property specifies whether to follow <meta>-tag redirections, i.e. redirections defined by a <meta> in a loaded page.

Apply XSL Style Sheets (Classic browser only)

This property specifies whether to apply referenced XSL style sheets when loading a page that contains XML. For instance, an XML document intended to be displayed in a browser can contain a reference to an XSL style sheet that transforms the XML document into HTML.

Use Preloading (Classic browser only)

Pre-load Javascript and style sheets in HTML documents, i.e. start loading the resources as soon as the HTML response from the web server is received. Enabling this option cuts down on the time that each step spends waiting for resource loads to complete because the loads are initiated before the robot reaches a state where it must block until the resources are ready.

Load Frames

This property specifies whether to automatically load the frames of a page.

Load Unsupported Formats (Classic browser only)

This property specifies whether to load the content of unsupported formats. An unsupported format is one that Design Studio cannot parse and present in the Page View, e.g. a video format. Often resources of such formats may take long to load and the robot may not be able to access the content so loading the content of the resource may just slow down the robots execution. The headers and status code are always obtained from the response even if loading of the contents is turned off. An additional use of this feature is to get the header information about a resource without loading it (in the case this is in an unsupported format). This is slightly different than just using a HEAD request because a HEAD request only obtain the headers for the initial request and not resources obtained through META or JavaScript redirects.

Images to Load

This property specifies whether to automatically load the images of a page. Usually, it is not necessary for the robot to load the images, but you may choose to load the images of a page if you suspect that the image loading has side-effects that are necessary for the navigation of the page. In that case, you may choose to load all of the images on the page or only some, depending on their URL.

Max. Loads Per Window (Classic browser only)

This property specifies the maximum number of page loads per window allowed in the action. This can be used for stopping the page loading if an infinite loop of redirections or reloads are encountered. Such an infinite loop will eventually cause the action to time out anyway, but by detecting this earlier, you can avoid putting excessive load on the web server that you are accessing. The action will generate an error if it stops because the maximum number of page loads has been reached.

Max. Window Nesting (Classic browser only)

This property specifies the maximum number of window allowed nested inside each other. A window in this context can mean several things. In a frameset each frame is a window, so the Max. Window Nesting property specifies how many frames a loaded page can have inside each other. The action will generate an error if it stops because the maximum number of nested windows has been reached. If you check the Ignore Load Errors option, the step action will complete successfully and output a page containing no more than the maximum number of nested windows. If you leave the field blank, there is no limit to the window nesting level.

Page Changes

This property allows you to change the loaded pages on-the-fly before they are parsed. This is useful for things like correcting syntax errors, solving other parsing problems, removing or changing tags, and so on.

The changes are done by specifying one or more data converters that will be applied to the pages before the parsing. You can either specify data converters to apply on all pages, or data converters to apply to individual pages depending on their URL.

The most common data converters to use for page changes are Replace Text, Replace Pattern, and Remove Tags. When configuring the data converters, remember that they will be applied to the original, unprocessed text of the page, before decoding of ampersand encodings etc. Therefore, it is recommended that you obtain this text, e.g. using the View Source function of your standard browser. In the data converter configuration window, you can paste the text into the input area in the lower left corner to test that the converter performs the desired action on the text.

> **Note** If you want to make changes to JavaScript, use the JavaScript Changes property on the JavaScript Execution tab instead.

Page Error Test (Classic browser only)

This property specifies a custom test for website errors based on the content of the page. Usually, a website will send an error code when something goes wrong, but if this is not sufficient to detect errors, this property can be used.

If Same for All Pages is selected, the test is performed on all pages. By selecting Depends on URL, you may set up individual tests for particular (groups of) URLs.

You can specify a pattern that matches an error page (by selecting Pattern Matches Rejected Page), or you can specify a pattern matching all other pages (by selecting Pattern Matches Accepted Page).

Output Page If Error (Classic browser only)

This property specifies whether or not to output a page even if the website sends an error code. If disabled, any website error will cause the action to fail. If enabled, certain website errors are accepted and the page is output, whereas all other server errors will still cause the action to fail. The accepted website errors are 403 Forbidden, 404 Not Found, and 500 Internal Server Error.

Output Page If Timeout

This property specifies what occurs when the action times out. If disabled, the action fails in the event of a timeout. If enabled, the result received so far is the output. Note the following for this property:

- When older Default browser (WebKit) robots that have "false" as default for this property are opened in a new version of Kofax RPA, "Output Page If Timeout" is set to false in the robot's browser configuration.
- When creating a new Default browser robot, the "Output Page If Timeout" property is set to true by default.
- When creating a new Classic browser robot, the "Output Page If Timeout" property is set to false.

URL Filtering Tab

This tab controls the configuration of what URLs to block, for instance to block the loading of ad frames. The settings on this tab are only valid for Web automation robots.

Filter URLs

This property specifies whether to block loading of certain URLs. The URLs to be blocked are specified as pattern in the list of Included URL Patterns and Excluded URL Patterns, respectively. Only URLs occurring in the following tags may be blocked:

Included URL Patterns

If specified, only URLs that match these patterns will not be blocked. Each pattern must be written on a separate line. A URL that matches one of these patterns may still be blocked, if it matches one of the "Blocked URL Patterns" specified below. A typical use of this property is to specify a pattern that matches only URLs of a single domain, so that only frames and scripts from that domain are loaded.

- `<frame src="URL">`
- `<iframe src="URL">`
- `<script src="URL">`

If a URL is blocked no request is performed and content is left empty. In the case of frame and iframes there will still be a new window in the page view and this will be showing a message explaining why the load was not performed. An 🚫 icon on the tab of the window in the page view will indicate that the URL was blocked.

Blocked URL Patterns

This property specifies which URLs to block. This is specified by writing a list of patterns with one pattern on each line.

JavaScript Execution Tab

This tab contains properties used for executing JavaScript. These properties allow you to customize the JavaScript execution in case the default automatic execution does not work correctly. Note that using the options of the Logging tab, the Log window in Design Studio can provide information about the JavaScript execution performed during execution of the robot. You can use this window to understand which JavaScripts are executed, which errors occur, and so on.

> **Note** Some of the properties on this tab differ in the Default and Classic browser engines.

Execute JavaScript

This property specifies whether JavaScript should be executed.

Ignore JavaScript Errors (Classic browser only)

This property specifies whether errors that occur during JavaScript execution should be ignored. In many cases, such errors can safely be ignored, as long as the outcome of the execution is as desired.

Ignore Alert Messages

If this option is selected, an alert message produced by the JavaScript method `alert()` is ignored, otherwise an error is generated. Alert messages are typically created by JavaScript to alert the browser user of an invalid action, such as trying to submit a form that is not correctly filled out.

A common way to handle alert messages in a robot is to configure this property to ignore them, and then configure the Store Ignored Alert Messages Here property below so that ignored alert messages are stored in an appropriate variable. In a subsequent step, this variable can then be tested and suitable action taken if it contains an alert message.

Store Ignored Alert Messages Here

This property specifies a variable in which ignored alert messages should be stored. This is relevant only when the Ignore Alert Messages option has been selected above.

Enable Timer Events (Classic browser only)

This property specifies whether timer events should be executed. Timer events are events that occur after a specified amount of time and can be set up either by JavaScript using setTimeout() or setInterval() or when a <meta> redirection specifies that the page should be redirected after a number of seconds.

Max. Wait for Timer Events (ms) (Classic browser only)

This property specifies the maximum number of milliseconds to wait for timer events to be executed, since the beginning of the execution of the action. For example, if a page loads in 3000 ms and sets up some timer events, and this property has been set to 30000 ms, only timer events that expire within 27000 ms after the page load will be executed. Note that the wait for the timer events is done either in real-time or just emulated, depending on the Wait Real-Time for Timer Events property below.

Wait Real-Time for Timer Events (Classic browser only)

This property specifies whether to actually wait the time specified by the Max. Wait for Timer Events property above, or to just emulate the wait and execute any triggered timer events immediately. For many timer events, it is not necessary to actually wait the period specified, and thus the robot can immediately proceed. However, if the reason for the timer event is that e.g. one must wait for the web server to process results, it may be necessary to wait real-time.

Delay Between Key Presses (ms)

This property specifies the amount of milliseconds to wait between key presses when emulating a user typing on a keyboard. This only has relevance for step actions that enter text into a form.

Use CSS Style Sheets (Classic browser only)

This property specifies whether to load and parse CSS style sheets during the execution of the robot. This may be necessary for the JavaScript on a page to work correctly. On the other hand, disabling the use of style sheets can speed up the execution of page loads. Even if this option is disabled, the page view may still load style sheets for display purposes, but this loading will not take place when the robot runs on the server.

JavaScript Changes

JavaScript changes are an optional list of data converters that will be applied to the JavaScript before it is executed. The changes are applied to all JavaScript that is executed, both event handlers, internal and external scripts. The data converters are useful for making changes and corrections to the JavaScript. For example, they can be used to define variables that the JavaScript expects to have been defined by VBScript. The most common data converters to use for this purpose are Replace Text and Replace Pattern.

When configuring the data converters, remember that they will be applied to the original JavaScript. Therefore, it is recommended that you obtain this text, e.g. using the View Source function of your standard browser for inline JavaScript or by downloading the file in case of external JavaScript. In the data converter configuration window, you can paste the text into the input area in the lower left corner to test that the converter performs the desired action on the text.

Important This option affects the way JavaScript is executed on pages that you load, that is the option is applicable to Load Page and Create Page steps.

JavaScript Polyfills (Default browser only)

The default Kofax RPA browser (WebKit) does not support some of the ES5 and ES6 JavaScript features. To enable support for new functionality, Kofax RPA can load predefined or custom JavaScript polyfills.

A polyfill is a piece of code (usually JavaScript on the Web) that provides modern functionality in browsers that do not natively support it. For example, a polyfill can replicate the functionality of an HTML Canvas element in Microsoft Internet Explorer 7 using a Silverlight plugin, or provide support for CSS rem units, and other features.

In case of error, the JavaScript console shows which JavaScript object does not exist. According to this information, a necessary polyfill can be found and applied to resolve an error.

Click **Add** (+) to select an object or API that you want the browser to support. You can also include a custom code that provides support for certain JavaScript objects or API. To include a custom implementation, click **Add** (+) and select **Custom** in the list. The Custom dialog box contains two panes, such as Name and Code. Specify the name of your code implementation in the Name pane and paste your JavaScript code to the Code pane.

The JavaScript object implementation code is executed before the page loads.

See the list of predefined polyfills in Predefined JavaScript Polyfills.

But there are lots of modern JavaScript constructions, where applying polyfills does not resolve an error. See the list of JavaScript known issues, containing some of them, below.

- The let statement

  CMAScript 2015 (6th Edition, ECMA-262)
- Constants

  CMAScript 2015 (6th Edition, ECMA-262)
- An arrow function expression () => {}

  CMAScript 2015 (6th Edition, ECMA-262)
- Default function parameters

  CMAScript 2015 (6th Edition, ECMA-262)
- for...of statement

  ECMAScript 2015 (6th Edition, ECMA-262)
- Rest parameters

  ECMAScript 2015 (6th Edition, ECMA-262)
- Method definitions

  ```
  var obj = {
            property( parameters… ) {},
            *generator( parameters… ) {},
            async property( parameters… ) {},
            async* generator( parameters… ) {},

            // with computed keys:
            [property]( parameters… ) {},
            *[generator]( parameters… ) {},
            async [property]( parameters… ) {},

            // compare getter/setter syntax:
            get property() {},
            set property(value) {}
          };
  ```

  ECMAScript 2015 (6th Edition, ECMA-262)

  ECMAScript 2016 (ECMA-262)
- Fetch API

  ```
  fetch('http://example.com/movies.json')
            .then(function(response) {
              return response.json();
            })
            .then(function(myJson) {
              console.log(JSON.stringify(myJson));
            });
  ```
- The Request interface of the Fetch API represents a resource request

  ```
  var a = new Request(url);
  ```

Plugins Tab

Default browser only

This tab contains parameters to add and configure simulated plugins when using the browser.

Simulate support for

- **From List**: Click the plus sign and select a plugin from the list.
- **From JSON Variable**: Construct your own plugins using a JSON variable.
  See Plugin Simulation from JSON Variable for more details.

Javascript Event Handlers Tab

Classic browser only

This tab contains properties that determine which JavaScript event handlers should be executed. Note that using the options of the Logging tab, you can obtain information about which event handlers are executed during execution of the robot, in the Log window in Design Studio.

Enable Click Event Handlers

This property specifies whether onclick event handlers, if any, are executed when clicking a tag.

Enable Change Event Handlers

This property specifies whether onchange event handlers, if any, are executed when modifying a value in a form.

Enable Form Event Handlers

This property specifies whether the onsubmit and onreset event handlers, if any, are executed when submitting or resetting a form, respectively.

Enable Load Event Handlers

This property specifies whether the onload and onunload event handlers, if any, are executed when loading / unloading a page or loading an image.

Enable Mouse Event Handlers

This property specifies whether the onmouseover, onmouseenter, onmouseout, onmouseleave, onmousedown and onmouseup event handlers, if any, are executed when moving the mouse over or clicking a tag.

Enable Drag Event Handlers

This property specifies whether the ondrag, ondragstart, ondragenter, ondragleave, ondragend and ondragover event handlers, if any, are executed when the mouse is dragged over a tag.

Enable Key Event Handlers

This property specifies whether the onkeydown, onkeypress and onkeyup event handlers, if any, are executed when entering text.

Enable Focus Event Handlers

This property specifies whether the onfocus, onfocusin, onfocusout, onblur, onactivate, onbeforeactivate and ondeactivate event handlers, if any, are executed when a tag or document gains or loses focus.

Enable Capture Event Handlers

This property specifies whether the onlosecapture event handlers, if any, are executed when a tag or document loses mouse capture.

Enable State Change Event Handlers

This property specifies whether the onreadystatechange event handlers, if any, are executed when the state of a tag or ActiveX object changes.

Enable Error Event Handlers

This property specifies whether the onerror event handlers, if any, are executed when an error occurs.

Logging Tab

Classic browser only

This tab contains properties used to determine the level of logging of the JavaScript execution performed during execution of the robot. The logging information can then be obtained in the Log window in Design Studio and may be used to understand which JavaScripts are executed, which errors occur, and so on.

Log All JavaScript Source

This property specifies whether the source of all JavaScripts executed are logged. Note that the JavaScript may declare functions that themselves are not executed until they e.g. are called from an event handler. You may use the JavaScript Changes property to make changes and corrections in the JavaScript.

Log JavaScript Execution Trace

This property specifies whether to log a detailed trace of the JavaScript execution. This trace includes all functions that are called and all properties that are set or retrieved.

For example, when

```
location.href =
"http://www.kofax.com"
```

is executed, the trace will read

```
GET location = [location]
```

followed by a

SET [location].href = "http://www.kofax.com"

Include Function Source in Trace

This property specifies whether to include the source code of the functions executed, in the trace of the JavaScript execution.

Note This option is relevant only if the **Log JavaScript Execution Trace** option is selected.

Log JavaScript Event Handlers

This property specifies whether to log executed JavaScript event handlers.

Log Timer Events

This property specifies whether to log executed timer events. Timer events are events that occur after a specified amount of time and can be set up either by JavaScript using setTimeout() or setInterval() or when a <meta> redirection specifies that the page should be redirected after a number of seconds.

Log Loads

This property specifies whether to log all page and resource loads.

Log XML HTTP Requests

This property specifies whether to log XML HTTP Requests sent.

Log Absolute Positioning

This property specifies whether to log the absolute positioning of visual components such as menus that are positioned using JavaScript.

Max. Log Entries

This property specifies the maximum number of log entries allowed. The minimum value is "1". If there are more log entries than allowed, the first log entries will be discarded and will thus not appear in the Log window.

Legacy Tab

This tab contains properties that should not be changed in most cases. The legacy properties have been introduced to ensure backward compatibility with older version of the product. If a new feature is introduced in the product that conflicts with the way things were done previously, an option is introduced on this tab to ensure old robots will work and be backward compatible. On this tab, the default setting represents the current way and the other setting is the old way.

Format Handling

This option specifies how to handle different document formats.

Download non-HTML (Default)
Kofax RPA loads all supported non-HTML content to work with. You can preview CSV, JSON, text, Excel, XML, and binary content and apply step actions to them. Use the Preview button to change the type of the content.

Classic loading
You can specify how to handle different document formats for classic browser engine.

JSON

This property specifies how to handle JSON, which is one of the common response types when calling web services. The default is **Do Not Convert**. You can convert the JSON to XML, which makes it easy to handle in a standard way in Design Studio. Alternatively, it can be converted to HTML. The HTML is more readable than XML, but somewhat harder to extract from automatically.

Convert XML to HTML

This property specifies whether to convert XML documents to HTML documents or keep them as-is. It is mainly used in old robots that work on the converted documents, as this was previously the only option. In newer robots, it is normally more convenient to work directly on the XML structure.

> **Note** To view XML content with the applied XSLT transformation in the Applications view, select **Configure Robot** > **Default Options:** > **Configure** > **Legacy** > **Format Handling** > **Classic loading** and clear the **Convert XML to HTML** option.

Convert Excel to HTML

This property specifies whether to convert Excel documents to HTML documents or keep them as-is. It is mainly used in old robots that work on the converted documents, as this was previously the only option. In newer robots, it is normally more convenient to work directly on the Excel document, as this provides a faster and more spreadsheet-like display and user interface.

CSV

This property specifies whether to convert CSV documents to HTML documents or keep them as text (in a PRE-tag). It is mainly used in old robots that work on the text representation, as this was previously the only option. In newer robots, it is normally more convenient to work on a HTML table representation of the CSV document and use the full power of Design Studio when doing so. It is assumed that the CSV documents is encoded using commas (,) as separator character, double quotes (") as quoting character and backslash (\) as escape character. If the document you are loading does not conform to this convention you should use the Convert to Text option and work on the document as text, e.g. using the Extract CSV step action.

Migration Tab

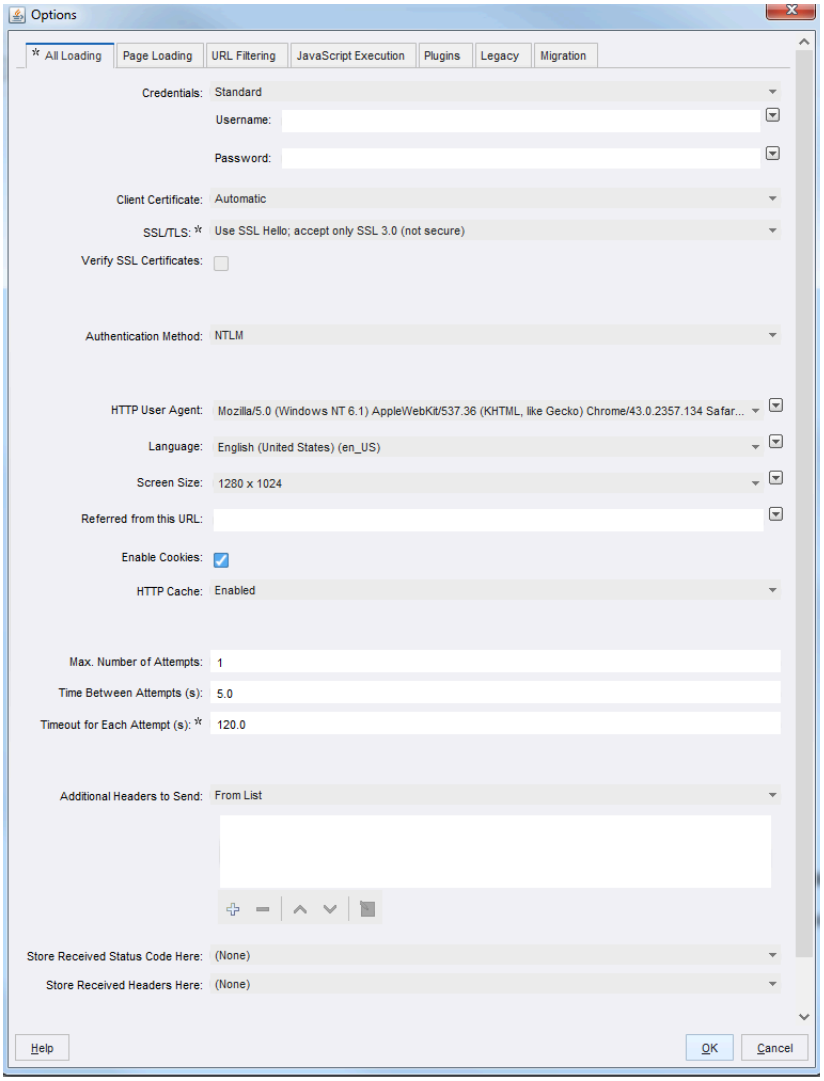
Default browser only

This tab contains a migration log of the robot that was migrated from the Classic browser to the WebKit (Default) browser. The log lists all values that were changed during the migration. See Migrate a Robot to a Different Browser Engine for details.

Show Changes from Default Robot Configuration

At times it is difficult to see non-standard parts of a robot configuration. For example, a Page Load step might be assigned a longer timeout than the default value of 60.0. To investigate the non-standard configurations, you may have to navigate through Design Studio to locate a property that is changed from the default setting. You would also need to remember the default values for all the properties.

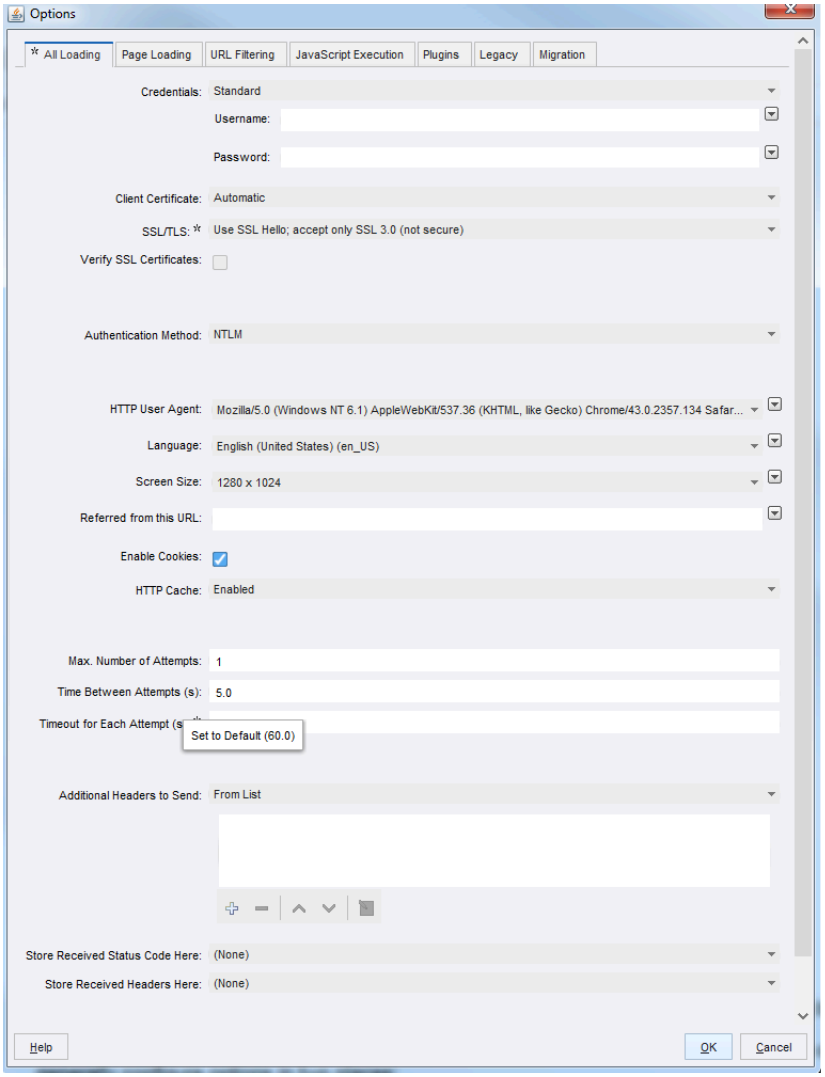
In Design Studio, you can use Show Changes to avoid these manual steps. Properties with a well-defined default value are marked with an asterisk * next to the property name if the value is changed, as shown in the following figure.



Credentials: Standard
Username:
Password:
Client Certificate: Automatic
SSL/TLS: * Use SSL Hello; accept only SSL 3.0 (not secure)
Verify SSL Certificates: □
Authentication Method: NTLM
HTTP User Agent: Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/43.0.2357.134 Safar...
Language: English (United States) (en_US)
Screen Size: 1280 x 1024
Referred from this URL:
Enable Cookies: ☑
HTTP Cache: Enabled
Max. Number of Attempts: 1
Time Between Attempts (s): 5.0
Timeout for Each Attempt (s): * 120.0
Additional Headers to Send: From List
Store Received Status Code Here: (None)
Store Received Headers Here: (None)

Notice that the tab also displays an asterisk. This indicates that some property on the tab is changed.

Right-click the property name or the asterisk to reset the value to its default. The context menu typically displays the default value.



Credentials: Standard

Username:

Password:

Client Certificate: Automatic

SSL/TLS: * Use SSL Hello; accept only SSL 3.0 (not secure)

Verify SSL Certificates: □

Authentication Method: NTLM

HTTP User Agent: Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/43.0.2357.134 Safar...

Language: English (United States) (en_US)

Screen Size: 1280 x 1024

Referred from this URL:

Enable Cookies: ☑

HTTP Cache: Enabled

Max. Number of Attempts: 1

Time Between Attempts (s): 5.0

Timeout for Each Attempt (s

Set to Default (60.0)

Additional Headers to Send: From List

Store Received Status Code Here: (None)

Store Received Headers Here: (None)

The preceding figure shows the Timeout for Each Attempt property on the Options dialog box.

Show Changes works differently on the Options window when it is used to configure step options. You can generally configure options in two places:

- From the Robot Configuration
- On steps that may depend on these options

In both places, you click a button to open the Options window to configure the options, but the default value is different for each situation. If you open the window from Robot Configuration, the default is a fixed application default, which displays the values provided by Design Studio. If you open the window from a step configuration, the default is the one defined under Robot Configuration (robot default). That is, a step inherits the option values from the robot configuration unless they are explicitly changed for the step. For example, if the option "Enable Cookies" is cleared in the robot configuration, all steps that depend on this option also use this setting unless you have explicitly changed them for the step. An asterisk on a step option indicates the step uses a value different from the one defined in the robot configuration, which is not necessarily different from the application default.

There is another way in which an asterisk on the step's Options dialog has a different meaning than on the Robot Configuration's Option window. For options on the step's Option window, an asterisk means that the given option is deliberately set to a fixed value, which is not necessarily different from the corresponding Robot Configuration value. Also, the value is not influenced by any changes in the corresponding Robot Configuration value. For example, if initially the Timeout for Each Attempt property for a step is set to 120 and the corresponding value under the Robot Configuration is 60, an asterisk appears next to the option. If the Robot Configuration value is changed to 120 so that the two values are actually the same, the step's value is still marked with an asterisk. If the value for the robot is again changed from 120, the value for the step stays unchanged at 120. If you change the step's value back to its default value (the value from the Robot Configuration) using the context menu or double-click, the step's value uses the Robot Configuration value and subsequently follows any changes made.

Another situation where the default value may depend on other configuration choices applies to error handling step configuration.

Migrate a Robot to a Different Browser Engine

Web Automation part of Kofax RPA currently comes with two different browsers: The classic engine for backward compatibility with previous Kofax Kapow products and the default (WebKit) engine for legacy web applications. However, these browsers might not be compatible with every internal application or Internet web site. If you encounter problems using any of the browsers, you can migrate your robot to other browser type in Design Studio using the following procedure. See Choose your browser for more information.

Migrating a Robot to the Classic Browser

1. Right-click a Web Automation robot identified by a blue icon in the tree view and select **Migrate**.
2. Select robots (and snippets) that you want to migrate and click **Next**.
3. In the **Backup** step, specify the folder to contain the backup files and click **Finish**.

   After Kofax RPA migrates your robot, you can see that the color of the robot icon is changed to orange .

   > **Note** When migrating a WebKit robot that has steps with legacy waiting criterion, this waiting criterion is converted to "Wait Real-Time for Timer Events" and "Max. Wait for Timer Events (ms)".

Migrating a Robot to the Default Browser

1. Right-click a Web Automation robot identified by an orange icon in the tree view and select **Migrate**.
2. Select robots (and snippets) that you want to migrate and click **Next**.
3. In the **Backup** step, specify the folder to contain the backup files and click **Finish**.

   After Kofax RPA migrates your robot, you can see that the color of the robot icon is changed to blue .

Web Automation Step Actions and Data Converters

In Design Studio, a short description is shown with each action and data converter. Click **More** next to the description to see additional information about the action or data converter associated with the description. You can also click help ? to get onscreen assistance associated with a selected step action or data converter.

Several of the actions, such as Extract, can run extracted text through a list of data converters and sort the result in a variable.

See Step Actions for the Web Automation steps and their options.

A data converter processes extracted text based on parameters you define. For example, the Extract Number data converter accepts an input text containing a number and outputs a text containing the same number in a standardized format.

Because a data converter takes a text as input and outputs another text, data converters can be chained so that the output of one data converter becomes the input to the next data converter. The final output is the text output of the last data converter in the data converter list. For example, if the list of data converters contains the converter Convert to Upper Case, followed by a Remove Spaces data converter, the input text to the list is "R oboMa ker", is output as "ROBOMAKER".

See Data Converters for information about available data converters.

Step Actions

This topic provides an overview of the available step actions.

You can add the most commonly used steps directly to the Insert Step menu that is available when you right-click a connection in the Robot view in Design Studio. See General Editing for details.

Standard

This category contains the most commonly used step actions.

Action	Description
Assign Variable	Assigns a value to a variable.
Call Desktop Automation Robot	Creates a step to call a Desktop Automation robot from a Web Automation robot.
Create Page	Creates a new page.
Desktop Automation	Deprecated. See "Call Desktop Automation Robot."
Open Variable	Opens a variable attribute - or a variable of simple type - in the view.
Load Page	Loads a web page from a URL.
Return Value	Returns a value from the robot.
Store in Database	Stores a value in a database.

Action	Description
Test Value	Causes execution beyond the step to stop or continue depending on a boolean value.

Assign/Transform Variable

This category contains the most commonly used step actions.

Action	Description
Assign Variable	Assigns a value to a variable.
Convert Variables	Converts the values of one or more variables by running them through data converters and storing the results in the same or other variables.
Transform XML	Transforms XML using XSLT.

Browser Session

This category contains step actions for saving and restoring entire browser sessions, as well as for extracting and manipulating cookies and HTML 5 web storage.

Action	Description
Save Session	Saves a session in a variable for later restoration by another robot run.
Restore Session	Restores a session in a variable previously saved by another robot run.
Extract Cookie	Extracts the value of a cookie matching patterns for name, domain, and path.
Create Cookie	Creates a cookie with the specified domain, path, name, and (optionally) value.
Remove Cookie	Removes one or more cookies matching patterns for name, domain, path, and value.
Extract Web Storage	Extracts data from the local and/or session storage. The data is stored in a variable in JSON format.
Load Web Storage	Loads data into the local and/or session storage. The data must be specified in JSON format.
Clear Web Storage	Clears data in the local and/or session storage.

Browser Windows

This category contains step actions for opening, selecting and closing browser windows.

Action	Description
New Window	Creates a new window.
Set Current Window	Selects another window as the current window, such as the window that subsequent steps will work on.
Close Window	Closes a window.

Call Web Service

This category contains step actions for calling REST and SOAP web services.

Action	Description
Call REST Web Service	Calls a REST web service and loads the result into the current window or stores it in a variable.
Call SOAP Web Service	Submits a SOAP XML request to a web service and returns a SOAP XML response.

Click/Move Mouse

This category contains step actions that mimic clicking or moving the mouse to and from elements in the browser view.

Action	Description
Click	Emulates a mouse click on the found tag.
Move Mouse To	Emulates a mouse move to the found tag.
Move Mouse From	Emulates a mouse move away from the found tag.
Scroll	Emulates scrolling a document or tag.
Scroll To	Emulates scrolling the found tag into view.

Database

This category contains step actions that can store, retrieve, query or delete items in databases.

Action	Description
Store in Database	Stores a value in a database.
Find in Database	Finds a value in a database.
Calculate Key	Calculates the key used to store the value of the selected variable.
Delete from Database	Deletes an value in a database.
Query Database	Submits an SQL query to a database, and loops through the results.
Execute SQL	Executes an SQL statement on a database.

Enter Data in Form

This category contains step actions for entering data in web forms.

Action	Description
Enter Text	Enters a text into a text field in a form.
Enter Password	Enters a password into a password field in a form.
Press Key	Emulates pressing a key in a form.
Select Option	Selects an option in a drop-down box or a list box in a form.
Select Multiple Options	Selects multiple options in a list box in a form. Note: This action can only be used for list boxes, not drop-down boxes.
Set Checkbox	Selects or clears a check box in a form.

Action	Description
Set Range Value	Sets a numeric value that must be no less than a minimum given value, and no more than a maximum given value.
Select Radio Button	Selects a radio button in a form.
Select File	Selects a file to upload in a file field of a form.

Extract

This category contains step actions for extracting data. Data may be extracted in text or HTML form from a web site, or from other formats such as PDF, CSV, Excel and Flash. It is also possible to extract images or specific data about the HTML or XML source such as attribute values or link URLs.

Action	Description
Extract	Extracts some text, runs it through a list of data converters, and stores the result in a variable.
Extract Binary Content	Extracts binary content from the Browser View.
Extract Cell	Extracts content from an Excel page, runs it through a list of data converters, and stores the result into a variable.
Extract Column in Data Row	Extracts data from a cell in the current range to a variable.
Extract from E-mail	Extracts information from a saved email.
Extract Form Parameter	Extracts a form parameter from a form URL in the found tag.
Extract Flash Content	Extracts content from a Flash object.
Extract from PDF	Extracts text from a PDF document contained in a variable.
Extract Image	Extracts an image and stores it in a variable or a file. It can optionally store the content type and file name of the image in other variables.
Extract JSON	Extracts the part of a JSON value found by the JSON finder as a JSON value into a variable.
Extract Path	Extracts the absolute path of the element found by the finder.
Extract Property Name	Extracts the property name of a JSON value found by the JSON finder into a variable.
Extract Screenshot	Extracts an image from the current page and saves it in a variable.
Extract Selected Option	Extracts the text or value of the selected option, runs it through a list of data converters, and stores the result in a variable.
Extract Source	Stores the previewed data in a variable.
Extract Tag Attribute	Extracts a tag attribute from the found tag, runs it through a list of data converters, and stores it in a variable.
Extract Target	Extracts data from a URL target and stores it in a variable or a file. It can optionally store the content type and file name of the loaded data in other variables.
Extract URL	Extracts a URL from the found tag and stores it in a variable.

File System

This category contains step actions for accessing the file system. You may read, write and modify files and directories, loop over files in a directory, or test for the existence of a given file.

Action	Description
Load File	Loads data from a file, either into the browser window or to a variable.
For Each File	Loops through the files in a directory.
Write File	Writes a new file or appends to an existing file.
Test File Existence	Causes execution beyond the step to stop or continue depending on whether a specific file exists.
Get File Info	Fetches metadata about a file in the file system.
Copy File	Copies a file on the local file system where the robot is executed. The action generates an error if the destination file exists.
Delete File	Deletes the specified file or directory.
Make Directory	Creates a new directory.
Rename File	Renames a file or directory on the local file system where the robot is executed. The action generates an error if the destination (New Name) exists.

Loop

This category contains step actions for looping. You may loop through HTML structures, windows, comma-separated values, form values, Excel ranges, or crawl entire domains. For looping through HTML structures, you have two options: For Each Tag and For Each Tag Path. The For Each Tag step action is the simpler of the two; it is used to loop through the immediate children of the found tag, while the For Each Tag path can loop through similar tags at any depth within the found tag. To loop through a number of pages connected by Next links or the like, you must use the Repeat and Next step actions.

Note If the selected element does not contain any sub-elements to loop through, all For Each steps throw an error. For example, the For Each Tag Path step throws an error if the found tag does not contain any tags to loop through.

However, when using the For Each File step to loop through files in a directory, if no files are present in the directory, this is not regarded as an error, and no error is thrown.

Action	Description
For Each Tag	Loops through tags contained immediately inside the found tag.
For Each Tag Path	Loops through tags contained at any level inside the found tag.
For Each URL	Loops through the URLs contained in the found tag.
For Each Option	Loops through the options in a drop-down box or list box in a form, selecting one option in each iteration.
For Each Radio Button	Loops through a group of radio buttons, selecting one of the radio buttons in each iteration. The found tag must be one of the radio buttons in the group.
Loop Field Values	Loops through the specified values, entering one value in the text field in each iteration.
Repeat	Creates a repeat loop together with the Next action.

Action	Description
Next	Requests another iteration in a repeat loop created using the Repeat action.
Loop in Excel	Loops over the rows, columns, cells in the found range or over all the sheets in the Excel page.
For Each Data Row	Loops through data rows in a CSV file.
For Each Property	Loops through all properties of a JSON object.
For Each Item	Loops through a group of tags.
For Each File	Loops through the files in a directory.
For Each Browser Window	Iterates through the browser windows, setting each in turn as the current window.
For Each Text Part	Splits a text at a specified delimiter and loops through the parts.
Get Iteration	Gets the current iteration of an enclosing loop step.

Load Page

This category contains step actions for loading pages from a given URL or creating a new page based on already extracted content. If required, you can also specify the page load request at the basic HTTP level.

Action	Description
Load Page	Loads a web page from a URL.
Create Page	Creates a new page.
Raw HTTP	Performs a Raw HTTP request of the selected method.
Open Variable	Opens a variable attribute - or a variable of simple type - in the view.
View as Excel	Opens downloaded Excel content in an Excel view.
View as JSON	Opens downloaded JSON content in a JSON view.
View as XML	Opens downloaded XML content in an XML view.
View as CSV	Opens downloaded CSV content in a CSV view.
Extract Source	Stores the previewed data in a variable.

Make Snapshot

This category contains step actions for saving offline snapshots of web pages. To save an offline HTML copy of a page and its resources, use Make Snapshot. To save multiple interlinked HTML pages, use Rewrite Page and Rewrite Style Sheet.

Action	Description
Make Snapshot	Creates a snapshot of the current window, including its frames and resources.
Rewrite Page	Extracts the HTML content of the current window and additionally rewrites and outputs the links to style sheets, images and other pages.
Rewrite Style Sheet	Acts as a helper for Rewrite Page. It task is to rewrite links to other style sheets or images in a given style sheet.

Modify Page

This category contains step actions for modifying the current web page, such as by removing, replacing or inserting content.

Action	Description
Insert Tag	Inserts a new tag.
Replace Tag	Replaces the found tag with a new tag.
Remove Tags	Removes tags from found tags. The Remove rules are executed in the order listed below. Any tags matching one or more of the Except rules are not removed. Defining no Remove rules defaults to removing all tags.
Remove Tag Range	Removes a range of tags.
Hide Tag	Hides the found tag.
Unhide Tag	Unhides the found tag.
Divide Text	Divides the text in the found tag into pieces.
Remove Table Rows	Removes from the input <table>-tag all rows (<tr>-tags) that do not have a specified number of columns (<td>- and <th>-tags).
Transpose Table	Transposes (flips) the input <table>-tag by mirroring its <td>-tags along the top-left to bottom-right diagonal.
Normalize Table	Normalizes a table by inserting extra cells to eliminate rowspan and colspan. The content from the original cell is copied to the new cells.

Output

This category contains step actions for returning values to the API that called this robot, sending e-mail and writing to files or logs.

Action	Description
Return Value	Returns a value from the robot.
Send Email	Sends an email. Note that the email is not sent during execution in Design mode in Design Studio.
Write File	Writes a new file or appends to an existing file.
Write Log	Writes a message to the log. This is useful when debugging a robot.

Test

This category contains conditional actions for testing, such as stopping the execution down the current branch if some condition is satisfied. This condition may depend on the contents of the found tag, a variable, or the existence of a given window.

Action	Description
Test Tag	Causes execution down the current branch to stop or continue, depending on the contents of the found tag.
Test URL	Causes execution down the current branch to stop or continue, depending on the URL contained in the found tag.
Test Value	Causes execution beyond the step to stop or continue, depending on a boolean value.

Action	Description
Test Variables	Causes execution beyond the step to stop or continue, depending on one or more variable values.
Test Row	Tests the number of columns in a table row.
Test Window	Causes execution beyond the step to stop or continue, depending on whether a specific window exists.
Test Page Type	Causes execution beyond the step to stop or continue, depending on the type of the page.
Test Cell Type	Tests the cell type, such as Blank or Number of the found range and causes execution beyond the step to stop or continue depending on whether all cells in the range are of the given type.
Test JSON Type	Tests the type of a JSON value.

Excel

This category contains actions that are specially designed for Excel pages.

Action	Description
Extract Cell	Extracts content from an Excel page, runs it through a list of data converters, and stores the result into a variable.
Extract Sheet Name	Extracts the name of a sheet in a spreadsheet document and stores it in a variable.
Extract Hyperlink	Extracts a hyperlink from a cell in a spreadsheet.
Loop in Excel	Loops through different elements of a spreadsheet.
Extract As HTML	Extracts a part of a spreadsheet document as an HTML table and stores it in a variable.
Set Content of Cell	Inserts the specified content to a spreadsheet cell.
Set Value of Cell	Sets the value of a cell.
Set Content of Column	Sets the content of a column in a spreadsheet from a variable of complex type.
Set Content of Row	Sets the content of a row in a spreadsheet from a variable of complex type.
Set Format of Cells	Sets the format of one or more cells in a spreadsheet.
Set Sheet Name	Sets the sheet name.
Set Hyperlink on Cell	Inserts a hyperlink to a cell.
Set Column Width	Sets the width of a column in a spreadsheet.
Set Row Height	Sets the height of a row in points.
Set Information Property	Sets the value of an information property in a spreadsheet.
Insert Sheet	Inserts a new sheet in a spreadsheet.
Insert Rows	Inserts one or more rows in a spreadsheet.
Insert Columns	Inserts one or more columns in a spreadsheet.
Remove Sheet	Removes the selected sheet from a spreadsheet.

Action	Description
Remove Rows	Removes the selected rows from a spreadsheet.
Remove Columns	Removes the selected columns from a spreadsheet.
Test Cell Type	Tests the type of one or more cells.
Set Named Range	Marks the found range as a named range, so that it can be used as a reference when finding ranges in subsequent steps.
Set Evaluation Mode	Changes the auto evaluation option for an Excel value of an Excel variable, allowing to enter any unsupported function into a cell, not causing an error.

JSON

This category contains step actions for managing JSON values.

Action	Description
Extract JSON	Extracts the part of a JSON value found by the JSON finder as a JSON value into a variable.
Extract Property Name	Extracts the property name of a JSON value found by the JSON finder into a variable.
For Each Property	Loops through all properties of a JSON object.
For Each Item	Loops through a group of tags.
Set JSON	Replaces the entire found part of a JSON value with a new JSON value.
Set Property Name	Replaces the property name of the found JSON object with a new name.
Insert JSON	Inserts a new property into a JSON object or a new item into a JSON array.
Remove JSON	Removes the found JSON from a JSON value.
Test JSON Type	Tests the type of a JSON value.
Set Named JSON	Marks the found JSON as a named JSON.

XML

This category contains step actions related to XML.

Action	Description
Extract	Extracts text and stores it in a variable
Extract Tag Attribute	Extracts a tag attribute from the found tag, runs it through a list of data converters, and stores the result in a variable.
For Each Tag	Loops through a group of tags.
For Each Tag Path	Loops through all tags of a given type in the subtree of the found tag.
Set Tag	Replaces the entire found tag with new content.
Set Content	Sets the specified content on a tag.
Set Text	Replaces the content of the found tag with a text.

Action	Description
Set Tag Name	Replaces the name of the found tag with a new name and optionally copies the attributes of the found tag to the new tag.
Set Attribute	Inserts or updates an attribute on the found tag with a specified name and value.
Insert Content	Inserts the specified content into a document relative to the found tag.
Remove Tag	Removes the found tag from its parent node.
Remove Content	Removes all the content of a tag.
Remove Attribute	Removes an attribute from a tag in XML.
Test Tag	Makes a test to determine whether execution should be allowed to continue down the current branch.
Set Named Tag	Marks the found tag as a named tag, so that it can be used as a reference when finding tags in subsequent steps.

Other

This category contains various other step actions.

Action	Description
Set Named Tag	Marks the found tag as a named tag, so it can be used as a reference when finding tags in subsequent steps.
Set Named Range	Marks the found range as a named range, so that it can be used as a reference when finding ranges in subsequent steps.
Set Named JSON	Marks the found JSON as a named JSON.
Clear Named Tags/Ranges	Unmarks a selected named tag or range, or all named tags/ranges, so they are no longer be named in the subsequent steps.
Do Nothing	Does nothing.
Wait	Waits for a specified period of time.
Resume Browser	Resumes the browser and lets it run after either specified wait criteria are met or the browser becomes idle (whichever comes first).
Stop	Causes the execution of the robot to stop without errors.
Generate Error	Generates an error.
Execute Command Line	Executes a command line or shell script. Ensure a RoboServer has sufficient privileges for this operation
Change Proxy	Changes the proxy server.
Execute JavaScript	Executes JavaScript.
Lookup Password	Retrieves a user password from the Password store.

Assign Variable

This action assigns a value to a variable. In most cases the value is a simple type. If the source of the value is another variable, the value a complex type if the variable itself (such as test), rather than a field (such as test.result), is selected from the list of variables. In all cases, the type of the target variable must be compatible with the incoming value.

Properties

Configure the Assign Variable action using the following properties.

Value

The value to assign to a variable. Use the Value Selector to specify the value.

Variable

Specify a variable to assign the value to.

Branch Point

A Branch Point marks a point in a robot where execution is divided into several branches.

When robot execution reaches a Branch Point each branch is executed sequentially unless execution of a branch is terminated by an error. In that case execution continues from the point specified under the Error Handling tab in the step action view of the step where the error occurred.

The order in which the branches are executed is top down unless the outgoing connections are annotated with numbers, in which case the branches are executed in the order indicated by these.

Each branch is executed in the same state (page in the Page View, Cookies, and so on), except for global variables which will survive from one branch to the next. Any change to the outside world will also persist from one branch to the next. Such changes could be writing to a file, storing in a database, submitting a form on a web site, e.g. anything that has an effect in the real world (such as buying a book on Amazon).

Branch Points have no properties and will be inserted or deleted automatically depending on whether they are needed or not. If the End step from a branch is removed leaving only one branch, the branch point will be deleted automatically.

Calculate Key

This step provides the possibility to calculate the key for a variable value. The value must be of a complex type, i.e. of a type that is specified in a .type file and not one of the built-in simple types such as Number. At least one attribute in the type must be marked 'Part of Database Key'. Knowing the key of a value can be useful if it needs to be linked to another value (for instance as a secondary key in another table), or linked to data stored in a file.

Properties

Variable

Select the variable for which to calculate the key. Note to customers with legacy robots (older than version 7.2):The type of the variable must have a Standard type kind and not the specialized Database Output type kind which exists only for legacy purposes.

Key (output)

The variable in which the calculated key will be stored. The variable can be both a simple type variable and an attribute of a complex type variable.

Call Desktop Automation Robot

This action creates a Call Desktop Automation Robot step required to call a Desktop Automation robot from a Web Automation robot. For more information, see Get Started.

Before using the Desktop Automation feature, you need to configure Automation Devices and specify a reference to an Automation Device (not required when automating terminals).

Properties

Configure the Call Desktop Automation Robot step using the following properties.

Desktop Automation Robot
Specify a Desktop Automation robot to use.

Input Value
Provide an input value for the Desktop Automation robot.

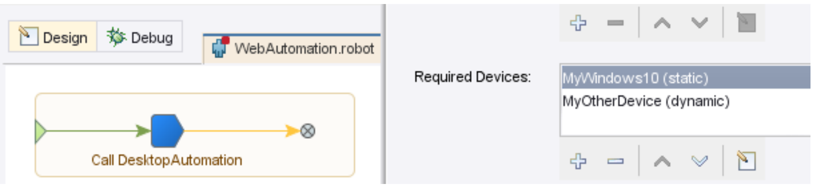
Output Mapping
Assign a variable to hold the output value from the Call Desktop Automation Robot step.

Required Devices
Specify the way to connect to an automation device for the Call Desktop Automation Robot step. You can select **Static Reference**, **Dynamic Reference**, or **Trigger Reference**. See Reference to Automation Device for more information.

If you select **Static Reference** or **Trigger Reference**, specify an Automation Device Mapping to use.

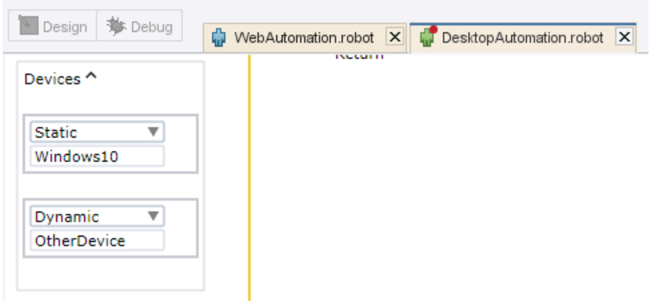
If you select **Dynamic Reference**, specify a mapping name to use in the Connect to Device step. Once the Dynamic Reference connection was used by the robot and device is connected, the connection stays alive and can be used by the next Call Desktop Automation Robot steps in your robot.

- The number of devices specified in the Web Automation robot and the Desktop Automation robot must match.



Required Devices:
MyWindows10 (static)
MyOtherDevice (dynamic)

- The devices names that you set in the Desktop Automation robot may differ from those in the Web Automation robot.



Devices
Static
Windows10
Dynamic
OtherDevice

Call REST Web Service

The Call REST Web Service action gets data from different sources on the Internet. In the step, requests made to the URI of the source return a response with a payload formatted in HTML, XML, JSON, or other format. The response is either presented in HTML form as the current page or stored in a variable.

If the web service returns a fault, the message is not returned by the action. Instead, the action will generate an error which can be handled using the standard error handling mechanisms.

The supported authentication for this action step includes Negotiate, NTML, Digest, Basic, and OAuth. You can configure the authentication method for this step on the **All Loading** tab by clicking **More** below **Options** in the step properties. For more information on the protocols, see Web Authentication. When setting the Negotiate protocol parameters for this step only (not for the entire robot), the spn.txt file cannot be used.

Properties

The Call REST Web Service action can be configured using the following properties:

URL

The base URL of the web service, excluding parameters. The URL can be specified in several ways using a URL Selector.

Request

Here, you specify the type of request to be made. REST supports five basic operations:

GET

Used for querying data. For GET requests, you can specify a number of parameters and/or files as name/value pairs to pass with the request. Click '+' to add a new parameter or upload a file.

POST

Used for updating selected parts of data. For POST requests, you can either specify a number of parameters as name/value pairs or give the entire body of the request. If you specify the request with parameters, you must select whether to use POST (application/x-www-form-urlencoded) or MULTIPART (multipart/form-data) to encode the parameters. If you give the entire ('raw') body of the request, you must specify the content type of the request data.

For POST and PUT requests, MULTIPART encoding can be selected to enable file uploads. If a binary variable is selected as the value of a File Upload parameter, the bytes are submitted as-is. If Base 64 encoding is desired, the value of the parameter should be an expression base64Encode(data) where data is the name of the variable containing the binary value. In that case, it is also recommended to specify the value base64 as Content Transfer Encoding - otherwise, this field can normally be left blank.

PUT

Used for replacing data. See POST for a description of the different ways of specifying a PUT request.

DELETE

Used for deleting data. For DELETE requests, you can specify a number of parameters and/or files as name/value pairs to pass with the request. Click '+' to add a new parameter or upload a file.

PATCH

Used for modifying data. For PATH requests, you can specify a number of parameters and/or files as name/value pairs to pass with the request. Click '+' to add a new parameter or upload a file.

Accept

The content types that will be accepted as response. By default, any type of response will be accepted. The accepted content types can be specified in several ways using the Value Selector.

Encoding

The encoding that will be used to encode special characters in the request. The encoding used for decoding the response, is controlled using the step option, on the Page Loading Tab.

Output

Here, you select what happens to the output of the web service call.

Load in browser

The result is loaded into the current window, just as if it had been the result of a Load Page action. You may configure the behavior of the browser using the Options property described below.

Store in variable

The result is stored in the selected variable.

Options

You can override the robot's options with the step's own options. An option that is marked with an asterisk in the Options dialog box will override the one from the robot's configuration. All other options will be the same as specified for the robot.

Call SOAP Web Service

The Call SOAP Web Service action sends a SOAP XML request to a web service and returns the web service's SOAP XML response. The response is either presented in XML (or HTML) form as the current page or set as the value of an XML variable.

As SOAP requests can be quite complex, you will typically use an external tool to generate the request and paste it into the SOAP XML Request property. The request can be dynamically altered by specifying XML from Expression to create a template SOAP request substituting literal values by expressions.

If the web service returns a SOAP fault, the message is not returned by the action. Instead, the action will generate an error which can be handled using the standard error handling mechanisms.

Properties

The Call SOAP Web Service action can be configured using the following properties:

Web Service URL

The location of the web service operation is specified here. Web services generally use the HTTP protocol. The value can be specified in several ways using the Value Selector.

SOAP Action

This property can contain an optional SOAP action. The value can be specified in several ways using the Value Selector. The SOAP action is sent as part of the HTTP headers. The SOAP action is typically, but not always, a URL specifying the requested action

SOAP Request

This property must contain a valid SOAP XML request. By default the XML can be specified literally. To dynamically create the SOAP XML request you can choose XML from Expression or XML from Variable.

SOAP Version

This property specifies which version of the SOAP specification to use for sending the SOAP request. SOAP 1.1 and SOAP 1.2 are supported. When specifying SOAP 1.1 the Content-Type will be set to text/xml and the (optional) SOAP Action will be set using an additional HTTP header. When specifying SOAP 1.2 the Content-Type will be application/soap+xml and the (optional) SOAP Action will be set as the action parameter of the Content-Type HTTP header.

Output

Choose whether to output the SOAP response as an XML page or as the value of an XML Variable. In robots created with Kofax RPA 7.2 or earlier, this may be Output Result as HTML Page, meaning that the XML will be transformed to an HTML representation.

Options

The robot's options can be overridden with the step's own options. An option that is marked with an asterisk in the Options Dialog will override the one from the robot's configuration. All other options will be the same as specified for the robot.

Change Proxy

This action changes the proxy server used in subsequent steps. Hereby, it allows the robot to switch between multiple proxy servers during its execution or use a specified proxy.

To use this action with auto select, you must specify a list of proxy servers that the action should switch between.

At each execution of the Change Proxy action a new proxy server is selected from the list. The proxy servers are selected in the order that they appear on the list, with the first one being chosen randomly at the first execution of the Change Proxy action.

See Use Proxy Services for more details.

Properties

The Change Proxy action can be configured using the following properties:

Remove Cookies

Specifies whether to remove all cookies when the proxy is changed. If the proxy is used to stay anonymous then cookies should be removed when changing proxy.

Auto Select

If checked, the step will select the next (round robin) proxy from the properties file. It will test that it is possible to connect to the proxy before selecting it. If Auto Select is not checked it will use the proxy specified manually in the properties explained below

Host Name

Specify the host name of the proxy.

Port Number

Specify the port number of the proxy.

User Name

Specify the user name to use for authenticating against the proxy.

Password

Specify the password to use for authenticating against the proxy.

Excluded hosts

Specify a list containing hosts that should be excluded from the proxy. You can specify either one host name per line or a host name pattern per line using wildcards (*).

Clear Named Tags/Ranges

This action unmarks a selected named tag or range, or all named tags and ranges, so that these will no longer be named in the subsequent steps.

Properties

The Clear Named Tags/Ranges action can be configured using the following properties:

Named Tags/Ranges to Clear

Specify which named tags or ranges to unmark, either a single one selected by name or all named tags and ranges in the current window.

Clear Web Storage

The Clear Web Storage step action clears data in the local and/or session storage. The local and session storages are used by some websites to persist larger amounts of data that can normally be stored in a cookie.

Related Step Actions

The Load Web Storage step action can be used to load new data into the local and/or session storage, or replace existing data. It does not remove the existing data, unless it overwrites it with a new value.

The Extract Web Storage step action is used to extract data from the local and/or session storage.

Properties

Clear Local Storage

When checked, the storage items from the local storage will be cleared. In a browser, the local storage is normally persisted across browser sessions, similarly to persistent cookies.

Clear Session Storage

When checked, the storage items from the session storage will be cleared. In a browser, the session storage is normally persisted for as long as the browser window or tab exists, similarly to a session cookie.

Key Pattern

If you wish to clear only the stored items with a particular key, you can specify a pattern that matches the key of interest. If you leave the field blank, all storage items will be cleared regardless of their keys. If you do specify a pattern, note that it must match the entire key.

Domain Pattern

If you wish to clear only the stored items that belong to a particular domain, you can specify a pattern that matches the domain of interest. If you leave the field blank, storage items of all domains will be cleared. If you do specify a pattern, note that it must match the entire domain.

Click

This action emulates a mouse click on the found tag.

This is the most common action to use for navigation such as following links, submitting forms, etc. It can be used when the robot should perform the same action as the browser when clicking on something. Depending on the found tag, the Click action will perform the necessary page loading, form submissions, JavaScript execution, and so on.

To emulate a mouse movement instead of a click, use the Move Mouse To and Move Mouse From actions.

Properties

Double-Click

Specify whether to emulate a double-click or a single click.

Right-Click

Specify whether to emulate a right-click or a left click.

Coordinates

When set to Automatic, the browser will select relevant coordinates to click. Alternatively, you can specify exactly which coordinates to click. These are specified in pixels relative to the upper left of the component (such as an image or button) clicked.

Continue when

Add a wait criterion for the step. If you have several wait criteria, execution stops when any wait criterion is met. You can have several met wait criteria such as if you are waiting for two HTML elements that appear on the same load, or if you are waiting for an element on the main frame, and *Initial Page Load Completes* is set. For more information, see Use Wait Criteria.

Options

The robot's options can be overridden with the step's own options. An option that is marked with an asterisk in the Options Dialog will override the one from the robot's configuration. All other options will be the same as specified for the robot.

Close Window

This action closes a window.

In Design Studio inserting a Close Window step is easily done by right-clicking on the window's tab and then choosing Close Window.

Properties

The Close Window action can be configured using the following properties:

Window to Close

Specify the window that should be closed (see the discussion on how to identify a window).

Continue when

Add a wait criterion for the step. If you have several wait criteria, execution stops when any wait criterion is met. You can have several met wait criteria such as if you are waiting for two HTML elements that appear on the same load, or if you are waiting for an element on the main frame, and *Initial Page Load Completes* is set. For more information, see Use Wait Criteria.

Options

The robot's options can be overridden with the step's own options. An option that is marked with an asterisk in the Options Dialog will override the one from the robot's configuration. All other options will be the same as specified for the robot.

Convert Variables

This action converts the values of one or more variables by running them through data converters and storing the results in the same or other variables. The Convert Variables action is used to convert values extracted from a particular web site to the values required in the variable(s) to return. It is also used for converting values in an input variable to the values used on a particular web site.

For every variable that should be converted, a conversion to the list of conversions should be added. Each conversion takes the value of a selected variable, passes it through the selected data converters, and writes the result in another (or the same) selected variable.

The two selected variables can be the same variable, or different variables, depending on where the converted variable value should be stored. A variable value can simply be copied from one variable to another by not selecting any data converters in the conversion.

Properties

The Convert Variables can be configured using the following properties:

Conversions

Specify the list of conversions to use.

Conversion Properties

The Convert Variables can be configured using the following properties:

From

Specify the variable holding the value to convert.

Conversions

Specify the list of data converters to apply to the variable value. This list may be empty, e.g. if the value of one variable should merely be copied to another variable.

To

Specify the variable in which to store the result of the conversion. This may be the same variable as specified in the "From" property or a different variable.

Copy File

This action copies a file on the local file system where the robot is executed.

Note that the action is only performed during execution in Design mode in Design Studio, if the option Execute in Design Mode has been selected.

Properties

The Copy File action can be configured using the following properties:

Source File

This is the file system path or a file URL of the source file to be copied. This can be specified in several ways using the Value Selector. The path must be absolute, including the drive name, if any, and the directory path to the directory. Alternatively, it can be a file URL, e.g. file:/C:/temp/myFile, in which case it must be URL encoded. The separators / and \ may be used interchangeably.

Destination File

This is the file system path or a file URL of the destination file. This can be specified in several ways using the Value Selector. The path must be absolute, including the drive name, if any, and the directory path to the directory. Alternatively, it can be a file URL, e.g. file:/C:/temp/myFile, in which case it must be URL encoded. The separators / and \ may be used interchangeably.

Execute in Design Mode

If this is enabled, the action will be executed in Design Mode inside Design Studio. If this is disabled, the action will do nothing when navigating the robot in Design Mode.

Create Cookie

The Cookie Creator creates a cookie with the specified domain, path, name and (optionally) value and adds it to the set of current cookies. If a cookie with the specified domain, path and name already exists, the old cookie is replaced by the new cookie.

Properties

The Create Cookie action can be configured using the following properties:

Domain

Specify the domain of the cookie. The domain can be specified in several ways using the Value Selector.

Path

Specify the path of the cookie. The path can be specified in several ways using the Value Selector.

Name

Specify the name of the cookie. The name can be specified in several ways using the Value Selector.

Value

Specify the value of the cookie. The value can be specified in several ways using the Value Selector. This property is optional.

Secure

If checked the cookie is sent when loading from the given domain via HTTPs, if not set the cookie is set when loading from the domain via HTTP.

HTTP Only

If checked the cookie is a HTTP Only cookie. That is, the cookie will only be used when transmitting HTTP (or HTTPS) requests, but its value will not be available to client side script (such as JavaScript).

Create Page

The Create Page action creates a new page which replaces the old page in the current window. The page is processed similarly to what takes place in the Load Page step action. This also entails that any JavaScript present in the HTML of the new page will be executed, unless JavaScript execution is disabled in the options. The Create Page action may also be used to load non-HTML pages, e.g. XML documents.

Properties

The Create Page action can be configured using the following properties:

Contents

The content of the new page can be specified in several ways using the Value Selector. It is for instance possible to acquire the contents from a variable, be it a variable with text or even binary content. The type of the content (e.g. HTML) will be detected automatically. If the automatic detection is insufficient or if the content should be loaded differently (e.g. to load an HTML document as plain text), you can override the content type detection in the options.

Page URL

Here, you specify the page URL of the new page. This is, among other things, used to resolve any relative links or resource references in the page. This can be specified in several ways using the Value Selector.

Continue when

Add a wait criterion for the step. If you have several wait criteria, execution stops when any wait criterion is met. You can have several met wait criteria such as if you are waiting for two HTML elements that appear on the same load, or if you are waiting for an element on the main frame, and *Initial Page Load Completes* is set. For more information, see Use Wait Criteria.

Options

The robot's options should be overridden with the step's own options. An option that is marked with an asterisk in the Options Dialog will override the one from the robot's configuration. All other options will be the same as specified for the robot.

Delete File

This action deletes a file or a directory on the local file system where the robot is executed.

Note that the action is only performed during execution in Design mode in Design Studio, if the option Execute in Design Mode has been selected.

Properties

The Delete File action can be configured using the following properties:

File or Directory

This is the file system path or a file URL for the file or directory to be deleted. This can be specified in several ways using a Value Selector. The path must be absolute, including the drive name, if any, and the directory path to the directory. Alternative it can be a file URL, e.g. file:/C:/temp/newDir, in which case it must be URL encoded. The separators / and \ may be used interchangeably.

Delete Non Empty Directories

This is an option that is only relevant if deleting a directory. If not selected the action will only delete empty directories. If selected the action will delete the directory including all files and subdirectories and it will do so recursively for all the subdirectories.

Execute in Design Mode

If this is enabled, the action will be executed even in Design Mode inside Design Studio. If this is disabled, the action will do nothing when you navigate the robot in Design Mode.

Delete from Database

This step helps you delete a value previously stored in a database. The database connection must be configured in Settings.

Properties

Database

Specify the database to delete from. Select or hard code a value at design time, or dynamically construct the database name at runtime using a variable, an expression or converters - an error will occur if no database with this name exists when the robot is executed.

Variable

Select the complex type variable containing the value to delete.

Key

Specify the unique key for the value to delete. The key may be defined in the variable (by marking attributes as "Part of Database Key" in the variable type), or can be defined using the Value Selector (excluding the value-parameter).

Execute in Design Mode

If this is enabled, the step will be executed even in Design Mode inside Design Studio. If this is disabled, the step will do nothing when you navigate the robot in Design Mode.

Desktop Automation

This step is deprecated and you should use the Call Desktop Automation Robot step instead.

Convert old Desktop Automation action step

In Kofax RPA versions 10.7 and earlier, a standalone Desktop Automation Editor was used to edit the Desktop Automation workflow contained in a Desktop Automation action step. Starting with Kofax RPA 11.0, you can execute Desktop Automation action steps created in version 10.7 or earlier, but to edit the workflow, you need to extract the Desktop Automation workflow from the step into a new Desktop Automation robot and change the Desktop Automation step to a Call Desktop Automation Robot step referring to the new robot.

The export can be performed with both Web Automation robots and snippets containing Desktop Automation steps.

> **Note** The export process cannot be undone, so you cannot revert to the initial Desktop Automation step once you exported it to a Desktop Automation robot.

1. To convert the action step to a robot, in the Projects tree, right-click a Web Automation robot containing a Desktop Automation step and click **Export Desktop Automation Robots**. If you open a single robot and select the Desktop Automation step, you can preview the future Desktop Automation robot by clicking **Preview** in the step properties.
    - For convenience when working with multiple robots, you can export several Desktop Automation robots at once. In the Projects view, right-click any folder or select and right-click multiple

Web Automation robots containing Desktop Automation steps and then click **Export Desktop Automation Robots**.

A new dialog box appears listing all found Desktop Automation steps to extract from.

2. If required, you can assign a new descriptive name to the future Desktop Automation robot, preview the future robot, and see the Desktop Automation step in the containing Web Automation robot.
   - Click to rename the selected file.
   - Click to preview the selected Desktop Automation robot to be created after the export.
   - Click to show the selected Desktop Automation step in the containing Web Automation robot.
3. Click **Next** to select a location for the exported robots inside the current project.
4. Click **Finish** to close the dialog box and start the export.

   An export summary appears listing how many robots and snippets have been exported. Any required devices configured in the Web Automation robots automatically appear in the Desktop Automation robots.

Divide Text

The Divide Text action divides the text contained in the found tag into pieces using a pattern and an expression. It is useful when looping over these pieces in a subsequent step.

Properties

The Divide Text action is configured using the following properties:

Pattern

Specify a pattern that will be matched against the text in the found tag. For every match of the pattern, the expression in the Output Expression field will be evaluated. The found tag will then be replaced by a <span>-tag, which, for each match of the pattern, contains the result of the expression surrounded by another <span>-tag.

Ignore Case

If this property is checked, then the pattern matching will be case insensitive.

Output Expression

This field contains an expression that is evaluated for every match of the pattern.

Example

If the found tag is the "Kofax RPA" text on this page:

```
<html>
  <body>
    <p>
      Kofax RPA
    </p>
  </body>
</html>
```

and the Pattern is set to "\S+\s?" (meaning at least one non-whitespace character, followed by an optional whitespace), and the Output Expression is set to "$0" (meaning the entire matched text), then the output will be:

```
<html>
  <body>
    <p>
      <span>
        <span>Kofax</span>
        <span>RPA</span>
      </span>
    </p>
  </body>
</html>
```

Do Nothing

The Do Nothing action does nothing.

This action is useful when adding comments to a robot.

Properties

The action has no properties.

End Step

The End Step marks the end of a branch in a robot.

This step is a useful marker to have at the end of a branch, as clicking on it enables all steps in the branch to execute. Without it another step would have to be inserted at the end in order to be able to execute the last step of the branch.

The End Step is not associated with any action and is comparable to an action step with a Do Nothing action when it comes to its execution. An End Step cannot be deleted, but several branches may share the same End Step.

Note The End Step does not stop execution of a robot. For this you should use an action step with a Stop action.

Enter Password

This action enters a password in a password field in a form.

This action is similar to the Enter Text action, except that when a fixed password is specified, the password is not visible to the user in Design Studio and in saved robot files.

The found tag must be a password field.

Note that entering the password may trigger execution of JavaScript if there are any registered event handlers on the password field.

Properties

The Enter Password action can be configured using the following properties:

Password to Enter

Specify the password to enter in the password field. The value can be specified in several ways using the Value Selector.

Obtain focus by

Indicates how you want the robot to set focus to the selected tag.

Select all text before typing

Sometimes it might be necessary to select the existing text in the field to ensure that it is overwritten when entering new text.

Use this option to select the existing text in the password field before entering a new password.

Continue when

Add a wait criterion for the step. If you have several wait criteria, execution stops when any wait criterion is met. You can have several met wait criteria such as if you are waiting for two HTML elements that appear on the same load, or if you are waiting for an element on the main frame, and *Initial Page Load Completes* is set. For more information, see Use Wait Criteria.

Options

The robot's options can be overridden with the step's own options. An option that is marked with an asterisk in the Options Dialog will override the one from the robot's configuration. All other options will be the same as specified for the robot.

Enter Text

This action enters a text in a found tag, which must be either a text field, a text area, a password field, a file field, a hidden field, or a tag whose content is editable.

To enter a fixed password in a password field, consider using the Enter Password action instead in order to prevent the password from being visible to the user in Design Studio and in saved robot files.

Note that entering the text may trigger execution of JavaScript if there are any registered event handlers on the field or tag.

Properties

The Enter Text action can be configured using the following properties:

Text to Enter

Specify the text to enter. The value can be specified in several ways using the Value Selector.

Obtain focus by

On some websites it is necessary to set focus on the input field before entering text. Use the **Obtain focus by** option to set focus on the input field.

Select all text before typing

Sometimes it might be necessary to select the existing text in a field to ensure that it is overwritten when entering new text. Use the **Select all text before typing** option to select the existing text in a field.

Continue when

Add a wait criterion for the step. If you have several wait criteria, execution stops when any wait criterion is met. You can have several met wait criteria such as if you are waiting for two HTML elements that appear on the same load, or if you are waiting for an element on the main frame, and *Initial Page Load Completes* is set. For more information, see Use Wait Criteria.

Options

The robot's options can be overridden with the step's own options. An option that is marked with an asterisk in the Options Dialog will override the one from the robot's configuration. All other options will be the same as specified for the robot.

Execute Command Line

This action executes a command line (external program)

Properties

The Execute Command Line can be configured with the following options

Command Line

The command line to execute. The value is configured using a Value Selector On Windows the command line is used as an argument to "cmd.exe /C". On other platforms the command line is used as an argument to "/bin/sh -c"

Extract

f the program writes text to the console, the extraction of the text is configured here, the options are

- "Nothing" extracts nothing
- "Stdout" extracts the text written to stdout into a variable
- "Stderr" extracts the text written to stderr into a variable
- "Separate stdout and stderr" extracts the text written to stdout and stderr into two separate variables
- "Joined stdout and stderr" extracts the text written to stdout into a single variable

If the program writes non-ASCII characters to the console, you can specify the encoding used to read the text. On western European Windows versions the console will most likely uses cp858 also known as "Latin-1, MS-dos, with Euro". Other platforms will most likely have to use utf-8 for encoding to read console text, but this is environment specific.

Store Exit Code Here

Specifies the variable the exit code will be stored in. When a program is done executing it returns an exit code, specifying the status of the execution. 0 means success, other values indicate some kind of error, but the meaning of the error is program specific (although there is some consensus in the area, fx. a value of 2 usually means File not found). If no variable is specified, the exit code will be discarded.

Execute in Design Mode

If this is enabled, the action will be executed even in Design Mode inside Design Studio. If this is disabled, the action will do nothing when you navigate the robot in Design Mode.

Note The program inherits the working directory and environments from Design Studio

Note The step has no timeout, and will wait until the external program completes

Execute JavaScript

This action executes JavaScript on the current page, or your own custom JavaScript.

Note that most step actions will automatically execute relevant JavaScript as part of their operation, so you generally do not need to use the Execute JavaScript action unless you have special needs for executing JavaScript.

The Execute JavaScript action supports the following ways in which HTML can contain JavaScript:

- <script>-tags, which can contain multiple lines of JavaScript to be executed, or can refer to external JavaScript files.
- event handlers, which may appear as special attributes on tags, and they always begin with "on," such as, onClick or onMouseOver. They may also be attached from JavaScript without being visible in the HTML source.
- JavaScript URLs, which specify the JavaScript: Protocol with values for tag attributes where there can be a URL, such as `<a href="javascript:open()">`

Properties

The Execute JavaScript action can be configured using the following properties.

JavaScript

This property specifies which JavaScript you want to execute:

- **JavaScript in All <script> Tags** executes all of the <script>-tags in the current page.
- **JavaScript in Selected <script> Tag** executes a single found <script>-tag.
- **JavaScript in URL** executes the JavaScript in a JavaScript URL, as specified by the JavaScript: Protocol.
- **JavaScript in Event Handler** executes the JavaScript in an event handler in a tag. The specific event handler that should be executed must be chosen from the drop-down box.
- **Custom JavaScript** executes your own custom JavaScript.
- **Custom JavaScript from Expression** executes your own custom JavaScript. This is similar to the Custom JavaScript option, except that an expression can be entered instead of a fixed text. For a list of JavaScript functions check Convert Using JavaScript

Continue when

Add a wait criterion for the step. If you have several wait criteria, execution stops when any wait criterion is met. You can have several met wait criteria such as if you are waiting for two HTML elements that appear on the same load, or if you are waiting for an element on the main frame, and *Initial Page Load Completes* is set. For more information, see Use Wait Criteria.

Options

The robot's options can be overridden with the step's own options. An option that is marked with an asterisk in the Options Dialog will override the one from the robot's configuration. All other options will be the same as specified for the robot.

Execute SQL

The Execute SQL action submits an SQL statement to a database and optionally stores the number of affected rows in a variable. Any other results returned by the database are ignored. In other words, this action is intended for SQL insert, update, and delete statements, but can also be used to invoke stored procedures and the like. The SQL is specified using an expression.

Note that the SQL is not executed during execution in Design mode in Design Studio.

> **Important** Do not execute `use <Some Other DB>` command in the SQL statements, because this will change the results of all future SQL commands (in the same robot or others).

Properties

The Execute SQL action can be configured using the following properties:

Database

Choose which database this action should submit its query to by using the drop-down list of databases available to Design Studio.

SQL

This field must contain a valid SQL statement in the form of an expression. The value of this expression is submitted to the chosen database.

Modified Rows

Choose a text or integer variable in which to store the number of rows affected by the SQL statement. This is optional.

Execute in Design Mode

If this is enabled, the step will be executed even in Design Mode inside Design Studio. If this is disabled, the step will do nothing when navigating the robot in Design Mode.

Extract

The Extract action extracts text and stores it in a variable.

There is possibilities for specifying what content should be extracted such as only the text, or everything including the tags. Before the text is stored, it can be processed using a list of data converters, and optionally trimmed for leading and trailing spaces.

The simplest way to use the Extract action is to extract from a single found tag. It is also possible to extract from a *tag range*, i.e. all tags from one found tag to another found tag.

Properties

The Extract action can be configured using the following properties:

Extract From

Specifies the part of the found tag that will be extracted.

- **Found Tag** specifies that the entire found tag should be extracted.
- **Tag Range** specifies that a range of tags should be extracted. Begin and end tags and whether or not to include these tags in the range can be selected.

Extract This

Specifies what content should be extracted.

- **Only Text** specifies that only the text should be extracted.
- **Structured Text** specifies that only the text should be extracted, but that it should be structured similarly to how it would appear in a browser. The system can guess at the location of a heading, and insert text before and/or after. You can set the following options.

  Include Aligned Tables and Images

  Specifies that the tables and images that are aligned to the left or right of the text are included in the output text. Disabling this can sometimes result in removing the desired content.

  Include URLs

  Specifies that the actual URLs in link tags will be included in the output text.

  Include Image Text Alternatives

  Specifies that the text representation of images will be included in the output text.

  Include Form Fields

  Specifies that the text representation of form fields will be included in the output text.

  Insert This Before a Heading

  Specifies that this action should guess at the location of headings and insert the specified text before them.

  Insert This After a Heading

  Specifies that this action should guess at the location of headings and insert the specified text after them.

- **Advanced Structured Text** specifies that only the text should be extracted, but that it should be structured similarly to how it would appear in a browser. Tag names can be converted into any text. You can set the following options.

  Include Aligned Tables and Images

  Specifies that the tables and images that are aligned to the left or right of the text are included in the output text. Disabling this can sometimes result in removing the desired content.

  Include URLs

  Specifies that the actual URLs in link tags will be included in the output text.

  Include Image Text Alternatives

  Specifies that the text representation of images will be included in the output text.

  Include Form Fields

  Specifies that the text representation of form fields will be included in the output text.

  Tag Conversions

  Specifies the tag conversions to use. A tag conversion is on the form tag=text. For instance "<h1>=<head1>" and "</h1>=</head1>" would convert HTML headings level 1 to special <head1>-

tags. Please note that the right sides of the conversions can be anything, they need not be ordinary tags.

- **HTML** specifies that the whole HTML should be extracted.

  Format HTML

  Specifies that the HTML should be pretty-printed.

  Encode URLs

  Specifies that URLs in attribute values should be HTML encoded. This is highly recommended, as it is necessary to generate standard compliant HTML that will work consistently across different browsers. In some cases when the HTML is to be subjected to simple processing for recognizing and comparing URLs it may, however, be necessary to leave the URLs unencoded.

  Extract Relative URLs

  Specifies that all URLs should be extracted as relative. Thus, if present, the base part of the URL is removed.

- **XML** specifies that the whole XML should be extracted. This only works if the page is an XML page.

  Include XML Declaration

  Specifies that the XML Declaration (e.g. <?xml version="1.0" encoding="UTF-8"?>) should, if present, be included in the extracted XML. This mean that one may extract part of an XML document an get a new XML document with a proper declaration at the top.

Converters

An optional list of data converters that should process the text.

Trim Spaces

If selected, spaces at the start and end of the text will be removed before storing the text in the variable.

Variable

Specifies the variable in which to store the extracted text.

Extract As HTML

The Extract As HTML action extracts a part of a spreadsheet document as an HTML table and stores it in a variable. The range finder of the step determines what is extracted and this may be either a part of a single sheet or all the information properties.

Properties

The Extract Cell action can be configured using the following properties:

Include Headers

This property specifies whether or not to include the spreadsheet headers (1,2,3,4 and A,B,C,D, etc.) in the generated table.

Extract This

This specifies what is extracted from the cells, but have no effect for extraction of information properties. There are three possible choices:

- "Formatted Values" specifies that the formatted should be extracted, e.g. numbers will be extracted as the view shows these and not the internal representation with extra decimal point.
- "Plain Values" specifies that the internal representation is extracted, e.g. numbers are extracted with full precision and dates are extracted as number of days since January 1, 1900. Cells for which there are no difference between formatted and plain values, e.g. cells containing text or logical values, the extracted values are the same as for "Formatted Values".
- "Formulas" specifies that the formulas should be extracted. For cells that do not contain a formula the extracted values are the same as for "Plain Values".

Variable

The variable to store the generated HTML table in.

Extract Binary Content

This action extracts binary content from the Browser View. If a step has loaded binary content into the Browser View this data will not be shown, but the data may be extracted into a binary variable using Extract Binary Content in the next step.

Properties

The Extract Binary Content action can be configured using the following properties:

Store Data In

The variable in which to store the binary content. This must be a binary variable.

Content Type

The variable in which to store the content type of the data.

File Name

The variable in which to store the original file name from the location where the data was loaded from, e.g. the file name from the URL. This name may for instance be needed if saving to a file happens in a subsequent step in the robot.

Extract Cell

The Extract Cell action extracts content from a spreadsheet document, runs it through a list of data converters, and stores the result into a variable.

Properties

The Extract Cell action can be configured using the following properties:

Extract This

This specifies what is extracted from the cell. There are three possible choices:

- "Formatted Values" specifies that the formatted should be extracted, e.g. numbers will be extracted as the view shows these and not the internal representation with extra decimal point.
- "Plain Values" specifies that the internal representation is extracted, e.g. numbers are extracted with full precision and dates are extracted as number of days since January 1, 1900. Cells for which there are no difference between formatted and plain values, e.g. cells containing text or logical values, the extracted values are the same as for "Formatted Values".
- "Formulas" specifies that the formulas should be extracted. For cells that do not contain a formula the extracted values are the same as for "Plain Values".

Converters

An optional list of data converters that should process the content.

Variable

The variable to assign the value to.

Extract Column in Data Row

This action extracts data from a cell in the current range to a variable. The range is a row in a CSV file produced by the For Each Data Row step. To use the Extract Column in Data Row step, first create and configure the For Each Data Row step.

Properties

Finders Tab

- Context: The name of the range of cells from the For Each Data Row step.
- Column: Name of the column or its index (starts from 1).

Action Tab

- Converters: An optional list of data converters that should process the content.
- Variable: The variable to assign the value to.

Extract Cookie

The Extract Cookie action extracts the value of a cookie from the set of current cookies. The cookie is selected using regular expressions for domain, path, and name. If more than one cookie matches the patterns, the value is extracted from the first matching cookie.

Properties

The Extract Cookie action can be configured using the following properties:

Domain Pattern

Specify a pattern that matches the domain of the cookie. The pattern must match the entire domain of the cookie.

Path Pattern

Specify a pattern that matches the path of the cookie. The pattern must match the entire path of the cookie.

Name Pattern

Specify a pattern that matches the name of the cookie. The pattern must match the entire name of the cookie.

Variable

Here you specify the variable in which to store the extracted value.

Extract Form Parameter

The Extract Form Parameter action extracts a form parameter from a form URL in the found tag and stores the value in a variable.

Properties

The Extract Form Parameter action can be configured using the following properties:

Form Parameter Name

Specifies the name of the form parameter.

Converters

Data converters to apply to the extracted value before storing it in the variable.

Variable

The variable in which to store the resulting value.

Encoding Used in URL

Specifies the character encoding used in the URL. If the extracted value contains incorrect characters, change this encoding to the encoding used in the page that contains the URL, or the page that contains the form from which the URL was created. The most commonly used encodings in form URLs are Unicode (UTF-8) and Latin-1 (ISO-8859-1).

Example

Consider this found tag:

```
<a href="http://www.abc.com/search?author=Johnson">
  Search
</a>
```

If Form Parameter Name is set to "author", the value "Johnson" will be extracted and stored in the selected variable.

Extract from E-mail

This action extracts information from a saved email. As an input this step takes either a string or a text variable containing an email message saved in MIME format in an `.eml` file. If the message file contains both HTML and plain text versions, the step creates two frames with HTML and plain text representation of the email. The extracted email is opened in the browser and you can extract information from it just as from any HTML or plain text file. Any attachment is added in the anchor tag and you can add it to a variable or open it for editing if the attachment type is supported by Kofax RPA. For example, you can edit attached PDF, Excel, or HTML documents. You can also open images and extract information from them in several ways, such as using the Extract text from image step.

Note The step currently does not support extracting from the Outlook `.msg` format.

Properties

The "Extract from E-mail" action can be configured using the following properties:

Variable containing e-mail
Select a string or a text variable that contains an email in MIME format.

Include all headers
By default the step includes only basic header information, such as From, To, Subject, Date, and Message ID. If you select this option, the entire message header is included in the extracted email.

Options

- **Images to Load**

  You can filter images that are loaded with the email. Select from **None**, **All**, **Depends on URL**. If you select **Depends on URL**, click the plus sign and specify a condition to filter the URLs.
- **More**

  Click **More** to edit the step action default options.

Extract Flash Content

This action extracts content from a Flash object in a found tag.

Static text, HTML fragments and images will be extracted from the Flash object and presented as an HTML page. Contained URLs, references to other Flash objects and configuration files will be converted to links in a list at the bottom of the page.

Properties

The Extract Flash Content action can be configured using the following properties:

Include images

Specifies whether images should be extracted from the Flash object or not.

Options

The robot's options can be overridden with the step's own options. An option that is marked with an asterisk in the Options Dialog will override the one from the robot's configuration. All other options will be the same as specified for the robot.

Extract from PDF

This action extracts text and images from a PDF document contained as binary data in a selected binary variable.

Typically, the PDF document has been downloaded into the variable using an Extract Target step. The output from the "Extract from PDF" action is an HTML page containing the text and images extracted from the PDF document. In subsequent steps, the desired information can then be extracted from the page, in the same way as for other HTML pages.

Note the following:

- PDF documents do not contain structure information such as tables or paragraphs, only positions of texts and graphics, that might or might not be positioned to look like tables or paragraphs. This can make it difficult to extract the desired information from PDF documents. However, the Extract from PDF step will apply some heuristics to group the text into HTML paragraphs based on the available position information.
- The Extract from PDF step cannot extract data entered in forms. To make the form data available for extraction, you need to flatten the document using a third-party tool.

Properties

The "Extract Text from PDF" action can be configured using the following properties:

PDF Variable

The binary variable containing the PDF document as binary data.

Include Images

Specifies whether embedded images should be extracted. Note that not all images and graphics can be extracted from PDF documents; it depends on the way they have originally been embedded in the document.

Include Form XObjects

This option enables extraction of the *Form XObjects* from the PDF. *Form XObjects* groups objects within a PDF file. The objects may include text, images, vector elements, and etc. *Form XObjects* is usually used to store objects that are referenced multiple times within a document.

Include Positioning

Specifies whether the positions of the texts should be extracted. The positions may be useful to derive the structure of the document.

Include Formatting

Specifies whether the formatting (font names, sizes etc.) of the texts should be extracted. Like the positions, the formatting may be useful to derive the structure of the document.

Merge Text

As default the converter that generated the HTML from the PDF will merge text that is on the same line into one HTML element even if these are represented as different text in the PDF document. Though this may often desirable, it may in some cases have the effect that text that originally far apart will be merges together and appear to be right next to each other. A typical case where it would be desirable to turn this feature off is if the document contains more than one column. Turning the feature off will attempt to preserve the column structure.

Extract Hyperlink

This action extracts a hyperlink from a cell in a spreadsheet.

Properties

The Extract Hyperlink action can be configured using the following properties:

Converters
An optional list of data converters that should process the content.

Variable
The variable to assign the value to.

Extract Image

This action extracts an image from the found tag and stores it in a variable or to a file.

The action can also optionally store the actual content type and file name of the extracted image in other variables.

Properties

The Extract Image action can be configured using the following properties.

Store In

Specifies where to store the extracted image. There are two choices for this:

Variable

Specifies the variable in which to store the extracted data. The variable must be of type Image or Binary. Using the specialized Image variable is recommended as it will be possible to see a preview of the extracted image in the Variables view.

File

Specifies the file to write the data to

File Name

Specifies the name and extension of the file.

Auto

With this option a file name is generated automatically using the following strategy:

1. First the content disposition header of the response is inspected to see if it has a filename parameter and if so that name will be used.
2. Next the URL is inspected to see if it contains a file name and if so that name will be used.
3. If none of the above options succeed then an error is generated.

Value, Variable, Expression and Converters

The value can be specified in several ways using the Value Selector.

Directory

Specifies the directory where the file will be placed. The value can be specified in several ways using the Value Selector.

Create Directories

Specifies whether to create all the directories, in the specified path, that does not exist. If the option is selected the directories are created. If the option is not selected the directories must exist and if not an error is generated.

Override Strategy

This specifies a strategy for what to do when the selected file already exists.

Override File

Any existing file will be replaced.

Never Override File

Ensures that an existing file will never be replaced. If the file exists then an error is generated.

Create a New File

This option ensures that a new file will always be created. If a file with the selected name already exists then a new unique file name will be generated for the new file. This new file name will be the originally selected file name with a serial number added to the end just before the extension, e.g. myImage_1.png where _1 was added to the original file name myImage.png.

Store Meta Data In

Specifies variables to be used to store meta data about the extracted image.

Content Type

Specifies an optional variable in which to store the content type of the image. For example, the content type could look like this: image/gif

File Name

Specifies the optional variable in which to store the file name of the extracted image. If the image is saved to a file then the file name will be the full path of the file actually used. If the image is loaded into a variable then the file name will be the file name of the original resource (obtained from the URL or from the content disposition header for the response).

Options

The robot's options can be overridden with the step's own options. An option that is marked with an asterisk in the Options Dialog will override the one from the robot's configuration. All other options will be the same as specified for the robot.

Extract JSON

The Extract JSON step action extracts the part of a JSON value found by the JSON finder as a JSON value into a variable. The extracted value must itself be valid JSON, e.g. an object, array or a basic value. In other word you cannot extract a name/value pair"answer" : 42 even if you can select it in the view. Selecting such a name/value pair and extracting from it will result in only the value (42) being extracted.

Properties

The Extract JSON action is configured using the following properties:

Converters

An optional list of data converters that should process the text.

Variable

Specifies the variable in which to store the extracted text.

Extract Path

The Extract Path step action extracts the absolute path of the element found by the finder on the step. Path in this respect is to be thought of in a very general sense. In HTML and XML, it is a tag path, such as html.body.div.text. In Excel it is a range, such as Sheet1!A5:B7. In JSON, it is the path used in JSON finders, such as @top:.a[5].b. The path extracted is always an absolute path, that is, not relative to a named tag, range, and so on, and without any wildcard (*) symbols. It is the kind of paths you see in the target views at the bottom of the Page Views, such as the Tag Path View in HTML/XML or the Cell Range View in Excel.

The extracted path may be used in the Finders of other steps in the Path property by referring to the variable containing the path.

Properties

The Extract Path action is configured using the following properties:

Variable

Specifies the variable in which to store the extracted path.

Extract Property Name

The Extract Property Name step action extracts the property name of a JSON value found by the JSON finder into a variable. The found item must be a property declaration (name/value pair) on an object, e.g. "answer" : 42..

Properties

The Extract JSON Property Name action is configured using the following properties:

Converters

An optional list of data converters that should process the text.

Variable

Specifies the variable in which to store the extracted name.

Extract Screenshot

This step must be used in the Default (WebKit) browser.

This action extracts the whole page, or a part of it, as an image and saves it to a variable.

The tag finder specifies which area to extract. Use the tag path * to extract the whole page.

Properties

The Extract Screenshot action can be configured using the following properties:

Padding

Padding for the extraction are as follows. Note that positive values make the area larger, negative values make it smaller. The value can be specified in several ways using the Value Selector.

Left (px)

Extra left margin (in pixels). May be negative.

Top (px)

Extra top margin (in pixels). May be negative.

Right (px)

Extra right margin (in pixels). May be negative.

Bottom (px)

Extra bottom margin (in pixels). May be negative.

Variable

The variable to assign the value to. It must may an image or binary variable.

Image Format

The format of the image. The possible values are PNG, JPG, BMP and GIF.

Load Images

Loads all images on page, if not already performed by the Load Page step (or similar).

Timeout (ms)

Timeout (in milliseconds) to use when loading images.

Extract Selected Option

The Extract Selected Option action extracts the selected option from a <select>-tag and stores it in a variable.

Either the option text or its value can be extracted. Before the text is stored, it can be processed using a list of data converters, and optionally trimmed for leading and trailing spaces.

Properties

The Extract action can be configured using the following properties:

Extract Value

If selected, the value of the selected option will be extracted rather than the option text itself.

Converters

An optional list of data converters that can process the text.

Trim Spaces

If selected, spaces at the beginning and the end of the text will be removed before storing the text in the variable.

Variable

Specifies the variable in which to store the extracted text.

Extract Sheet Name

The Extract Sheet Name action extracts the name of a sheet in an spreadsheet document and stores it in a variable. The sheet is identified with the aid of a Range Finder.

Properties

The Extract Sheet Name action can be configured using the following property:

Variable

The variable to store the sheet name in. This has to be a text variable.

Extract Source

This action stores the previewed data in a variable.

The step action only works on a preview.

Properties

The Extract Source action can be configured with the following properties:

Source

The type of source to extract from. Choose either Binary or Text. If the source is text, an encoding can be explicitly defined. The default encoding is the encoding provided by the web server.

Extract Data To

The variable to extract data to.

Extract Tag Attribute

This action extracts a tag attribute from the found tag, runs it through a list of data converters, and stores the result in a variable.

Properties

The Extract Tag Attribute action can be configured using the following properties:

Tag Attribute Name

The name of the tag attribute to extract.

Converters

The data converters to apply to the attribute value.

Variable

The variable to store the result into.

Example

Consider this found tag:

```
<img
src="mypicture.gif">
```

Assume that Tag Attribute Name has been set to "src", and that the Variable property has been set to "TemporaryData.shortText0". This will store the value "mypicture.gif" in the "shortText0" attribute of the "TemporaryData" variable.

Extract Target

This action extracts data from a target URL and stores it in a variable or to a file. If the selected variable can not store the actual data (such as attempting to store a PDF file in an XML variable), an error may be generated when executing the action.

The action can also optionally store the actual content type and file name of the extracted data in user specified variables.

Properties

The Extract Target action can be configured using the following properties:

Location

This property specifies which target URL to extract from.

URL

Enter the URL directly in the text field provided. Note that standard URLs using the HTTP protocol can be written in shorthand. For example, "http://www.kofax.com" can be written instead as "www.kofax.com".

URL in Found Tag

Specifies that the found tag contains the URL.

URL in Variable

Specifies that the URL should be read from a specified variable.

URL from Expression

Specifies an expression as the URL to open.

URL from Converters

Specifies a list of data converters whose output is used as the URL to open.

URL Loaded when Clicking

Specifies that the found node should be clicked, and the URL that would have been loaded as a result of this is used. For instance, if the result of clicking the found node resulted in a form submission, the data loaded by the Load Page step action will be the result of the form submission.

Store Data In

This specifies where to store the extracted data. There are two choices for this:

Variable

Specifies the variable in which to store the extracted data. The variable must be one of the following type: Binary, Image, PDF, Text, HTML, XML, or Excel.

File

Specifies the file which to write the data to.

> **Note** If a robot is in the Direct (Minimal Execution) Design Mode, then *Store In File* will work only in the Debug mode and on the RoboServer.
>
> If a robot is in the Full Execution (Smart Re-execution) Design Mode, then *Store In File* will work both in the Design and Debug modes.

File Name

Specifies the name and extension of the file.

Auto

With this option a file name is generated automatically using the following strategy:

1. First the content disposition header of the response is inspected to see if it has a filename parameter and if so that name is used.
2. Next the URL is inspected to see if it contains a file name and if so that name is used.
3. If none of the above options succeed then an error is generated.

Value, Variable, Expression and Converters

The value can be specified in several ways using the Value Selector.

Directory

Specifies the directory where the file will be placed. The value can be specified in several ways using the Value Selector.

Create Directories

Specifies whether to create all the directories in the specified path that does not already exist. If the option is selected the directories are created. If the option is not selected the directories must exist and if not then an error is generated.

Override Strategy

Specifies a strategy for what to do when the selected file already exists.

Override File

Any existing file is replaced

Never Override File

Ensures that an existing file will never be replaced. If the file already exists then an error is generated.

Create a New File

Ensures that a new file is always created. If a file with the selected name already exists, a new unique file name is generated for the file. This new file name will be the originally selected file name with a serial number added to the end just before the extension, such as myData_1.dat where _1 was added to the original file name myData.dat.

Store Meta Data In:

Specify variables to be used to store meta data about the extracted data.

Content Type

This specifies an optional variable in which to store the content type of the data. For example, the content type could look like this for an image:

image/gif

and like this for a plain text:

text/plain; charset=iso-8859-1

File Name

This specifies the optional variable in which to store the file name of the extracted data. If the data is saved to a file then the file name will be the full path of the file actually used. If the data is loaded into a variable then the file name will be the file name of the original resource (obtained from the URL or from the content disposition header for the response).

Options

The robot's options can be overridden with the step's own options. An option that is marked with an asterisk in the Options Dialog will override the one from the robot's configuration. All other options will be the same as specified for the robot.

Extract URL

Extracts a URL from a tag and stores it in a variable. If the URL is relative, it can be converted to an absolute URL using the URL of the current page.

The URL can be extracted from an attribute, e.g. in the case of an <a>-tag with a href attribute. It can also be extracted as if the tag was clicked, e.g. in the case of a button whose onClick event handler loads a new page or submits a form. If the tag is a <form>-tag, either the value of the action attribute can be extracted or the URL that corresponds to the form submission (also when the POST method is used) can be extracted.

Properties

The Extract URL action can be configured using the following properties:

Extract How

Determines how the extraction is done

Automatic

The way to extract the URL is determined automatically.

From Tag Attribute

The URL can be extracted directly from the relevant attribute of the tag for the following tags:

- <a>
- <area>
- <form>
- <frame>
- <iframe>
- <script>
- <img>
- <input type="image">
- <param>
- <link>
- <meta>
- <body>, <table>, <tr>, <td> or <th> where the tag has a background attribute

The From Tag Attribute extraction has two additional properties:

Execute JavaScript URLs

If the tag attribute contains a JavaScript URL and this property is checked, the JavaScript URL is executed in the hope that it will result in the load of a non-Javascript URL. No actual loading is done, however. If this property is not checked, the JavaScript URL itself is extracted.

Convert to Absolute URL

If this is checked, relative URLs are converted to absolute URLs.

Click Without Loading

The URL is extracted as if the tag was clicked, except that no loading is done. This can be useful for tags with onClick / onMouseDown / onMouseUp event handlers, or for buttons that submit forms.

Submit Form Without Loading

The URL is extracted as if the form was submitted, except that no actual requests are sent to the server. This type of extraction can only be applied to <form>-tags. To submit using a submit button, select the Click Without Loading extraction instead, using the submit button as the found tag.

Variable

The variable used to store to store the extracted URL.

Options

The robot's options can be overridden with the step's own options. An option that is marked with an asterisk in the Options Dialog will override the one from the robot's configuration. All other options will be the same as specified for the robot.

Extract Web Storage

The Extract Web Storage step action extracts data from the local and/or session storage from the current browser session. The local and session storages are used by some websites to persist larger amounts of data than can be stored in a cookie. The extracted web storage data is stored in a variable in JSON format. To further process the extracted data in a robot, a Create Page step can be used if it is configured to use the variable as input. The JSON will then be loaded into the browser view and additional steps can be added in order to, for instance, loop through the extracted values.

Properties

The Extract Web Storage action can be configured using the following properties:

Include Local Storage

When checked, the storage items from the local storage will be included in the extracted data. In a browser, the local storage is normally persisted across browser sessions, similarly to persistent cookies.

Include Session Storage

When checked, the storage items from the session storage will be included in the extracted data. In a browser, the session storage is normally persisted for as long as the browser window or tab exists, similarly to a session cookie.

Key Pattern

If only the stored items with a particular key are to be extracted, a pattern that matches the key of interest can be specified. If the field is left blank, all storage items will be included regardless of their keys. If a pattern is specified, note that it must match the entire key.

Domain Pattern

If only the stored items that belong to a particular domain are to be extracted, a pattern that matches the domain of interest can be specified. If the field is left blank, storage items of all domains will be included. If a pattern is specified, note that it must match the entire domain.

Output

The variable in which to store the extracted storage. The storage will be extracted in JSON format.

Find In Database

Find a value of complex type which was previously stored in a database. If the value is not found an error is generated; if the value is found it will be loaded. The database connection must be configured in Settings.

Properties

Find in Database can be configured using the following properties

Database

The database in which to find the value. A value can either be selected or hard coded at design time, or the database name can be dynamically constructed at runtime using a variable, an expression or converters - An error will occur if no database with this name exists when the robot is executed.

Variable

Select the variable to load the found value into. If any storable attribute has been added to its type since the value was stored, the value can not be loaded by using Find in Database. The variable must be of a regular complex type and not a specialized Database Output Types that existed prior to 7.2.

Key

The unique key for the value to find. This must be the key used when the value was stored. The key may have been defined in the type as "Part of Database Key", or it is can be defined using a variable, an expression or converters. If a value with the given key exists, the value will be loaded from the database into the variable. If no value was stored with the given key an error is generated; this error may be handled like any other error.

Only Override Empty Attributes

If this option is enabled, only empty attributes will be overridden with values loaded from the database. This provides the possibility to extract data before using Find in Database, and not have the extracted attribute values overridden.

For Each Browser Window

This action loops through the open browser windows (including frames and popup windows), selecting each in turn as the current window, i.e. the window that subsequent steps will work on.

For Each Data Row

This action loops through data rows in a CSV file. To use this step, load a CSV file using the Load File step. On the Preview window, click **Select action** and select **Open**. This action automatically adds a View as CSV step. Now you can use the For Each Data Row and Extract Column in Data Row steps.

Properties

Range Name

Specifies a name for the cell range the action loops through.

- Auto: Design Studio automatically assigns a name.
- Named: Provide a name for the range.

The result of this step is a named range (a row) of cells that can be used to extract data in the Extract Column in Data Row step.

For Each File

This action loops through the files in a directory.

The action loops through the files in the specified directory, optionally including the files contained in subdirectories as well (directly and indirectly). In each iteration, the file name of the current file, including the file path, will be stored in the selected variable.

If a file name pattern is specified, only the files whose file name matches this pattern will be included. Note that the pattern is matched against the file name without the path, and must match the entire name.

Note All For Each steps throw an error if the selected element does not contain any sub-elements to loop through. However, when using the For Each File step to loop through files in a directory, if no files are present in the directory, this is not regarded as an error, and no error is thrown.

Properties

The For Each File action can be configured using the following properties:

Directory Name

The name of the directory to loop through. The name can be specified in several ways using the Value Selector. The name must be an absolute directory name, including the drive name, if any, and the path to the directory.

Include Subdirectories

If this option is checked, the files contained in subdirectories (directly and indirectly) of the directory will be included; otherwise only files contained directly in the directory itself will be included.

File Name Pattern

If a pattern is specified here, only files whose name matches this pattern will be included. The pattern is matched against the file name without the path, and must match the entire name.

Store File Names Here

The variable in which to store the current file name in each iteration. The file name stored is the entire file name, including the drive name, if any, and the directory path.

For Each Item

The For Each Item action loops through all items of a JSON array. In each iteration, the appropriate item is marked as a named JSON.

The For Each Item action does not work on global variable.

Properties

The For Each Item action can be configured using the following properties:

Name

Has two options: **Auto** or **Named**.

Auto: Gives the item a name that is a number. The first auto-numbered item is 1, the next is 2, and so on.

Named: Gives the item a fixed and explicitly stated name.

See Named Tags, Ranges, and JSON for details.

Keep Existing Named Items

If checked, the existing named items will be kept, otherwise these will be unmarked as named items and only the found item will be a named item after this step.

For Each Option

This action loops through the options in a drop-down box or list box, selecting one option in each iteration.

The found tag must be a <select>-tag.

Note that selecting the options may trigger execution of JavaScript, if there are any registered event handlers on the <select>-tag.

To select options without looping, use the Select Option or Select Multiple Options action.

Properties

The For Each Option action can be configured using the following properties:

Skip these Options

If any of the options should be skipped in the loop, they should be specified here.

Continue when

Add a wait criterion for the step. If you have several wait criteria, execution stops when any wait criterion is met. You can have several met wait criteria such as if you are waiting for two HTML elements that appear on the same load, or if you are waiting for an element on the main frame, and *Initial Page Load Completes* is set. For more information, see Use Wait Criteria.

Options

The robot's options can be overridden with the step's own options. An option that is marked with an asterisk in the Options Dialog will override the one from the robot's configuration. All other options will be the same as specified for the robot.

For Each Property

The For Each Property action loops through all properties (name/value pairs) of a JSON object. In each iteration, the appropriate property is marked as a named JSON.

The For Each Property action does not work on global variable.

Properties

The For Each Property action can be configured using the following properties:

Name

Has two options: **Auto** or **Named**.

Auto: Gives the item a name that is a number. The first auto-numbered item is 1, the next is 2, and so on.

Named: Gives the item a fixed and explicitly stated name.

See Named Tags, Ranges, and JSON for details.

Keep Existing Named Items

If checked, the existing named items will be kept, otherwise these will be unmarked as named items and only the found item will be a named item after this step.

For Each Radio Button

This action loops through a group of radio buttons, selecting one of the radio buttons in each iteration.

The found tag must be one of the radio buttons in the group, *not* a tag containing all of the radio buttons.

Note that selecting the radio buttons may trigger execution of JavaScript, if there are any registered event handlers on the buttons.

To select a radio button without looping, use the Select Radio Button action.

Properties

The For Each Radio Button action can be configured using the following properties:

Skip these radio buttons
Select radio buttons to skip while looping.

Continue when
Add a wait criterion for the step. If you have several wait criteria, execution stops when any wait criterion is met. You can have several met wait criteria such as if you are waiting for two HTML elements that appear on the same load, or if you are waiting for an element on the main frame, and *Initial Page Load Completes* is set. For more information, see Use Wait Criteria.

Options

The robot's options can be overridden with the step's own options. An option that is marked with an asterisk in the Options Dialog will override the one from the robot's configuration. All other options will be the same as specified for the robot.

For Each Tag

The For Each Tag action loops through a group of tags. In each iteration, the appropriate tag is marked as a named tag.

Most often, the loop should loop from the beginning to the end, such as looping through all the <tr>-tags of a table. However, it is also possible to configure the For Each Tag action to loop through only some of the tags, such as the last n tags in a sequence.

The For Each Tag action is similar to the For Each Tag Path action. The main difference is that the For Each Tag action only finds the immediate children of the found tag, whereas the For Each Tag Path action searches the entire subtree.

Properties

The For Each Tag action can be configured using the following properties:

Tag

The name of the tags to loop through, for example, **tr**.

Classes to Include

Specify the classes of nodes to include in the result. Conjunction (logical AND) is denoted by space and disjunction (logical or) is denoted by |. Conjunction takes precedence over disjunction, for example class1 class2 specifies the nodes that are both of class1 and class2, class1 | class2 | class3 specifies the nodes that are either of class1, class2, or class3, whereas class1 class2 | class3 class4 specifies the nodes that are either both of class1 and class2 or both of class 3 and class4.

Classes to Exclude

Specify the classes of nodes to exclude from the result. Conjunction (logical AND) is denoted by space, disjunction (logical or) is denoted by |, and the explicit absence of classes is denoted by $. Conjunction takes precedence over disjunction, for example class1 class2 specifies the nodes that are both of class1 and class2, class1 | $ specifies the nodes that are either of class1 or of no class at all, class1 | class2 | class3 specifies the nodes that are either of class1, class2, or class3, whereas class1 class2 | class3 class4 specifies the nodes that are either both of class1 and class2 or both of class 3 and class4.

First Tag Number

The number of the first tag to include in the loop. The number can be specified to count either forward from the first tag, or backward from the last tag.

Last Tag Number

The number of the last tag to include in the loop. The number can be specified to count either forward from the first tag, or backward from the last tag.

Tag Number Increment

Make the loop skip tags. For example, if an increment of 2 is specified, the loop will skip every second tag

Loop Backwards

Select that the loop should loop through the matching tags in reverse order. Please note that the loop will go through exactly the same tags as if it were looping forward just in reversed order. This means that the First Tag Number is referring to first tag in the selection of tags to loop over and not the first tag visited when looping (actually it will be the last).

Number of Tags to Include Before

The number of tags (of the same name) before the named tag to include in each output.

Number of Tags to Include After

The number of tags (of the same name) after the named tag to include in each output.

Tag Name

Has two options, **Auto** or **Named**.Auto gives the tag a name which is number. The first Auto-numbered tag will have number 1, the next number 2 etc. Note that the number may change if additional Auto-numbered tags are inserted before this step (on the same page).Named gives the tag a fixed and explicitly stated name, which has several advantages:

- It is easier to remember what the named tag identifies if it has a well-chosen name
- An explicitly named tag is not affected if another named tag is inserted before it
- If you use the same name again in Set Named Tag, the name will simply be made to refer to the new tag (useful for stateful in-page looping)

Keep Existing Named Tags

If this option is selected, existing named tags are kept along with the named tag marking the result of each iteration. If this option is not selected, existing named tags are removed, and each output state will only contain the named tag marking the result of the iteration.

Example

Consider this found tag:

```
<tbody>
  <tr>...
  <tr>...
  <tr>...
  <tr>...
  <tr>...
</tbody>
```

With Tag Name set to "tr", First Tag Number set to 0 (From First), andLast Tag Number set to 1 (From Last), the For Each Tag action will loop through <tr>-tags 0, 1, 2, and 3. In each iteration, the named tag set by the action will be the appropriate <tr>-tag as shown below:

Iteration	Named Tag
1	<tr>-tag 0
2	<tr>-tag 1
3	<tr>-tag 2
4	<tr>-tag 3

For Each Tag Path

The For Each Tag Path action loops through all tags of a given type in the subtree of the found tag. In each iteration, the appropriate tag is marked as a named tag.

The For Each Tag Path action is very similar to the For Each Tag action. The main difference is that the For Each Tag action only finds the immediate children of the found tag whereas the For Each Tag Path action searches the entire subtree.

Properties

The For Each Tag Path action can be configured using the following properties:

Tag Path

Specify the tag path which is to be looped through. The tag path is specified as in a Tag Finder. All of the matching tags in the entire subtree will be found.

Classes to Include

Specify the classes of tags to include in the result. Conjunction (logical AND) is denoted by space and disjunction (logical or) is denoted by |. Conjunction takes precedence over disjunction, for example class1 class2 specifies the tags that are both of class1 and class2, class1 | class2 | class3 specifies the tags that are either of class1, class2, or class3, whereas class1 class2 | class3 class4 specifies the tags that are either both of class1 and class2 or both of class 3 and class4.

Classes to Exclude

Specify the classes of tags to exclude from the result. Conjunction (logical AND) is denoted by space, disjunction (logical or) is denoted by |, and the explicit absence of classes is denoted by $. Conjunction takes precedence over disjunction, for example class1 class2 specifies the tags that are both of class1 and class2, class1 | $ specifies the tags that are either of class1 or of no class at all, class1 | class2 | class3 specifies the tags that are either of class1, class2, or class3, whereas class1 class2 | class3 class4 specifies the tags that are either both of class1 and class2 or both of class 3 and class4.

First Tag Number

The number of the first matching tag to include in the loop. The number can be specified to count either forward from the first tag, or backward from the last tag.

Last Tag Number

The number of the last matching tag to include in the loop. The number can be specified to count either forward from the first tag, or backward from the last tag.

Tag Number Increment

Make the loop skip tags. For example, an increment of 2 is specified, the loop will skip every second tag.

Loop Backwards

Select that the loop should loop through the matching tags in reverse order. Please note that the loop will go through exactly the same tags as if it were looping forward just in reversed order. This means that the First Tag Number is referring to first tag in the selection of tags to loop over and not the first tag visited when looping (actually it will be the last).

Tag Name

Has two options, **Auto** or **Named**.Auto gives the tag a name which is number. The first Auto-numbered tag will have number 1, the next number 2 etc. Note that the number may change if additional Auto-numbered tags are inserted before this step (on the same page).Named gives the tag a fixed and explicitly stated name, which has several advantages:

- It is easier to remember what the named tag identifies if it has a well-chosen name
- An explicitly named tag is not affected if another named tag is inserted before it
- If you use the same name again in Set Named Tag, the name will simply be made to refer to the new tag (useful for stateful in-page looping)

Keep Existing Named Tags

If this option is selected, existing named tags are kept along with the named tag marking the result of each iteration. If this option is not selected, existing named tags are removed, and each output state will only contain the named tag marking the result of the iteration.

Example

Consider this found tag:

```
<table>
  <tbody>
    <tr>
      <td> 1 </td>
      <td> 2 </td>
    </tr>
  </tbody>
</table>
```

Set the Tag Path to td. The first iteration will set a named tag to <td> 1 </td> and in the next iteration, the named tag will be <td> 2 </td>

Now consider this found tag:

```
<table>
  <tbody>
    <tr>
      <td>
        <table>
          <tbody>
            <tr>
              <td> 1 </td>
              <td> 2 </td>
            </tr>
          </tbody>
        </table>
      </td>
      <td> 3 </td>
    </tr>
  </tbody>
</table>
```

Set the Tag Path to tr.td. The first iteration will set a named tag to

```
<td><table><tbody><tr><td> 1
</td><td> 2
</td></tr></tbody></table></td>
```

and in the next iteration, the named tag will be

```
<td> 3 </td>
```

For Each Text Part

This action splits a text at a specified delimiter pattern and loops through the part, assigning the next text part to a selected variable in each iteration.

Properties

The For Each Text Part step action can be configured using the following properties:

Input

The string to split can be specified in several ways using the Value Selector. If the contents of a tag should be split, you must first extract the tag text into a variable using the Extract step action.

Delimiter

Specify the delimiter at which to split the text.

See the example below where we split a text using "," as delimiter.

Output

Specifies the variable in which the text part will be stored in each iteration.

Skip Empty Output

If checked, the loop will skip iterations whose output would have been a text of length zero. For example if looping over the text "a,b,,c", the loop only contains three iterations (outputting "a", "b" and "c") if this property is checked. If the "Skip Empty Output" property is left unchecked, the loop contains four iterations (outputting "a", "b", "" and "c").

Example

We have the following input text:

```
apple,pear,banana,grape,kiwi,pineapple
```

We want to iterate over the fruits and perform some action for each; for example store the name of the fruit in a database.

As delimiter, we specify: ,

and as output variable we select Fruit.name.

In the first iteration, the Fruit.name variable will contain the value apple, in the second iteration it will contain the value pear. The entire loop will contain six iterations and in the final iteration the Fruit.name variable will contain the value pineapple.

For Each URL

The For Each URL action loops through the URLs contained in the found tag, optionally skipping duplicate URLs. The tags containing the URLs can be located at any depth inside the found tag. In each iteration, the appropriate tag is marked as a named tag.

Properties

The For Each URL action can be configured using the following properties:

URL Tags

Specifies the HTML tags whose URLs to loop through.

First URL Number

The number of the first URL to include in the loop. The number can be specified to count either forward from the first URL, or backward from the last URL.

Last URL Number

The number of the last URL to include in the loop. The number can be specified to count either forward from the first URL, or backward from the last URL.

Loop Backwards

Select that the loop should loop through the matching tags in reverse order. Please note that the loop will go through exactly the same tags as if it were looping forward just in reversed order. This means that the First Tag Number is referring to first tag in the selection of tags to loop over and not the first tag visited when looping (actually it will be the last).

URL Pattern

A pattern to match against each URL. The Action property then determines whether the URL should be skipped or not. The pattern must match the entire URL. If no pattern is specified, no pattern matching is done.

Action

If set to "Skip URLs that Match Pattern", all URLs that match the pattern will be skipped. If set to "Skip URLs that Do Not Match Pattern", all URLs that do not match the pattern will be skipped.

Skip Duplicate URLs

Specifies whether duplicate URLs (i.e. multiple identical URLs) should be skipped and hence not be looped through. Check this property the same URL should not be looped through more than once. (This is usually the case.)

Tag Name

Has two options, **Auto** or **Named**.Auto gives the tag a name which is number. The first Auto-numbered tag will have number 1, the next number 2 etc. Note that the number may change if additional Auto-numbered tags are inserted before this step (on the same page).Named gives the tag a fixed and explicitly stated name, which has several advantages:

- It is easier to remember what the named tag identifies if it has a well-chosen name
- An explicitly named tag is not affected if another named tag is inserted before it
- If you use the same name again in Set Named Tag, the name will simply be made to refer to the new tag (useful for stateful in-page looping)

Keep Existing Named Tags

f this option is selected, existing named tags are kept along with the named tag marking the result of each iteration. If this option is not selected, existing named tags are removed, and each output state will only contain the named tag marking the result of the iteration.

Generate Error

This action generates an error.

Useful if a situation has been detected that should be considered an error, such as if an error page has been received from a web site when trying to perform a certain action on the site. Using the Generate Error action, a robot error can be generated that will be handled like other errors that occur during the execution of a robot.

If it is just important to log that a certain situation has occurred and then continue execution along the current branch, consider using Write Log instead.

Properties

The Generate Error action can be configured using the following properties:

Error Message

The error message to include in the error. The error message can be specified in several ways using the Value Selector. If no error message is specified, a default error message will be used.

Get File Info

This action fetches meta data about a file in the file system.

Properties

The Get File Info action can be configured using the following properties:

File Name

The name of the file to look for. The name can be specified in several ways using the Value Selector. The name must be an absolute file name, including the drive name, if any, and the directory path to the file.

Store Last Modified In

Specifies the variable in which to store the last modified date.

Store Size In

Specifies the variable in which to store the size of the file, measured in bytes. This is the actual size of the file contents, not the file size on disk.

Get Iteration

This action gets the current iteration of an enclosing loop step and stores it in a variable.

When looping through a list of objects, this action is useful if the number of the current object that you are extracting is important.

Properties

The Get Iteration action can be configured using the following properties:

Loop Step

The number of the loop step whose iteration to get, either counting forward from the first loop step in the robot, or backward from the immediately enclosing loop step. For example, if 2 is entered and "From Last" is chosen, the iteration of the second innermost enclosing loop step will be chosen.

Store Iteration Here

The variable to store the iteration in.

Group

The Group step is designed to contain other steps that can then be hidden by collapsing the group step into a single step, which is a convenient way to structure your robot. Grouping steps inside Group steps has no effect on the execution of a robot.

Group steps contain an expand/collapse icon (+/-) located at the top left corner. Clicking on this icon will expand/collapse the Group step. When the Group step is collapsed, it looks similar to an ordinary step, and all steps inside are hidden. When the Group step is expanded, its contained steps are visible and have a layout that resembles what you would see if they were not grouped. Group steps may also be expanded or collapsed by using the Expand All / Collapse All buttons on the Toolbar which that perform these actions on all group steps in the robot, or by the Expand Groups / Collapse Groups that will perform these actions on all groups in the selection.

Steps are grouped by selecting the steps to group and using the Group action on the Toolbar, the Edit menu, or the context menu on steps. Only steps that form a subgraph may be grouped. This means that all steps must be directly connected to another step in the selection. There can only be one connection entering the selection (coming from a step outside the selection). All connections leaving the selection leading to steps outside the selection will be connected to the end of the group.

Steps are un-grouped by selecting the group steps to un-group and using the Ungroup action on the Toolbar, the Edit menu, or the context menu on steps.

At the beginning and end of the group step (when the group is expanded), you can notice a triangular marker. It marks the entry and exit of the Group step, and if you right-click one of these markers, it will be possible to perform a selection of actions on the Group step using the context menu that appears, such as insert a branch at the start of the group.

Hide Tag

This action hides the found tags.

The hiding is done by configuring the style attribute of the tags. That is, the tags are kept in the page, but configured using styles to be invisible on the page.

This action is particularly useful if tags should not be visible when clipping from the page. Hiding the tags instead of removing them avoids breaking things like JavaScript that may rely on the page having a specific structure and contents.

Properties

The Hide Tag action can be configured using the following properties:

Adjust Layout Accordingly

If this option is selected, the hiding will be done in such a way that the layout of the page adjusts in order to use the space that was taken up by the tag. If this option is not selected, the hiding will be done in such a way that the tag continues to take up the same space in the layout, but is just not shown.

Continue when

Add a wait criterion for the step. If you have several wait criteria, execution stops when any wait criterion is met. You can have several met wait criteria such as if you are waiting for two HTML elements that appear on the same load, or if you are waiting for an element on the main frame, and *Initial Page Load Completes* is set. For more information, see Use Wait Criteria.

Options

The robot's options can be overridden with the step's own options. An option that is marked with an asterisk in the Options Dialog will override the one from the robot's configuration. All other options will be the same as specified for the robot.

Insert Columns

This action inserts one or more columns in a spreadsheet.

Properties

The Insert Columns action can be configured using the following properties:

Insert Where

This option specifies where to insert the columns. Options are:

- First
- Before (Default)
- After
- Last

When a cell range is selected, the Before option inserts a column before the range, and the After option inserts a column after the range, while the First option inserts a column at the beginning within the range, and the Last option inserts a column at the end within the range.

Number of Columns

The number of columns to insert.

Set as Named Range

Range name options are:

- Auto (Default)
- Named: If you select this option, specify a name of the range.

Range Name

Has two options, **Auto** or **Named**. Auto gives the range a name which is number. The first Auto-numbered range will have number 1, the next number 2, and so on. Note that the number may change if additional Auto-numbered ranges are inserted before this step (on the same page). Named gives the range a fixed and explicitly stated name, which has several advantages:

- It is easier to remember what the named range identifies if it has a well-chosen name
- An explicitly named range is not affected if another named range is inserted before it
- If you use the same name again, the name will simply be made to refer to the new range (useful for stateful in-page looping)

Insert Content

This action inserts the specified content into a document relative to the found tag.

The step action only works on XML variables. Note that this step neither validates the XML nor resolves entities.

Properties

The Insert Content action can be configured using the following properties:

New Content

The new content to insert.

Insert Tag Where

Choose where to insert the new tag relative to the found tag. Options are:

- As First Child of Found Tag
- As Last Child of Found Tag
- Before Found Tag
- After Found Tag

Note that tags can not be inserted in a text node. Instead, the surrounding tag must be selected.

Set as Named Tag

Select this property to set a named tag on the item.

Auto

Gives the item a name which is number. The first auto-numbered item is 1, the next is 2, and etc.

Named

Gives the item a fixed and explicitly stated name.

See Named Tags, Ranges, and JSON for details.

Insert JSON

This action inserts a new property (name/value pair) into a JSON object or a new item into a JSON array.

The step action only works on JSON variables.

Properties

The Insert JSON action can be configured using the following properties:

Insert What

This option defines a new property to insert. If this option is used then the found JSON value must be an object otherwise an error is produced. The new property is defined by the two properties:

Object Property

This option defines a new property to insert. If this option is used then the found JSON value must be an object otherwise an error is produced. The new property is defined by the two properties:

Name

This is the name of the new property. This must of course be a valid property name, e.g. quotation marks should be escaped.

Value

This is the value of the property which must be a valid JSON value, e.g. if you want to insert a text this must be quoted.

Array Item

This option defines a new item to insert. If this option is used then the found JSON value must be an array otherwise an error is produced. The new property is defined by this property:

Value

This is the value of the item which must be a valid JSON value, that is. if you want to insert a text this must be quoted.

Insert Where

Choose where to insert the new JSON relative to the found JSON value. The options are:

Before

This will insert the property or item before the found JSON.

First

This will insert the property or item as the first property in a JSON object or array.

Last

This will insert the property or item as the last property in a JSON object or array.

After

This will insert the property or item after the found JSON.

Set as Named JSON

Select this property to set a named JSON on the item.

Auto

Gives the item a name that is a number. The first auto-numbered item is 1, the next is 2, and so on.

Named

Gives the item a fixed and explicitly stated name.

See Named Tags, Ranges, and JSON for details.

Insert Rows

This action inserts one or more rows in a spreadsheet.

Properties

The Insert Rows action can be configured using the following properties:

Insert Where

This option specifies where to insert the rows. Options are:

- First
- Before (Default)
- After
- Last

When a cell range is selected, the Before option inserts a row before the range, and the After option inserts a row after the range, while the First option inserts a row at the beginning within the range, and the Last option inserts a row at the end within the range.

Number of Rows

The number of rows to insert.

Set as Named Range

Range name options are:

- Auto (Default)
- Named: If you select this option, specify a name of the range.

Range Name

Has two options: **Auto** or **Named**.

Auto: Gives the item a name that is a number. The first auto-numbered item is 1, the next is 2, and so on.

Named: Gives the item a fixed and explicitly stated name.

See Named Tags, Ranges, and JSON for details.

Insert Sheet

This action inserts a new sheet in a spreadsheet.

Properties

The Insert Sheet action can be configured using the following properties:

Insert Where

This option specifies where to insert the sheet. Options are:

- First
- Before (Default)
- After
- Last

Name

This option specifies a name of the inserted sheet.

Insert Tag

The Insert Tag action inserts a new tag. Any JavaScript present in the HTML of the new tag will be executed, unless JavaScript execution is disabled in the options. The use of this step is only supported in Minimal Execution mode.

Properties

The Insert Tag action can be configured using the following properties:

HTML of New Tag

Specify the HTML of the new tag. The HTML can be specified in a number of ways as described below.

Insert Tag Where

Choose where to insert the new tag relative to the found tag. Options are:

- As First Child of Found Tag
- As Last Child of Found Tag
- Before Found Tag
- After Found Tag

Note the following rules:

- If a new tag is inserted before an <html>, <head>, or <body> tag, it will become the first tag in <body> if the document contains a <body> tag. Otherwise, it will become the first tag in <html>.
- If you insert the new tag after a <head> tag, it will become the first tag in <body>.
- If you insert the new tag after a <html> or <body> tag, it will become the last tag in <body>.
- HTML tags can not be inserted in a text node. Instead, the surrounding tag must be selected.

Continue when

Add a wait criterion for the step. If you have several wait criteria, execution stops when any wait criterion is met. You can have several met wait criteria such as if you are waiting for two HTML elements that appear on the same load, or if you are waiting for an element on the main frame, and *Initial Page Load Completes* is set. For more information, see Use Wait Criteria.

Options

The robot's options can be overridden with the step's own options. An option that is marked with an asterisk in the Options Dialog will override the one from the robot's configuration. All other options will be the same as specified for the robot.

Specifying the HTML

The HTML of the new tag can be specified in the following ways:

HTML

Simply write the HTML of the new tag.

HTML from Expression

Write an expression which result is used as the HTML of the new tag.

HTML from Expression and Pattern

Write a pattern which is matched against the found tag, and an expression whose result is used as the HTML of the new tag. Use this way of specifying the HTML if parts of the found tag should be used when creating the HTML of the new tag.

Pattern

A pattern which is matched against the found tag for the Insert Tag action. The pattern must match the entire found tag, otherwise an error will be generated.

Match Against

Specifies what the pattern should be matched against from the found tag.

- "Only Text" specifies that the pattern should be matched only against the text in the found tag.
- "HTML" specifies that the pattern should be matched against the HTML of the found tag.

Ignore Case

If this is checked, the pattern is matched against the input without regard to the character case; e.g. "KoFaX" is considered equal to "kOfax".

Expression

This field contains an expression whose result is used to create the new tag. The expression can refer to the submatches of the pattern in the Pattern field using the $n notation. For example, enter $1 in the expression to get the first submatch in the pattern (i.e. the text that matches the contents of the first pair of parentheses in the pattern).

HTML Converted from XML Variable

Choose an XML variable whose content will be transformed to HTML and used as the HTML of the new tag.

Load File

This action loads a file, either into the browser or to a variable.

Properties

The Load File action can be configured using the following properties:

File Name

This property specifies the path to the file to load. The path may be constructed using an expression or converter.

Output

This specifies what to do with the file contents; either load it into the browser or store it in a variable.

Options

The robot's options can be overridden with the step's own options. An option that is marked with an asterisk in the Options Dialog will override the one from the robot's configuration. All other options will be the same as specified for the robot.

Load Page

This action loads a page from a URL obtained either from a link on the current page, or from somewhere else. Note that loading from a link on the current page is usually easier done by using the Click action to perform a click on that link.

Properties

The Load Page action can be configured using the following properties:

Location

This property specifies which URL to open. The URL can be specified in several ways using a URL selector.

URL

Enter the URL directly in the text field provided. Note that standard URLs using the HTTP protocol can be shortened. For example, you can enter www.kofax.com instead of http://www.kofax.com.

URL in Found Tag

Specifies that the found tag contains the URL. The found tag must be one of the following types:

- <a href="URL">
- <area href="URL">
- <frame src="URL">
- <iframe src="URL">
- <script src="URL">
- <param value="URL">
- <meta http-equiv="Refresh" content="...; url=URL">

URL in Variable

Specifies that the URL should be read from a specified variable.

URL from Expression

Specifies an expression as the URL to open.

URL from Converters

Specifies a list of data converters whose output is used as the URL to open.

Load Into

This property specifies which window to load the page into. The window can be an existing one or a new one. The following options are available:

- Automatic specifies that the page should be loaded into the same window as a browser would. If the page is loaded from a URL in the found tag, this will take into account "target" attributes etc. on the found tag.
- Existing Window specifies that the page should be loaded into a selected existing window (see the discussion on how to identify a window).
- New Window specifies that the page should be loaded into a new window. An optional name for the new window can be specified, and it can be selected which window should be registered as the opener of the new window (see the discussion on how to identify a window).

Continue when

Add a wait criterion for the step. If you have several wait criteria, execution stops when any wait criterion is met. You can have several met wait criteria such as if you are waiting for two HTML elements that appear on the same load, or if you are waiting for an element on the main frame, and *Initial Page Load Completes* is set. For more information, see Use Wait Criteria.

Options

The robot's options can be overridden with the step's own options. An option that is marked with an asterisk in the Options Dialog will override the one from the robot's configuration. All other options will be the same as specified for the robot.

Ignore Styles

Select this option to ignore CSS styles for the loaded HTML page. The option is located below the Applications view.

Load Web Storage

The Load Web Storage step action loads data into the local and/or session storage. The local and session storages are used by some websites to persist larger amounts of data that can normally be stored in a cookie.

The data to load can be specified in the data field (see below for an example of how to do this), but this step action can also be used to load storage data that has been extracted using an Extract Web Storage step action.

Properties

The Load Web Storage action can be configured using the following properties:

Data

Specify the data to load into the local and/or session storage. Normally, this data is taken from a variable that stores the result of having used the Extract Web Storage action, but the data can also be entered or generated specifically.

The data must be specified in JSON format and must be specified as an array of objects. Each object must contain the following properties:

storage-type

The storage type must be either "session" or "local". Session storage will be loaded into the current window and will persist as long as its top-level window exists, similarly to a session cookie. Local storage is shared between all of the browser windows, similarly to a persistent cookie.

domain

The domain whose sites have access to the storage.

storage

An array of items that comprise the storage. Each item is an object with the following properties:

> **key**
>
> The name that is used to look up the item in the storage.
>
> **value**
>
> The stored value, which must be of type String.

Example

In this example, we wish to define one storage item named "help" with the value "http://help.kofax.com" in the local storage, and two storage items in the session storage, namely "product" with the value "Kofax RPA" and "version" with the value "11". All storage items will be defined for the domain "www.kofax.com".

We enter the following data into the Load Web Storage step action:

```
    [
            { "storage-type": "local",
              domain: "www.kofax.com",
              storage: [
                { key: "help",
                  value: "http://help.kofax.com"
                }
              ]
            },

            { "storage-type": "session",
              domain: "www.kofax.com",
              storage: [
                { key: "product",
                  value: "Kofax RPA"
                },
                { key: "version",
                  value: "11"
                }
              ]
            }
        ]
```

Loop Field Values

This action loops through a list of values, entering one into a text field in each iteration.

The found tag must be a <textarea> tag or an <input> tag of type "text" or "password".

Note that entering the value may trigger execution of JavaScript if there are any registered event handlers on the <input> or <textarea> tag.

To enter text without looping, use the Enter Text action.

Properties

The Loop Field Values action can be configured using the following properties:

Values

The values to loop through. These can be either:

a...z

The letters from a-z, in alphabetical order.

aa..zz

The two-letter combinations of a-z, in alphabetical order.

Number range

A range of integers. The numbers that define the endpoints of the range must be specified, as well as the size of the step taken in each iteration. E.g. setting "From" to 0, "To" to 100 and "Step" to 10 will cause the step to loop through the sequence 0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100.

List of values

Explicitly specify the values to loop through. The values must be separated by commas or placed on separate lines, and may be quoted with double quotes. Using the Value Selector, a list from e.g. a variable can be obtained.

Continue when

Add a wait criterion for the step. If you have several wait criteria, execution stops when any wait criterion is met. You can have several met wait criteria such as if you are waiting for two HTML elements that appear on the same load, or if you are waiting for an element on the main frame, and *Initial Page Load Completes* is set. For more information, see Use Wait Criteria.

Options

The robot's options can be overridden with the step's own options. An option that is marked with an asterisk in the Options Dialog will override the one from the robot's configuration. All other options will be the same as specified for the robot.

Loop in Excel

The Loop in Excel action loops through different elements of a spreadsheet. An element in this context could be a sheet, a column, a row or a cell and is identified by the step's Range Finder. In each iteration, the appropriate element is marked as a named range.

Properties

The Loop in Excel action can be configured using the following properties:

Loop Over

This determines what kind of element the action will loop over. There are 4 possibilities for this:

Sheets

The action will loop over sheets in the spreadsheet document. No range finder is needed for this choice.

Columns

The action will loop over the columns in the range found by the range finder.

Rows

The action will loop over the rows in the range found by the range finder.

Cells

The action will loop over the cells in the range found by the range finder.

First Index

The number of the first element to include in the loop. The number can be specified to count either forward from the first element, or backward from the last element.

Last Index

The number of the last element to include in the loop. The number can be specified to count either forward from the first element, or backward from the last element.

Increment

Makes the loop skip elements. For example, if an increment of 2 is specified, the loop will skip every second element.

Loop Backwards

Selects that the loop should loop through the matching elements in reverse order. Please note that the loop will go through exactly the same elements as if it were looping forward just in reversed order. This means that the First Index is referring to first element in the selection of elements to loop over and not the first element visited when looping (actually it will be the last when looping backwards).

Range Name

Has two options, **Auto** or **Named**.Auto gives the range a name which is number. The first Auto-numbered range will have number 1, the next number 2 etc. Note that the number may change if additional Auto-numbered ranges are inserted before this step (on the same page).Named gives the range a fixed and explicitly stated name, which has several advantages:

- It is easier to remember what the named range identifies if it has a well-chosen name
- An explicitly named range is not affected if another named range is inserted before it
- If you use the same name again in Set Named Range, the name will simply be made to refer to the new range (useful for stateful in-page looping)

Lookup Password

This action retrieves a user password from the Password store and stores it in a variable. This step action is designed to use sensitive information without disclosing it. You can store the retrieved password in a password type variable and use it in a Desktop Automation robot.

Before you can retrieve password information, the Management Console administrator must create a Password access entry in the Management Console with your access token. To obtain the token, in Design Studio, navigate to **Help** > **About** and copy the token from the **Design Studio Access Token** section. Then share the copied token with the administrator.

Important Every time you upload your robot or any of its components, such as types, snippets, and so on to the Management Console, a new Password access entry must be created for the robot. Previous entries are kept in the Password Access list and the administrator can delete them manually.

Properties

The Lookup Access action can be configured using the following properties:

User Name
Specifies the name of the user to get a password for.

Target System

Specifies the external system to get a password for. The value you type in this filed must match the value in the **Target System** property of the Password entry.

Note The target systems can be configured by an administrator to provide passwords for the same user on different systems, such as production, pre-production, and development. The Target System can also be used to partition the access to different parts of virtual machines if customers have a credit checking automation in one partition and an accounting summarization in another partition.

Variable
Name of the Password type variable to store the retrieved password.

Make Directory

This action creates a new directory on the local file system where the robot is executed.

Note that the action is only performed during execution in Design mode in Design Studio, if the option Execute in Design Mode has been selected.

Properties

The Make Directory action can be configured using the following properties:

Directory

This is the file system path or a file URL for the directory to be created. This can be specified in several ways using the Value Selector. The path must be absolute, including the drive name, if any, and the directory path to the directory. Alternatively it can be a file URL, e.g. file:/C:/temp/newDir, in which case it must be URL encoded. The separators / and \ may be used interchangeably.

Generate Error if Directory Exist

Makes the action generate an error if the directory already exists.

Create Directories

Specifies whether to create the necessary directories on the path before creating the directory. If not selected, then the action will fail if any directory on the path does not exist.

Execute in Design Mode

If this is enabled, the action will be executed even in Design Mode inside Design Studio. If this is disabled, the action will do nothing when you navigate the robot in Design Mode.

Make Snapshot

The Make Snapshot step action takes the page located in the current window and stores it in the file system, including any style sheets and images necessary to display the page as it was at the time when the snapshot was made. No JavaScript or server interaction is necessary to view the page at a later time - even if the content was dynamically generated.

This step is only supported in Minimal Execution (Direct) mode

Related Step Actions

For downloading a large number of interlinked pages, reuse shared resources and preserve links between the offline snapshots, consider using the Rewrite Page step action. Robots using the Rewrite Page step action need an external controller application to feed it URLs of pages and resources to download.

Properties

The Make Snapshot step action can be configured using the following properties:

Output Folder

The folder in which to output the snapshot. The main page (corresponding to the document in the current window) will be outputted in a file named index.html.

Download Resources

When enabled, images and other resources that have not already been loaded by the previous step actions will be downloaded and saved as part of the snapshot.

Options

The robot's options can be overridden with the step's own options. An option that is marked with an asterisk in the Options Dialog will override the one from the robot's configuration. All other options will be the same as specified for the robot.

Move Mouse From

This action emulates a mouse move away from the found tag.

This action is useful for closing JavaScript-based menus that open when the mouse is moved over them and close when the mouse is moved away from them.

To emulate a mouse movement to a tag, use the Move Mouse To action. To emulate a mouse click instead of a movement, use the Click action.

Properties

The Move Mouse From action can be configured using the following properties:

Continue when
Add a wait criterion for the step. If you have several wait criteria, execution stops when any wait criterion is met. You can have several met wait criteria such as if you are waiting for two HTML elements that appear on the same load, or if you are waiting for an element on the main frame, and *Initial Page Load Completes* is set. For more information, see Use Wait Criteria.

Options

The robot's options can be overridden with the step's own options. An option that is marked with an asterisk in the Options Dialog will override the one from the robot's configuration. All other options will be the same as specified for the robot.

Move Mouse To

This action emulates a mouse move to the found tag.

This action is useful for triggering JavaScript-based menus that open when the mouse is moved over them.

To emulate a mouse movement away from a tag, use the Move Mouse From action. To emulate a mouse click instead of a movement, use the Click action.

Properties

The Move Mouse To action can be configured using the following properties:

Continue when
Add a wait criterion for the step. If you have several wait criteria, execution stops when any wait criterion is met. You can have several met wait criteria such as if you are waiting for two HTML elements that appear on the same load, or if you are waiting for an element on the main frame, and *Initial Page Load Completes* is set. For more information, see Use Wait Criteria.

Options

The robot's options can be overridden with the step's own options. An option that is marked with an asterisk in the Options Dialog will override the one from the robot's configuration. All other options will be the same as specified for the robot.

New Window

This action creates a new window.

Properties

The New Window action can be configured using the following properties:

Window Name

The name of the new window, if any. Leave the field empty if the new window should be unnamed.

Window Opener

The window that should be registered as the opener of the new window, if any (see the discussion on how to identify a window). The opener relationship may be important to JavaScript executing in the windows

Next

This action requests another iteration in a repeat loop created using the Repeat action.

In each iteration of the repeat loop, another iteration can be requested using the Next action. The windows, pages, etc. at the Next step will be sent back to the Repeat step and will become the output from the Repeat step in the next iteration. If no Next step is executed in a given iteration, that iteration will be the last one, and the repeat loop will end.

See the Repeat action for further information.

Normalize Table

This action normalizes a table by introducing extra cells where rowspan and colspan are used, which makes iterating over table columns or rows easier. The normalization only works if colSpan/rowSpan is part of the HTML source, not if it is created using CSS.

As this action modifies the page, it can cause JavaScript or form submission to stop working, as these may rely on the structure of the page. This is however rarely a problem since the action is usually applied to data tables from which there is no further navigation.

Properties

Delete from Database can be configured using the following properties

Do not copy form fields

If this option is selected, any form fields (input, select, button, textarea) will not be copied when extra cells are inserted.

Examples

colSpan

Before

1	2
3	
4	5

After

1	2
3	3
4	5

rowSpan

Before

1	2	X
3		X

	4	X
5	6	X

After

1	2	X
3	2	X
3	4	X
5	6	X

colSpan and rowSpan

Before

<table>
<tr><td>1</td><td>2</td><td colspan="2" rowspan="3">3</td><td>4</td></tr>
<tr><td colspan="2">5</td><td rowspan="2">6</td></tr>
<tr><td colspan="2">7</td></tr>
<tr><td>8</td><td>8</td><td>8</td><td>8</td><td>8</td></tr>
</table>

After

1	2	3	3	4
5	5	3	3	6
7	7	3	3	6
8	8	8	8	8

Obsolete Step

Some of the steps have been either replaced or removed in the newer versions of Kofax RPA.

The step you are trying to get help for might have been created in one of the previous versions of the program. Please see the online help of the corresponding version of Kofax RPA.

If you want to edit your robot in the newer version of Kofax RPA, replace the obsolete step with one of the existing steps if possible. See the *Kofax RPA Release Notes* and *Kofax RPA Upgrade Guide* for upgrade instructions. Use our customer support portal for solving any problem you might have using Kofax RPA.

Open Variable

This action opens a variable attribute - or a variable of simple type - in the view, making it possible to manipulate it through here. This provides a way to visually adjust the contents of variables, making it easier to, for instance, form the precise content required for a web service parameter.

The step action only works on XML, JSON or Excel variables.

Properties

The Open Variable action can be configured using the following property:

Variable

This property specifies which variable to open. The variable must be of type XML, JSON or Excel to be loaded in this way. Note, that pre-9.3 robots use a now deprecated XML attribute type, so these will have to be upgraded to the new XML type in order to be opened. For variables of simple type, this is done by editing the variable and selecting the new XML type. For variables of complex types, the relevant .type file must be changed to use the new XML attribute type instead of the old.

Press Key

This action emulates pressing a key on the keyboard.

The found tag can be any tag, that can assume focus and receive keyboard events.

This action is useful for sending key presses, like <TAB>.

Properties

The Press Key action can be configured using the following properties:

Key to press

The key to press.

Predefined

Select from the list. The list of predefined keys contains the most commonly used keys.

Value

Specify a value of the key to press. The value must be one of the Qt key codes in a decimal number. For the key names used by Qt, see the Qt::Key documentation.

Variable

Specify a variable that contains the key code of the key to press. The key code must be one of the Qt key codes in a decimal number.

Expression

Specify an expression.

Converters

Specify a converter.

Focus on element

Focus on the element before pressing the key.

Continue when

Add a wait criterion for the step. If you have several wait criteria, execution stops when any wait criterion is met. You can have several met wait criteria such as if you are waiting for two HTML elements that appear on the same load, or if you are waiting for an element on the main frame, and *Initial Page Load Completes* is set. For more information, see Use Wait Criteria.

Options

The robot's options can be overridden with the step's own options. An option that is marked with an asterisk in the Options Dialog will override the one from the robot's configuration. All other options will be the same as specified for the robot.

Query Database

The Query Database action submits an SQL query to a database and loops through the results. The SQL should be specified using an expression. At each iteration of the result loop, the values of the current row in the result set can be assigned to variables.

Note The SQL editor shows only twenty results of a result set.

Properties

The Query Database action can be configured using the following properties:

Database

Choose which database this action should submit its query to by using the drop-down list of databases available to Design Studio.

SQL Query

This field must contain a valid SQL query in the form of an expression. The value of this expression is submitted to the chosen database. The "Edit" popup dialog allows the SQL query to be tested, showing a sample of the output.

Variables Map

Specify the mapping from result columns to variables. Click the plus sign to add a new mapping and the minus sign to remove an existing one. A mapping consists of a column name and a variable name. The column name must match the name of a column returned by the SQL Query, and the variable name is chosen from a list of existing variables. Note, that the type of the column should match the type of the chosen variable. Otherwise, an error may be generated during execution. That is, trying to store a text column in an integer variable will cause an error.

Retrieve While Looping

If this is enabled, result rows are retrieved from the database only as they are needed by the loop, iteration by iteration. See the note below on execution.

First Row in Design Mode

Used only in Design Mode in Design Studio, this is the number of the first iteration or query result row that will be accessible (counting from 1). See the note below on execution.

Rows to Use in Design Mode

Used only in Design Mode in Design Studio, this specifies the maximum number of result rows to make available for iteration. See the note below on execution.

Note: Depending on whether Retrieve While Looping is disabled or enabled, retrieval of the result rows is done in two different ways:

- Disabled: The result rows are all retrieved and saved in memory before the first iteration is executed. Thus, the database connection will be reserved for the shortest possible length of time, and the results will not be affected by any steps that are part of the loop (for example, Store In Database steps). On the other hand, available memory puts a limit to the number of result rows that can be handled without error. (This was the only option available until release 8.3).

- Enabled: The results rows are retrieved from the database one at a time as they are needed for executing each iteration of the loop. Thus, the step will be able to handle very large numbers of result rows but will hold the database connection open until all iterations of the loop have finished execution. As a side effect, the results may be affected if you make changes to the database tables referenced by the SQL query while the loop executes. However, many factors work together to determine whether the changes will in fact be visible in any particular situation.

In Debug Mode in Design Studio, retrieving while looping implies that the database connection will be held open while execution is stopped at a breakpoint or during single-stepping. If the database has a timeout for inactive connections, you may see a database error when you continue execution of the robot after a long pause.

In Design Mode in Design Studio, result rows will always be retrieved before the loop starts, thus Retrieve While Looping is effectively disabled. This is done to make it possible to switch between different iterations interactively. In order to limit the amount of memory used for this, the First Row in Design Mode and Rows to Use in Design Mode together specify a subset of result rows to load. For example, if

- First Row in Design Mode = 301
- Rows to Use in Design Mode = 100

then the loop will iterate over result rows 301 to 400 (provided the SQL query returns so many rows).

Raw HTTP

The Raw HTTP action sends a HTTP request to a web server. How the response is treated depends on the method, but in general the status code and the response headers are returned in variables defined as part of the page load options.

Properties

The Raw HTTP action can be configured by its properties:

Location

This property specifies which URL to open. The URL can be specified in several ways using the URL Selector. See the URL section in the Load Page topic for the URL Selector details.

Method

Here you specify which method to use:

- GET is used for performing a GET HTTP request.
- POST is used for performing a POST HTTP request. For POST requests specify a number of parameters as name/value pairs or give the entire body of the request. If the request is specified with parameters, you must selected whether to use POST (application/x-www-form-urlencoded) or

MULTIPART (multipart/form-data) to encode the parameters. If the entire ('raw') body of the request is given, the content type of the request data must be specified.

- PUT is used for performing PUT HTTP requests. For PUT requests specify a number of parameters as name/value pairs or give the entire body of the request.

  The following settings are common for GET, POST, and PUT requests.

  Parameters

  Here you can specify a number of parameters as name/value pairs. Click '+' to add a new parameter.

  For POST and PUT requests, MULTIPART encoding can be selected to enable file uploads. If a binary variable is selected as the value of a File Upload parameter, the bytes are submitted as-is. If Base 64 encoding is desired, the value of the parameter should be an expression base64Encode(data) where data is the name of the variable containing the binary value. In that case, it is also recommended to specify the value base64 as Content Transfer Encoding - otherwise, this field can normally be left blank.

  Accept

  This is the content types that will be accepted as response. By default, any type of response will be accepted. The accepted content types can be specified in several ways using the Value Selector.

  Encoding

  This is the encoding that will be used to encode special characters in the request.

  Store In

  This is the name of a required variable in which to store the result.

- HEAD is used for performing a HEAD HTTP request. As the HEAD request does not return any data as part of its response (only a status code and the response headers) it uses variables defined as part of the page load options to access this. A GET request can be used to simulate a HEAD request. This will result in a GET HTTP request being sent that will be aborted as soon as the header information is received i.e. the entire response will not be loaded.
- OPTIONS is used for performing an OPTIONS HTTP request. As the OPTIONS request does not return any data as part of its response (only a status code and the response headers) it uses variables defined as part of the page load options to access this. The response would normally include header fields that indicate optional features implemented by the server and applicable to the requested resource (URL), e.g. Allow: OPTIONS, TRACE, GET, HEAD

Options

The robots options can be overridden with the steps own option. The option that are changed will override the ones from the robots configuration and will be marked with an asterisk in the Options dialog.

Note If additional request headers are needed, this can also be configured under Options.

Refind Object

This step is deprecated, and it cannot be used with the new storage mechanism

This action attempts to refind an object in storage, e.g. a database. If the object can be refound, then execution along the current branch will stop. Otherwise, extraction will continue past the step containing the Refind Object action. As a result, the Refind Object action is a means to minimize robot execution time.

Properties

The Refind Object action can be configured using the following properties:

Variable

The object which is affected can be selected by choosing one from the Object drop-down box. The objects available in the drop-down box are the configured output objects.

Attributes to use for Refinding

This box is a list of object attributes that will uniquely identify the object in the storage. The Refind Object action will try to find an object in storage that matches with what you have found so far at this point in the robot execution. If such an object exists, there is little point in continuing, as the rest of the fields are probably also correct in storage.

You should try to select as few object attributes as possible, while still maintaining global uniqueness during the entire lifetime of the object.

You should not include object attributes which either are not present in all of the objects or which could change during the lifetime of the object.

Remove Attribute

This action removes an attribute from a tag in XML.

The step action only works on XML variables. Note that this step neither validates the XML nor resolves entities.

Properties

The Remove Attribute action can be configured using the following property:

Attribute Name

The name of the attribute to remove.

Remove Columns

This action removes selected columns from a spreadsheet.

Properties

The action has no properties.

Remove Content

This action removes all the content of a tag. In other words, it clears the content of a tag leaving back only the tag itself.

The step action only works on XML variables. Note that this step neither validates the XML nor resolves entities.

Properties

The Remove Content action has no properties.

Remove Cookie

The Remove Cookie action removes one or more cookies from the set of current cookies.

Properties

The Remove Cookie action can be configured using the following properties:

Domain Pattern

Specify a pattern that matches the domain of the cookie. The pattern must match the entire domain of the cookie.

Path Pattern

Specify a pattern that matches the path of the cookie. The pattern must match the entire path of the cookie.

Name Pattern

Specify a pattern that matches the name of the cookie. The pattern must match the entire name of the cookie.

Value Pattern

Specify a pattern that matches the value of the cookie. The pattern must match the entire value of the cookie.

Example

Consider a cookie named "PREF" having domain ".kofax.com", path "/" and value "123". This cookie can be removed by setting Name Pattern to "PREF". However, this will remove *all* cookies named "PREF" regardless of their domain, path or value. So if only this cookie should be removed (assuming that this is the only cookie with this name, domain and path), set Domain Pattern to ".kofax.com", Path Pattern to "/", Name Pattern to "PREF" and Value Pattern to "123".

In order to remove all cookies unconditionally, set all patterns to ".*".

Remove JSON

This action removes the found JSON from a JSON value. The action will either remove a property (name/value pair) from an object, an item from an array or the entire JSON value (in which case the result will be an empty JSON value).

The step action only works on JSON variables.

Properties

The Remove JSON action has no properties.

Remove Rows

This action removes selected rows from a spreadsheet.

Properties

The action has no properties.

Remove Sheet

This action removes a selected sheet from a spreadsheet.

Properties

The action has no properties.

Remove Table Rows

This action will remove all rows (<tr>-tags) in the input <table>-tag that do not have a certain number of columns (<td>- and <th>-tags).

Properties

The Remove Table Rows action can be configured using the following properties:

Min. Number of Columns

Enter the minimum number of columns which should always be in a table row.

Max. Number of Columns

Enter the maximum number of columns which are allowed in a table row.

Any rows that do not have a number of columns that lie in the range [Min. Number of Columns; Max. Number of Columns] will be removed.

If all rows in the table end up being removed, the Remove Table Rows action will generate an error.

Example

Consider the input:

```
<table>
  <tbody>
    <tr><td> 1 </td></tr>
    <tr><td> 2 </td><td> 2 </td></tr>
    <tr><td> 3 </td><td> 3 </td><td> 3 </td></tr>
  </tbody>
</table>
```

Assume that:

- Min. Number of Columns is set to 1
- Max. Number of Columns is set to 2

This setting will remove the last row with the three <td>-tags.

Remove Tag

The Remove Tag action removes the found tag from its parent node.

The step action only works on XML variables. Note that this step neither validates the XML nor resolves entities.

Properties

The action has no properties.

Remove Tag Range

The Remove Tag Range action removes a range of tags, i.e. all tags from one tag to another tag.

Properties

The Remove Tag Range action can be configured using the following properties:

Begin Tag

Specifies the begin tag of the tag range to remove.

Remove Begin Tag

Determines whether or not the begin tag should be removed.

End Tag

Specifies the end tag of the tag range to remove.

Remove End Tag

Determines whether or not the end tag should be removed.

Remove Tags

The Remove Tags action removes tags from the found tags that match the specified criteria, based on two sets of rules.

Set-up

The Remove Tags action is configured by defining a set of rules for tags to remove, and a set of rules to except tags from the deletion rules.

Adding and removing rules from the lists

To add a rule to one of the two lists, click '+' below the list. Removing a rule is done by selecting the rule in the list, and clicking '-' below. To reorder the rules, use the arrows below the list. After adding a rule, you can edit it by using the 'edit' button below the list.

Note Defining no rules in the list of remove rules removes all tags that are not matched by any of the except rules.

Tag matching rule configuration

On the tag matcher rule editor that opens upon clicking '+' or 'edit' below a list, you can see a choice of three different methods for matching the tags to be either removed or excluded from removal.

Select one of the following options from the Tag Matcher list.

- **Tags with this name** will match all tags with the exact name written in 'Tag Name'.
- **Tags with this name and attribute** will match all tags with the exact name written in 'Tag Name' and matching the criteria for the attributes.
- **Tags matching this pattern** will match a regular expression against the start tags including their attributes (everything between < and >). Tag matching patterns can be inverted to match all tags, that does not match the pattern.
- **Texts matching this pattern** will match a regular expression against the texts inside the found tags. If the regular expression is left empty, it will match all texts.
- **Comments** will match all comments.

Properties for the tag matching methods

The 'include children' option

Checking the 'include children' checkbox (default) means including the children for this rule. For remove rules, this means not only removing the tag (and its corresponding end tag) but also removing all contents of the tag. For except rules, this means keeping also the children (even if any of the children matches a remove rule).

This option is strongest for the except rules. That means, if a tag is matched by one of the remove rules, which is set to include its children, if any of the child tags of that tag matches one of the except rules, the child tag is kept (including its children).

Options for the 'Tags with this name' matching method

Tag Name

Specifies the exact name of the tags matched by this rule.

Options for the 'Tags with this name and attribute' matching method

Tag Name

Specifies the exact name of the tags matched by this rule. This option can be left blank, if tags should only be matched on the attribute criteria.

Attribute Name

Specifies the exact name of the attribute to match.

Attribute Value Pattern

Specifies a pattern that must match the attribute value of the specified attribute.

Options for the 'Tags matching this pattern' matching method

Tag Name

Specifies the exact name of the tags matched by this rule.

Invert Pattern

Select this option to make the rule match all tags, that do not conform to this pattern.

Options for the 'Texts matching this pattern' matching method.

Pattern

Specifies a pattern that must match the text. Leaving this pattern blank, will make it match all texts.

Options for the 'Comments' matching method.

This method has no options, and will always match all comments. Use tag finders to point out specific comments to be deleted.

Continue when

Add a wait criterion for the step. If you have several wait criteria, execution stops when any wait criterion is met. You can have several met wait criteria such as if you are waiting for two HTML elements that appear on the same load, or if you are waiting for an element on the main frame, and *Initial Page Load Completes* is set. For more information, see Use Wait Criteria.

Rename File

This action renames a file or a directory on the local file system where the robot is executed.

Note that the action is only performed during execution in Design mode in Design Studio, if the option Execute in Design Mode has been selected.

Properties

The Rename File action can be configured using the following properties:

File or Directory

This is the system path or a file URL for the file or directory to be renamed. This can be specified in several ways using the Value Selector. The path must be absolute, including the drive name, if any, and the directory path to the directory. Alternative it can be a file URL, such as file:/C:/temp/oldFile.txt, in which case it must be URL encoded. The separators / and \ may be used interchangeably.

New Name

This is the new name of the file or directory. This is actually a relative path so it can be used to move files to a different location by specifying a new name that contains a directory structure, such as ../SomeOtherFolder/newText.txt

Execute in Design Mode

If this is enabled, the action will be executed even in Design Mode inside Design Studio. If this is disabled, the action will do nothing when you navigate the robot in Design Mode.

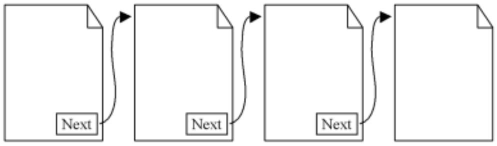
Repeat

This action creates a repeat loop together with the Next action.

The Repeat action marks the start of the repeat loop. In a subsequent step, another iteration of the loop can be requested using a Next action. The windows, pages, and so on, at the Next step will be sent back to the Repeat step and will become the output from the Repeat step in the next iteration. If no Next step is executed in a given iteration, that iteration will be the last one, and the repeat loop will end.

Notice that a robot continues along a path with the Next step on it until the path completes. And if there are database queries, which provide results to loop through on this path, the robot loops through the results of the database query before returning to the Repeat step.

The Repeat action is particularly useful for looping through pages that are connected with *next-page* links, like this:



See Each Page Links to Next for examples on how to use the Repeat action.

Replace Tag

The Replace Tag action replaces the found tag with a new tag. Any JavaScript present in the HTML of the new tag will be executed, unless JavaScript execution is disabled in the options.

Properties

The Replace Tag action can be configured using the following properties:

HTML of New Tag

Specify the HTML of the new tag. The HTML can be specified in a number of ways as described below.

Continue when

Add a wait criterion for the step. If you have several wait criteria, execution stops when any wait criterion is met. You can have several met wait criteria such as if you are waiting for two HTML elements that appear on the same load, or if you are waiting for an element on the main frame, and *Initial Page Load Completes* is set. For more information, see Use Wait Criteria.

Options

The robot's options can be overridden with the step's own options. An option that is marked with an asterisk in the Options Dialog will override the one from the robot's configuration. All other options will be the same as specified for the robot.

Specifying the HTML

The HTML of the new tag can be specified in the following ways:

HTML

Simply write the HTML of the new tag.

HTML from Expression

Write an expression whose result is used as the HTML of the new tag.

HTML from Expression and Pattern

Write a pattern which is matched against the found tag, and an expression whose result is used as the HTML of the new tag. Use this way of specifying the HTML if parts of the found tag should be used when creating the HTML of the new tag.

Pattern

A pattern which is matched against the found tag for the Replace Tag action. The pattern must match the entire found tag, otherwise an error will be generated.

Match Against

Specifies what the pattern should be matched against from the found tag.

- "Only Text" specifies that the pattern should be matched against only the text in the found tag.
- "HTML" specifies that the pattern should be matched against the HTML of the found tag.

Ignore Case

If this is checked, then the pattern is matched against the input without regard to the character case; e.g. "KoFaX" is considered equal to "kOfax".

Expression

This field contains an expression whose result is used as the HTML of the new tag. The expression can refer to the submatches of the pattern in the Pattern field using the $n notation. For example, you can enter $1 in the expression to get the first submatch in the pattern (i.e. the text that matches the contents of the first pair of parentheses in the pattern).

HTML Converted from XML Variable

Choose an XML variable whose content will be transformed to HTML and used as the HTML of the new tag.

Use Default Options

If checked, the default options of the robot are used (configured as part of the robot configuration). Otherwise, the specific options configured in the Options property below are used.

Options

Configure the options to use if the Use Default Options property is not checked.

Restore Session

The Restore Session action restores a session previously saved in a variable by another robot run using the Save Session action. See the help on Save Session for more information on reusing sessions.

If no session exists with the given parameters (e.g. has not been saved yet), the Restore Session action produces an error.

Properties

The Restore Session action can be configured using the following properties:

Restore From

Variable

Select a Session Variable.

Note The Session Pool option has been removed from the step. If an old robot with the Session Pool option is opened, a warning with a tooltip is displayed.

Resume Browser

This action resumes the browser and lets it run after either specified wait criteria are met or the browser becomes idle (whichever comes first).

Continue when
Add a wait criterion for the step. If you have several wait criteria, execution stops when any wait criterion is met. You can have several met wait criteria such as if you are waiting for two HTML elements that appear on the same load, or if you are waiting for an element on the main frame, and *Initial Page Load Completes* is set. For more information, see Use Wait Criteria.

Options

The robot's options can be overridden with the step's own options. An option that is marked with an asterisk in the Options Dialog will override the one from the robot's configuration. All other options will be the same as specified for the robot.

Return Value

This action returns a value from the robot.

The value is returned back to the client that initiated this robot run. This would normally be Java or .Net code using the Kofax RPA API to execute robots on RoboServer. The Return Value step action is also used to return values as Kapplet results and REST responses.

Prior to version 7.2 of Kofax RPA the returned value may also be handled by a storage environment, that stored the value in a database. From Kofax RPA version 7.2 and forward, values are stored in databases using the Store in Database action.

Properties

The Return Value action can be configured using the following properties:

Variable

Select the variable whose value should be returned.

If Missing Required Attributes

Here you select what should happen if one or more of the required attributes are missing.

Rewrite Page

The Rewrite Page step action takes the HTML page located in the current window, extracts the HTML content of that page and any frames that it may have and additionally outputs the links to other pages as well as the URLs of images, style sheets and other resources that the page depends on. Later, the page can be viewed offline exactly as it was at the time of the extraction.

All JavaScript and event handlers will be removed from the extracted HTML because the extracted HTML represents the result obtained from having already loaded the page and its frames and executed any

JavaScript that could generate additional content. All URLs of the page will be rewritten, first according to a user-specified transformation and then they will be converted relative URLs. URLs in inline style sheets will also be rewritten.

The external style sheets whose URLs are outputted by the step action should be run though the Rewrite Style Sheet step action which applies a similar transformation; rewriting URLs of imported style sheets and images referenced in the style sheet.

The Rewrite Page step action is intended to be used in robots that have an external controller that feeds URLs of pages, style sheets and other resources to be rewritten into the robot.

Related Step Actions

To create a quick offline snapshot of a page, the Make Snapshot step action can be used. It does not require that the robot is controlled by an external application but will - in a single step - download and save all necessary resources in the file system, forming a complete, stand-alone snapshot.

Unlike the Rewrite Page step action, the Make Snapshot step action does not preserve links between different snapshots and does not reuse shared resources between snapshots.

Note Execution of this step is controlled by the license key.

Properties

The Rewrite Page step action can be configured using the following properties:

Original Page URL

Specify the variable containing the original URL of the page in the current window. This is the URL that was used to load the page. Note that the current URL of the page may be different if the server redirected to a different page than the one that was requested.

Data Converters

The data converters that specify the transformation to perform on the URLs of the page. This can be used to specify the transformation from URL to a location in the file system. The data converters should output an absolute URL (which may be a file URL), which the step action will automatically convert to a URL that is relative to the original page URL. For advanced URL rewriting, we recommend the Convert Using JavaScript data converter.

Extracted Pages

The variable in which to store the extracted pages. The step action will extract the HTML of the page in the current window as well as HTML for each of the frames. This will be outputted in JSON format, which also contains both the original URL and rewritten URL for each of the pages. Only the main page will however have its original URL specified.

To load the JSON output into a window, use the Create Page step action with the name of the variable containing the JSON as its source of content. In the Options of the step, you may need to explicitly specify that the content type is JSON and that the encoding is UTF-8.

URLs

The variable in which to store the extracted URLs. The step action will extract the URLs of all pages, images, style sheets and other resources directly linked to by the page and its frames. Note that the style sheets and pages linked to may themselves contain URLs; these are not included in the list.

The URLs are outputted in JSON format, giving both the original URL as well as the absolute rewritten URL of each URL. Also, the type of URL is given, which is determined by the context in which the URL occurs - for instance, all URLs found in <IMG> tags are marked with type IMAGE.

The available types are:

PAGE

A link found in an anchor tag. Note that this does not imply anything about the content type of that page, as it has not yet been loaded.

IMAGE

An image.

STYLESHEET

An external CSS style sheet.

RESOURCE

A binary resource, for instance a PDF found in a frame or a Flash object.

Continue when

Add a wait criterion for the step. If you have several wait criteria, execution stops when any wait criterion is met. You can have several met wait criteria such as if you are waiting for two HTML elements that appear on the same load, or if you are waiting for an element on the main frame, and *Initial Page Load Completes* is set. For more information, see Use Wait Criteria.

Options

The robot's options can be overridden with the step's own options. An option that is marked with an asterisk in the Options Dialog will override the one from the robot's configuration. All other options will be the same as specified for the robot.

Rewrite Style Sheet

The Rewrite Style Sheet step action is to be used in conjunction with the Rewrite Page step action. It rewrites all of the URLs found in a style sheet according to a user-specified transformation and additionally converts them so that they are relative to the style sheet URL.

The Rewrite Style Sheet step action is intended to be applied on the URLs of the external style sheets found by the Rewrite Page step action. It is important that the URL transformation (specified by a list of data converters) of these two step actions are the same.

Note Execution of this step is controlled by the license key.

Properties

Style Sheet URL

The URL of the style sheet to load. If acquired from the output of a Rewrite Page step action, this is found in the originalURL property of an extracted URL of type STYLESHEET.

Data Converters

The data converters that specify the transformation to perform on the URLs of the page. This can be used to specify the transformation from URL to a location in the file system. The data converters should output an absolute URL (which may be a file URL), that the step action will then automatically convert to a URL that is relative to the original page URL. For advanced URL rewriting, we recommend the Convert Using JavaScript data converter. As mentioned above, it is important that the data converters transform URLs in the same way as the Rewrite Page step action that outputted the style sheet URL.

Output

The variable in which to store the rewritten style sheet.

URLs

The variable in which to store the extracted URLs. The step action will extract the URLs of all images and imported style sheets directly referenced by the loaded style sheet. Note that the imported style sheets may themselves contain URLs; these are not included in the list.

The URLs are outputted in JSON format, giving both the original URL as well as the absolute rewritten URL of each URL. Also, the type of URL is given, which is determined by the context in which the URL occurs - for instance, all URLs found in import statements are marked with type STYLESHEET.

The available types are:

> **IMAGE**
>
> An image.
>
> **STYLESHEET**
>
> An imported CSS style sheet.

To load the JSON output into a window, use the Open Variable step action with the name of the variable containing the JSON as its source of content. In the Options of the step, you may need to explicitly specify that the content type is JSON and that the encoding is UTF-8.

Options

The robot's loading options can be overridden with the step's own options. An option that is marked with an asterisk in the Options window will override the one from the robot's configuration. All other options will be the same as specified for the robot.

Save Session

The Save Session action saves a browser session. To transfer a session between robots, the variable containing the session can be stored in a database and restored by other robots reading it from the database. If you have a solution utilizing Kofax RPA API, you can have the session as an output from the robot and then use it as an input for another robot.

By default the complete browser state is saved. If this requires too much space, a partial state can be saved consisting of the page, the page URL and the associated cookies and authentications.

The session is stored in a variable and can be later restored by another robot run using the Restore Session action.

Properties

The Save Session action can be configured using the following properties:

Save Where

Variable

Select a Session Variable.

Note The Session Pool option has been removed from the step. If an old robot with the Session Pool option is opened, a warning with a tooltip is displayed.

Save Complete Browser State

Choose whether to save the complete browser state, or only a partial state consisting of the page, the page URL and the associated cookies and authentications.

Scroll

This action emulates scrolling a page or tag. It is particularly useful on sites where the full contents does not appear until the page or content tag has been scrolled. Note that the scroll step does not change the position of scroll bars in the browser view.

Most commonly, the Scroll step action is used to scroll an entire document, in which no tag finder should be added to the step. To scroll a particular tag that has its own scroll bar, add a tag finder that selects that tag.

Properties

The Scroll action can be configured using the following properties:

Horizontal Scroll

Specify the number of pixels to scroll to the right, relative to the current scroll position. Specify a negative number to scroll to the left. The value can be specified in several ways using the Value Selector.

Vertical Scroll

Specify the number of pixels to scroll down, relative to the current scroll position. Specify a negative number to scroll up. The value can be specified in several ways using a Value Selector.

Continue when

Add a wait criterion for the step. If you have several wait criteria, execution stops when any wait criterion is met. You can have several met wait criteria such as if you are waiting for two HTML elements that appear on the same load, or if you are waiting for an element on the main frame, and *Initial Page Load Completes* is set. For more information, see Use Wait Criteria.

Options

The robot's options can be overridden with the step's own options. An option that is marked with an asterisk in the Options Dialog will override the one from the robot's configuration. All other options will be the same as specified for the robot.

Scroll To

This step must be used in the Default (WebKit) browser.

This action scrolls the found tag into view. It is particularly useful on sites where the full contents - including images - only appear when a specific area of the page would be visible to the user interacting with the browser.

Properties

The Scroll To action can be configured using the following properties:

Continue when
Add a wait criterion for the step. If you have several wait criteria, execution stops when any wait criterion is met. You can have several met wait criteria such as if you are waiting for two HTML elements that appear on the same load, or if you are waiting for an element on the main frame, and *Initial Page Load Completes* is set. For more information, see Use Wait Criteria.

Options
The robot's options can be overridden with the step's own options. An option that is marked with an asterisk in the Options Dialog will override the one from the robot's configuration. All other options will be the same as specified for the robot.

Select File

This action selects a file to upload in a file field.

The found tag must be the <input>-tag of type file.

Properties

The Select File action can be configured using the following properties:

File to Select

Specifies the file to upload. The file must be obtained from one of the following locations:

File Contained in Variable

The file contents is read from a variable.

File Name

Specify the file name. The value can be specified in several ways using the Value Selector.

File Content Type

Specify the content type of the file. See below how it may be specified.

File Contents

Select the variable that holds the contents of the file.

File in Local File System

The file is located in the local file system.

File Name

Specify the file name. The value can be specified in several ways using the Value Selector.

File Content Type

Specify the content type of the file. See below how it may be specified.

Continue when

Add a wait criterion for the step. If you have several wait criteria, execution stops when any wait criterion is met. You can have several met wait criteria such as if you are waiting for two HTML elements that appear on the same load, or if you are waiting for an element on the main frame, and *Initial Page Load Completes* is set. For more information, see Use Wait Criteria.

Options

The robot's options can be overridden with the step's own options. An option that is marked with an asterisk in the Options Dialog will override the one from the robot's configuration. All other options will be the same as specified for the robot.

Specifying the File Content Type

The file content type can be specified in the following ways:

Automatic

The content type will be automatically determined from the content.

From Variable

The content type is specified by a variable in an object. For example, the variable should contain the text "image/gif" for a GIF image.

Predefined

The content type is selected from a list.

Custom

The content type is specified directly, e.g. "text/plain" for text contents.

Select Multiple Options

This action selects multiple options in a list box.

The found tag must be a <select>-tag with multi-selection enabled, i.e. with a "multiple" attribute present.

Note that selecting the options may trigger execution of JavaScript, if there are any registered event handlers on the <select>-tag.

To select a single option in a drop-down box or list box, use the Select Option action. To loop through options, use the For Each Option action.

Properties

The Select Multiple Options action can be configured using the following properties:

Options to Select

The options to select. You may either select ALL options or specify exactly the options to select. The list of options may be left empty to not select any options.

Keep Existing Selections

Specifies whether existing selections in the list box should be kept. If this is not selected, and if no options have been selected in the Options to Select property, all existing selections in the list box will be unselected, and no new selections will be made.

Continue when

Add a wait criterion for the step. If you have several wait criteria, execution stops when any wait criterion is met. You can have several met wait criteria such as if you are waiting for two HTML elements that appear on the same load, or if you are waiting for an element on the main frame, and *Initial Page Load Completes* is set. For more information, see Use Wait Criteria.

Options

The robot's options can be overridden with the step's own options. An option that is marked with an asterisk in the Options Dialog will override the one from the robot's configuration. All other options will be the same as specified for the robot.

Select Option

This action selects an option in a drop-down box or a list box.

The found tag must be the <select>-tag of the drop-down/list box, *not* the <option>-tag for the option to select.

Note that selecting the option may trigger execution of JavaScript, if there are any registered event handlers on the <select>-tag.

To select multiple options at the same time in a list box, use the Select Multiple Options action. To loop through options, use the For Each Option action.

Properties

The Select Option action can be configured using the following properties:

Option to Select

The option to select. This can be specified in several ways using the Value Selector. Note that this value must correspond to an <option>-tag inside the <select>-tag, and that it may match either the value as it appears in the form, or the value shown for the option in the drop-down/list box. More specifically, the matching is done as follows:

- If the value exactly matches the "value" attribute of an <option>-tag, the first such <option>-tag is selected.
- Otherwise, if the value matches the "value" attribute of an <option>-tag without regard for leading or trailing space characters, the first such <option>-tag is selected.
- Otherwise, if the value matches the text inside an <option>-tag without regard for leading or trailing space characters, the first such <option>-tag is selected.

Case Sensitive

When checked, matching of the option value is done case-sensitively.

Continue when

Add a wait criterion for the step. If you have several wait criteria, execution stops when any wait criterion is met. You can have several met wait criteria such as if you are waiting for two HTML elements that appear on the same load, or if you are waiting for an element on the main frame, and *Initial Page Load Completes* is set. For more information, see Use Wait Criteria.

Options

The robot's options can be overridden with the step's own options. An option that is marked with an asterisk in the Options Dialog will override the one from the robot's configuration. All other options will be the same as specified for the robot.

Select Radio Button

This action selects a radio button, thereby unselecting any other radio buttons in the same radio button group.

The found tag must be an <input>-tag of type radio button.

Note that selecting the radio button may trigger execution of JavaScript, if there are any registered event handlers on the radio buttons in that radio button group.

To loop through the radio buttons in a group, use the For Each Radio Button action.

Properties

The Select Radio Button action can be configured using the following properties:

Continue when

Add a wait criterion for the step. If you have several wait criteria, execution stops when any wait criterion is met. You can have several met wait criteria such as if you are waiting for two HTML elements that appear on the same load, or if you are waiting for an element on the main frame, and *Initial Page Load Completes* is set. For more information, see Use Wait Criteria.

Options

The robot's options can be overridden with the step's own options. An option that is marked with an asterisk in the Options Dialog will override the one from the robot's configuration. All other options will be the same as specified for the robot.

Send Email

The Send Email action sends an email.

Note The email is not sent during execution in Design mode in Design Studio.

Properties

The Send Email action can be configured using the following properties:

Message Tab

This tab contains the properties that allow you to specify the message content, sender and receiver.

FROM Address

Specifies the FROM Address (sender address) of the email. The value can be specified in several ways using a Value Selector.

TO Address

Specifies the TO Addresses (receiver addresses) of the email. At least one To Address must be specified and if there are more than one address these has to be separated by commas. The value can be specified in several ways using the Value Selector.

CC Address

Specifies the CC Addresses (receiver addresses) of the email. These addresses are optional, but if more than one address is specified the addresses has to be separated by commas. The value can be specified in several ways using the Value Selector.

Subject

Specifies the subject of the email. The value can be specified in several ways using the Value Selector.

Message

Specifies the message body of the email. The value can be specified in several ways using the Value Selector.

Message Type

Specifies the type of the message body, either text or HTML.

Server Tab

This tab contains the properties that allow you to configure the mail server to use.

Mail Server

Specifies the mail server to be used when sending the email. The value can be specified in several ways using a Value Selector.

Port

Specifies the port number on the mail server to be used when sending the email. The appropriate port number is most often 25 when SSL is not used, and 465 when using SSL. The value can be specified in several ways using the Value Selector.

User

Specifies the user name to be used for authentication when sending the email. If left empty, authentication is not done. The value can be specified in several ways using the Value Selector.

Password

Specifies the password to be used for authentication when sending the email. The value can be specified in several ways using the Value Selector.

Encryption

Specifies if encryption should be used.

- NONE: Credentials and email are sent in clear text.
- TLS: TLS encryption is used. Currently, TLS version 1.2 is used.

  After connection, the STARTTLS command is used to upgrade the communication channel to TLS. This only works if the SMTP server has a trusted certificate (if the server uses a self-signed certificate, it must be exported and imported into the JVM keystore using the keytool utility).
- SMTPS: SMTP over SSL. A secure channel of communication is established, over which the credentials and email is sent. This is rarely supported by SMTP servers, but will work even if the servers certificate is self-signed.

Attachment Tab

This tab contains the properties that allow you to add an attachment to the e-mail.

Include Attachment

To add an attachment to the message, check this option.

Content

The content of the attachment to include. The value can be specified in several ways using a Value Selector.

Content Type

Specifies the type of the attachment. If you select "Automatic", the type will be determined from the extension of the given file name.

File Name

Specifies the type of the attachment. If you do not specify one, a default name will be generated.

Export and import certificate

This procedure shows how to export the SMTP certificate and then import it to the JVM keystore. Follow these steps only if you configure the Send Email step with TLS and the SMTP server uses a self-signed certificate. For more information, see the section "Encryption" above.

1. Back up the existing certificate store **cacerts**, which resides in:

   `C:\Program Files\Kofax RPA <version number> x64\jre\lib\security`
2. Dump the certificate(s) specified in the STARTTLS command.

   For example, you can perform the dump using the OpenSSL tool and executing the following command:

   ```
   C:\Program Files\OpenSSL-Win64\bin>openssl s_client -showcerts
   -starttls smtp -crlf -connect smtp.gmail.com:587
   ```

   Change the server and port number as applicable.
3. Create a new certificate file (or multiple files).

   a. Copy the entire content of the dumped certificate, including the BEGIN and ENG headers, and paste it in a new file using Notepad.

   b. Select the .cer extension for the new file. For example, ABCD.cer.

      If you create multiple certificates, save them as separate files, but keep the original order, such as ABCD1.cer, ABCD2.cer, and so on.
4. Save your .cer file to the following location:

   `C:\Program Files\Kofax RPA <version number>\jre\lib\security`
5. Import the .cer file to the Kofax RPA JVM keystore.

   To do so, run the Command Prompt as administrator and execute the command similar to the following to import the dumped certificates to the cacerts file:

   ```
   C:\Program Files\Kofax RPA <version number> \jre\bin>keytool -import -
   trustcacerts -alias ABCD -file ..\lib\security\ABCD.cer -keystore ..\lib\security
   \cacerts -storepass changeit
   ```
6. When prompted, type **yes**.
7. If you have multiple certificates, repeat steps 4-6 for them. Make sure to change the `alias` parameter in the command.
8. Restart Design Studio and test the Send Email step with TLS enabled in Debug mode.

Set Attribute

This action inserts or updates an attribute on the found tag with a specified name and value. If the found tag contain an attribute with the given name its value is updated with the specified value otherwise a new attribute is inserted with the value specified.

The step action only works on XML variables. Note that this step neither validates the XML nor resolves entities.

Properties

The Set Attribute action can be configured using the following properties:

Attribute Name

The name of the attribute to set or update.

Value

The value of the attribute. This will be correctly encoded according the rules for encoding attribute values in the given type of document, e.g. XML.

Note See the Value Selector section for more information.

Set Evaluation Mode

This action changes the evaluation option for an Excel value of an Excel variable.

It is only used for an Excel variable data.

This step action makes it possible to enter any unsupported function into a cell, not causing an error.

Properties

The Set Evaluation Mode action can be configured using the following properties:

Evaluation

This property specifies whether to enable or disable an auto evaluation.

Type in `true` or `false` to enable or disable the auto evaluation.

Choose one of the following options:

- Value - Sets the value directly.
- Variable - Specifies a variable that stores the value.
- Expression - Creates an expression to specify the value.
- Converters - Specifies a data converter to set the value.

Set Checkbox

This action checks or clears a checkbox.

The found tag must be an <input>-tag of type checkbox.

Note that setting the checkbox may trigger execution of JavaScript, if there are any registered event handlers on the checkbox.

Properties

The Set Checkbox action can be configured using the following properties:

Set Checkbox to

Specifies the state that the checkbox should be set to, i.e. checked or cleared. The state is specified using the Value Selector. The resulting value must be either "true", "on", "1", or "checked" to check the checkbox, and either "false", "off", "0", or "unchecked" to clear the checkbox.

Continue when
Add a wait criterion for the step. If you have several wait criteria, execution stops when any wait criterion is met. You can have several met wait criteria such as if you are waiting for two HTML elements that appear on the same load, or if you are waiting for an element on the main frame, and *Initial Page Load Completes* is set. For more information, see Use Wait Criteria.

Options

The robot's options can be overridden with the step's own options. An option that is marked with an asterisk in the Options Dialog will override the one from the robot's configuration. All other options will be the same as specified for the robot.

Set Column Width

This action sets the width of a column in a spreadsheet. The width is specified in a number of characters.

Properties

The Set Column Width action can be configured using the following properties:

Width

This property specifies the width of a column. Options are:

- Value - Sets the value directly
- Variable - Specifies a variable that stores the width value.
- Expression - Creates an expression to specify the width.
- Converters - Specifies a data converter to set the column width.

Set Content

This action sets the specified content on a tag.

The step action only works on XML variables. Note that this step neither validates the XML nor resolves entities.

Properties

The Set Content action can be configured using the following properties:

Content Modification Mode

This property specifies which tag to modify. You can select from **Set Existing Tag** and **Set Root Tag** options.

New Content

The new content to insert.

Set Content of Cell

This action inserts the specified content to a spreadsheet cell.

Properties

The Set Content of Cell action can be configured using the following properties:

Content

This property specifies the content of a cell. Options are:

- Value - Sets the value directly
- Variable - Specifies a variable that stores the value.
- Expression - Creates an expression to specify the value.
- Converters - Specifies a data converter to set the value.

Format

This property specifies the format of a cell. Options are:

- Value - Sets a value directly
- Variable - Specifies a variable that stores the value.
- Expression - Creates an expression to specify the value.
- Converters - Specifies a data converter to set the value.

Set Content of Column

This action sets the content of a column in a spreadsheet from a variable of complex type. The attribute values of the variable is inserted consecutively beneath each other starting from the first cell of the found range.

Properties

The Set Content of Column action can be configured using the following properties:

Variable

This property specifies the name of a variable that stores the cell values.

Date Format

If any of the inserted cell values is a date then this property may be used to specify the date format of the cell. Options are:

- Value - Sets the format directly
- Variable - Specifies a variable that stores the format.
- Expression - Creates an expression to specify the format.
- Converters - Specifies a data converter to set the format.

Set Content of Row

This action sets the content of a row in a spreadsheet from a variable of complex type.

Properties

The Set Content of Row action can be configured using the following properties:

Variable

This property specifies the name of a variable that stores the row values.

Date Format

If any of the inserted cell values is a date then this property may be used to specify the date format of the cell. Options are:

- Value - Sets the format directly
- Variable - Specifies a variable that stores the format.
- Expression - Creates an expression to specify the format.
- Converters - Specifies a data converter to set the format.

Set Current Window

This action selects another window as the current window, i.e. the window that subsequent steps will work on.

In Design Studio, an easy way to insert a **Set Current Window** step is to right-click on the window's tab and choose **Set as Current Window**.

Properties

The **Set Current Window** action can be configured using the **Set Window to** properties:

Window or Frame

Use this property define a window or frame that should be selected as the current one (see Identifying a Window on how to identify a window).

Use the following options to set a current window:

- **Window Name**: Finds a window or frame by its name as displayed in the window's tab.
- **Where Window ID matches pattern**: Finds a window or frame by its ID matching a specified pattern.
- **Where HTML matches pattern**: Finds a window or frame by the HTML code matching a specified pattern.

Frame found by Tag finder

Use this property to define a certain frame as the current one with a help of a Tag finder. See the Use Tag Finders section for more information.

Set Format of Cells

This action sets the format of one or more cells in a spreadsheet.

Properties

The Set Format of Cell action can be configured using the following properties:

Format

The format of the data to insert. Options are:

- Value - Sets the format directly
- Variable - Specifies a variable that stores the format.
- Expression - Creates an expression to specify the format.
- Converters - Specifies a data converter to set the format.

Set Hyperlink on Cell

This action inserts a hyperlink to a cell.

Properties

The Set Hyperlink on Cell action can be configured using the following properties:

URL

This property specifies the address of a link. Options are:

- Value - Sets the link address directly
- Variable - Specifies a variable that stores the link address.
- Expression - Creates an expression to specify the link address.
- Converters - Specifies a data converter to set the link address.

Use default link style

This property specifies whether the default style for link should be set on the cell, i.e. text is underlined and has blue foreground color.

Set Information Property

This action sets the value of an information property in a spreadsheet.

Properties

The Set Information action can be configured using the following properties:

Name

This property specifies the name of the information property.

Value

This property specifies the content of the value. Options are:

- Value - Sets the value directly
- Variable - Specifies a variable that stores the value.
- Expression - Creates an expression to specify the value.
- Converters - Specifies a data converter to set the value.

Set JSON

This step action replace the entire found part of a JSON value with a new JSON value. The new value must be a valid JSON value, suc as an object { "answer" : 42 } or a quoted string "Life, the Universe and Everything".

The step action only works on JSON variables.

Properties

The Set JSON action can be configured using the following properties:

New Content

The new JSON value.

The value can be specified in several ways using an extended version of the Value Selector. This value selector contains the usual 4 ways to specify the value and an extra very useful selector: Generate From Variable. This selector automatically generates JSON from a variable of complex type. E.g. if you have a variable with a type that has two attributes: name and age and the values of these are "Joe" and 23 then the generated JSON could look like: { "name" : "Joe", "age" : 23 }.

Set Named JSON

This action marks the found JSON as a named JSON, so that it can be used as a reference when finding other parts of a JSON value in subsequent steps.

This is useful if several steps should work relative to this.

Properties

The Set Named JSON action can be configured using the following properties:

Name

Has two options: **Auto** or **Named**.

Auto: Gives the item a name that is a number. The first auto-numbered item is 1, the next is 2, and so on.

Named: Gives the item a fixed and explicitly stated name.

See Named Tags, Ranges, and JSON for details.

Keep Existing Named Items

If checked, the existing named items will be kept, otherwise these will be unmarked as named items and only the found item will be a named item after this step.

Set Named Range

This action marks the found range as a named range, so that it can be used as a reference when finding ranges in subsequent steps.

This is useful if several steps should work relative to this range.

Properties

The Set Named Range action can be configured using the following properties:

Range Name

Has two options, **Auto** or **Named**. Auto gives the range a name which is number. The first Auto-numbered range will have number 1, the next number 2, and so on. Note that the number may change if additional Auto-numbered ranges are inserted before this step (on the same page). Named gives the range a fixed and explicitly stated name, which has several advantages:

- It is easier to remember what the named range identifies if it has a well-chosen name
- An explicitly named range is not affected if another named range is inserted before it
- If you use the same name again in Set Named Range, the name will simply be made to refer to the new range (useful for stateful in-page looping)

Keep Existing Named Ranges

If checked, the existing named ranges will be kept, otherwise these will be unmarked as named ranges and only the found range will be a named range after this step.

Set Named Tag

This action marks the found tag as a named tag, so that it can be used as a reference when finding tags in subsequent steps.

This is useful if several steps should work relative to this tag.

Properties

The Set Named Tag action can be configured using the following properties:

Tag Name

Has two options, **Auto** or **Named**. Auto gives the tag a name which is number. The first Auto-numbered tag will have number 1, the next number 2, and so on. Note that the number may change if additional Auto-numbered tags are inserted before this step (on the same page). Named gives the tag a fixed and explicitly stated name, which has several advantages:

- It is easier to remember what the named tag identifies if it has a well-chosen name
- An explicitly named tag is not affected if another named tag is inserted before it
- If you use the same name again in Set Named Tag, the name will simply be made to refer to the new tag (useful for stateful in-page looping)

Keep Existing Named Tags

If checked, the existing named tags will be kept, otherwise these will be unmarked as named tags and only the found tag will be a named tag after this step.

Set Property Name

This action replaces the property name of the found JSON object with a new name. The value of the property remains unchanged.

The step action only works on JSON variables.

Properties

The Set Property Name action can be configured using the following properties:

Name

The new name of the property. This must of course be a valid property name, e.g. quotation marks should be escaped.

Set Range Value

This action sets a numeric value that must be no less than a minimum given value, and no more than a maximum given value. The control is represented as a slider. For example, it can be a volume control slider on a web page.

Set Row Height

This action sets the height of a row in points.

Properties

The Set Content action can be configured using the following properties:

Height

This property specifies the height of a row. Options are:

- Value - Sets the row height directly
- Variable - Specifies a variable that stores the row height.
- Expression - Creates an expression to specify the row height.
- Converters - Specifies a data converter to set the row height.

Set Sheet Name

This action sets the sheet name.

Properties

The Set Content action can be configured using the following properties:

Name

This property specifies the name of a sheet. Options are:

- Value - Sets the name directly
- Variable - Specifies a variable that stores the name.
- Expression - Creates an expression to specify the name.
- Converters - Specifies a data converter to set the name.

Set Tag

This step action replaces the entire found tag with new content. The step action is similar to the Set Content step action, but where the later only replaces the content of the tag the Set Tag step action also replaces the tag it self.

The step action only works on XML variables. Note that this step neither validates the XML nor resolves entities.

Properties

The Set Tag action can be configured using the following properties:

New Content

The new content for the entire tag.

Set Tag Name

This action replaces the name of the found tag with a new name and optionally copies the attributes of the found tag to the new tag. The child nodes of the tag are preserved.

The step action only works on XML variables. Note that this step neither validates the XML nor resolves entities.

Properties

The Set Tag Name action can be configured using the following properties:

Content Modification Mode

This property specifies the mode in which to modify the source.

Tag Name

The new name of the tag. This must of course be a valid tag name for the given document type.

Clear Attributes

If the property is set then all the attributes of the found tag are removed. If not then the attributes are copied to the new tag.

Set Text

This step action replaces the content of the found tag with a text. The step action is similar to the Set Content step action, but where the later insert the given text as markup, the Set Text step action insert the text either ampersand encoded or as CDATA.

The step action only works on XML variables. Note that this step neither validates the XML nor resolves entities.

Properties

The Set Text action can be configured using the following properties:

Text

The text to insert.

Encoding

The way the text should be encoded. There are two choices:

- "Ampersand" specifies that the text should be ampersand encode when inserted, e.g. < becomes &lt;.
- "CDATA" specifies that the text should be placed in a CDATA section.

Set Value of Cell

This action sets the value of a cell.

Properties

The Set Value of Cell action can be configured using the following properties:

Type

This property specifies the type of the value. Options are:

- Text
- Number
- Logical
- Date
- Formula

Value

This property specifies the content of the value. Options are:

- Value - Sets the value directly
- Variable - Specifies a variable that stores the value.
- Expression - Creates an expression to specify the value.
- Converters - Specifies a data converter to set the value.

Format

This property specifies the format of a cell. Options are:

- Value - Sets a format directly
- Variable - Specifies a variable that stores the format.
- Expression - Creates an expression to specify the format.
- Converters - Specifies a data converter to set the format.

Snippet Step

The Snippet step is used to insert a Snippet into a robot. A snippet is a sub-part of a robot that may be reused several times either in one robot or across robots. Snippets consists of two parts: a number of connected steps and a set of variables. When a snippet is inserted into a robot via a Snippet step the steps of the snippet is inserted at its location and the variables of the snippet is added to the variables of the robot.

Snippet steps are similar to Group steps. Just as Group steps Snippet steps contain an expand/collapse icon (+/-) located at the top left corner. Clicking on this will expand/collapse the snippet step and like group steps the inserted steps must be such that they have one entry point and one exit point. But unlike group steps, if the steps inside a Snippet step are modified then the corresponding snippet is changed and all other Snippet steps referring to the snippet will change their contained steps.

Properties

The Snippet step is configured using the following properties on the **Basic** and **Snippet** tabs:

Step Name

This is a name for the step. If you enter a step name here, it will be shown in the Robot View below the step. If no step name is provided, the snippet name is shown instead. In this case, the name is underlined.

Step Comment

This is a comment describing the snippet step. This is not a comment describing the snippet itself, which should be edited directly on the snippet when it is open in an editor.

Snippet

This is the name of the snippet. It refers to a snippet of that name inside the same project as the robot containing the Snippet step. By selecting a different name, a new snippet is inserted at the snippet step's location. If the snippet has a comment, it is shown here.

You can open the snippet that the step refers to by clicking **Open**. It is opened on a new tab.

Open XML Data Mapper

Opens the XML Data Mapper dialog. For more information, see the XML Data Mapper section.

Stop

This action causes the execution of the robot to stop without errors. That is, none of the remaining iterations, branches or steps in the robot are executed.

Store In Database

This step provides the opportunity to store a variable in a database. The database connection must be configured in Design Studio Settings.

Please read the section on Storing Data in Databases before using this step in a robot.

Properties

Store in Database can be configured using the following properties:

Database

Controls which database the variable will be stored in. A value can either be selected or hard coded at design time, or the database name can be dynamically constructed at runtime using a variable, an expression or converters - An error will occur if no database with this name exists when the robot is executed.

Variable

Select the variable to store. This must be a regular variable and not a simple type variable.

Key

The unique key for the variable which is stored. The key may be defined in the type (by marking variables as "Part of Database Key"), or defined using a variable, an expression or converters. If a value with the given key already exists, this step will override (SQL Update) the existing record; if no value with this key exists, a new record will be inserted.

Execute in Design Mode

If this is enabled, the step will be executed even in Design Mode inside Design Studio. If this is disabled, the step will do nothing when you navigate the robot in Design Mode.

Store in HBase Table

Important This step is deprecated.

This step enables you to store a variable in a HBase Table. The connection settings are configured in the step.

The description below assumes that the general architecture of the HBase cluster is understood. See the notes below for a description of how the data is inserted into the HBase Table.

Properties

Store in HBase Table is configured using the following properties:

ZooKeeper Quorum Hostnames

The list of hostnames used for connecting to the underlying ZooKeeper Quorum that directs access to HBase. The Quorum should give access to a HBase master holding the table the variable will be stored in. A value can either be selected or hard coded at design time, or the list can be dynamically constructed at runtime using a variable, an expression or converters - An error will occur if no database with this name exists when the robot is executed.

HBase Table

Controls which HBase table the variable will be stored in. A value can either be selected or hard coded at design time, or the table name can be dynamically constructed at runtime using a variable, an expression or converters - An error will occur if no database with this name exists when the robot is executed.

Column Family

Controls which column family the variable will be stored in. A value can either be selected or hard coded at design time, or the column family name can be dynamically constructed at runtime using a variable, an expression or converters - An error will occur if no database with this name exists when the robot is executed.

Variable

Select the variable to store. This must be a regular variable and not a simple type variable.

Key

The unique key for the variable which is stored. The key may be defined in the type (by marking variables as "Part of Database Key"), or defined using a variable, an expression or converters. If a value with the given key already exists, this step will update the existing record; if no value with this key exists, a new record will be inserted. See note below for important information on the key.

Execute in Design Mode

If this is enabled, the step will be executed even in Design Mode inside Design Studio. If this is disabled, the step will do nothing when you navigate the robot in Design Mode.

Note

As all data (including keys and column names) stored by HBase are arrays of bytes, Design Studio will convert all data to byte arrays using UTF-8 encoding before inserting into the table.

The variable data is written to the row with the key specified in the attribute key. If the key is set to be composed of the attributes marked as "Part of Database Key", the key is the concatenation of the value of those attributes separated by underscores (and ordered as in the type editor). For example, if writing a variable of type Company with two ordered attributes marked "Part of Database Key":

- Name = Kofax
- Year = 1999

the resulting key is Kofax_1999. Note that if choosing a binary attribute as part of the key, a hash of the value will be used instead of the actual value. Furthermore, an underscore present in the attribute values used as "Part of Database Key" will be replaced by double underscores.

Writing data to the HBase table is done by putting the value of each attribute in a column having the same qualifier as the name of the attribute. The columns are located in the column family specified. All data in the variable, including the attributes selected to be "Part of Database Key" is written to each row. The two house-hold fields RobotName and ExecutionId described in the section on Storing Data in Databases are updated in the table every time the step is executed. The remaining five house-hold fields from that page are not stored in the table. No specific timestamp is chosen for the write, it is up to HBase to choose a reasonable value. Using the example from above and Column Family = cf, the data inserted is (ignoring the timestamp): Kofax_1999 => [cf:Name = Kofax, cf:Year = 1999, cf:RobotName = XXXX, cf:ExecutionId = YYYY]

Test Cell Type

This action tests the type of a cell or cells. The types are the data types of Excel: Text, Number, Logical (boolean), Blank, Error, Formula. These essentially corresponds to the Excel functions ISTEXT, ISNUMBER, etc. If more than one cell is selected by the Range Finder, then all the found cells must satisfy the condition of the test.

Properties

The Test Cell Type action is configured using the following properties:

Condition

This contains the condition that must hold:

Is Blank

This is true if and only if the found cells are blank. This corresponds to the Excel function ISBLANK.

Is Text

This is true if and only if the found cells all contains text. This corresponds to the Excel function ISTEXT.

Is Number

This is true if and only if the found cells all contains numbers. This corresponds to the Excel function ISNUMBER.

Is Logical

This is true if and only if the found cells all contains boolean. This corresponds to the Excel function ISLOGICAL.

Is Error

This is true if and only if the found cells all contains errors. This corresponds to the Excel function ISERROR.

Is Formula

This is true if and only if the found cells all contains formulas. This corresponds to the Excel function ISFORMULA.

If

Specifies the exact condition to test for, and thus enables inversion of the stated condition. The default is to test if the "Condition is Not Satisfied". If this is selected and the condition evaluates to false, execution will be affected as indicated by the Do property; however if the condition evaluates to true, execution will continue down the current branch. It is possible instead to test if the "Condition is Satisfied", which reverses the outcome.

Do

Determines what happens when the condition and the If property together indicate that execution should not continue down the current branch.

As Specified Under Error Handling

The Error Handling tab specifies in detail what to do.

Skip Following Steps

Execution down the current branch will stop.

Test File Existence

This action tests whether a specific file exists, to determine whether execution should continue down the current branch or if something else should be done.

Properties

The Test File Existence action can be configured using the following properties:

File Name

The name of the file to look for. The name can be specified in several ways using the Value Selector. The name must be an absolute file name, including the drive name, if any, and the directory path to the file.

If

Specifies the exact condition to test for, either that the "File Exists" or that the "File Does Not Exist". If this condition is satisfied, execution will be affected as indicated by the Do property; if the condition is not satisfied, execution will continue down the current branch.

Do

Determines what happens when the condition and the If property together indicate that execution should not continue down the current branch.

> **As Specified Under Error Handling**
>
> The Error Handling tab specifies in detail what to do.
>
> **Skip Following Steps**
>
> Execution down the current branch will stop.

Test JSON Type

This action tests the type of a JSON value. The types are: Object, Array, String, Integer, Float, Boolean and Null. Besides these types, it is also possible to test for two more general types: Simple and Number.

Properties

The Test JSON Type action is configured using the following properties:

Condition

This contains the condition that must hold:

> **Is Object**
>
> This is true if and only if the found JSON value is an object. This is a JSON value of the form {...}. It is not equivalent to the JavaScript type of operation which may also return object for an array as this is considered an object in JavaScript.
>
> **Is Array**
>
> This is true if and only if the found is an Array. This is a JSON value of the form [...].
>
> **Is Simple**
>
> This is true if and only if the found JSON value is a Simple type. A Simple type is any JSON value that is not an object or an array.
>
> **Is Integer**
>
> This is true if and only if the found JSON value is a Integer. This type is an arbitrary-precision integer, but be you should be careful whenever you convert a JSON integer value to some other representation of integers, e.g. in a integer variable, that this may result in overflow.
>
> **Is Float**
>
> This is true if and only if the found JSON value is a Float. This type is an arbitrary-precision number that is not an integer, e.g. 23.345E-42, but be you should be careful whenever you convert a JSON

float value to some other number representation, e.g. in a number variable, that this may result in overflow.

Is Number

This is true if and only if the found JSON value is an Integer or a Float.

Is String

This is true if and only if the found JSON value is a String. A string must begin and end with a quotation marks ("), and various characters must be escaped with backslash (\), e.g. ", \, /, b, f, n, r, t, uXXXX, and so on.

Is Boolean

This is true if and only if the found JSON value is a Boolean. That is, oner of the two value true or false.

Is Null

This is true if and only if the found JSON value is the value null.

Note These condition are not mutually exclusive, such as "Is Number" and "Is Integer" are both true for integer values

If

Specifies the exact condition to test for, and thus enables inversion of the stated condition. The default is to test if the "Condition is Not Satisfied". If this is selected and the condition evaluates to false, execution will be affected as indicated by the Do property; however if the condition evaluates to true, execution will continue down the current branch. It is possible instead to test if the "Condition is Satisfied", which reverses the outcome.

Do

Determines what happens when the condition and the If property together indicate that execution should not continue down the current branch.

As Specified Under Error Handling

The Error Handling tab specifies in detail what to do.

Skip Following Steps

Execution down the current branch will stop.

Test Page Type

This action test the type of the page in the current window. There are currently 5 page types: HTML, XML, JSON, Excel, and Binary.

Properties

The Test Page Type action is configured using the following properties:

Page Type

This contains the page type to test for:

HTML

This is true if and only if the page in the current window is an HTML page. This does not necessarily mean that the source was HTML, but only that after the page has been loaded it was passed or converted to an HTML page and presented as such in the Page View.

XML

This is true if and only if the page in the current window is an XML page.

JSON

This is true if and only if the page in the current window is a JSON page.

Excel

This is true if and only if the page in the current window is a spreadsheet document.

Binary

This is true if and only if the page in the current window is a Binary page.

If

Specifies the exact condition to test for, and thus enables inversion of the stated condition. The default is to test if the "Condition is Not Satisfied". If this is selected and the condition evaluates to false, execution will be affected as indicated by the Do property; however if the condition evaluates to true, execution will continue down the current branch. It is possible instead to test if the "Condition is Satisfied", which reverses the outcome.

Do

Determines what happens when the condition and the If property together indicate that execution should not continue down the current branch.

As Specified Under Error Handling

The Error Handling tab specifies in detail what to do.

Skip Following Steps

Execution down the current branch will stop.

Test Row

The Test Row action tests the number of columns in a table row.

The found tag must be a <tr>-tag. Specify an interval defined by the Minimum Number of Columns and Maximum Number of Columns and choose what to do if the number of columns is outside (or inside) the specified interval.

Properties

The Test Row action can be configured using the following properties:

Minimum Number of Columns

Enter the minimum number of columns in a table row.

Maximum Number of Columns

Enter the maximum number of columns in a table row.

If

Specify the exact condition to test for, either that the "Row Matches" or that the "Row Does Not Match" the interval. The row matches the interval if the number of columns in the row is within the specified limits. If the stated condition is satisfied, execution will be affected as indicated by the Do property; if the condition is not satisfied, execution will continue down the current branch.

Do

Determines what happens when the condition and the If property together indicate that execution should not continue down the current branch.

> **As Specified Under Error Handling**
>
> The Error Handling tab specifies in detail what to do.
>
> **Skip Following Steps**
>
> Execution down the current branch will stop.

Test Tag

The Test Tag action makes a test to determine whether execution should be allowed to continue down the current branch or if something else should be done. The test consists of matching the found tag against a pattern.

Using a Test Tag action is the most common way of getting conditional execution in a robot.

Properties

The Test Tag action can be configured using the following properties:

Pattern

The pattern to match against the found tag. The pattern must match the entire found tag.

Match Against

Specifies what the pattern should be matched against from the found tag.

- **Only Text** specifies that the pattern should be matched against only the text in the found tag.
- **HTML** specifies that the pattern should be matched against the HTML of the found tag.

Ignore Case

If checked, the pattern matching will be case insensitive.

If

Specifies the exact condition to test for, either that the "Pattern Matches Found Tag" or that the "Pattern Does Not Match Found Tag". If this condition is satisfied, execution will be affected as indicated by the Do property; if the condition is not satisfied, execution will continue down the current branch.

Do

Determines what happens when the condition and the If property together indicate that execution should not continue down the current branch.

As Specified Under Error Handling

The Error Handling tab specifies in detail what to do.

Skip Following Steps

Execution down the current branch will stop.

Converters

These data converters (if any are selected) are applied to the text or HTML of the found tag *before* the pattern matching. If the data converters fail to produce any output, an error will be generated.

Test URL

This action tests the URL contained in the found tag, to determine whether execution should continue down the current branch or if something else should be done.

The URL must be contained in a tag of one of the following types:

- <a href="URL">
- <area href="URL">
- <frame src="URL">
- <iframe src="URL">
- <script src="URL">
- <param value="URL">
- <meta http-equiv="Refresh" content="...; url=URL">

Properties

The Test URL action can be configured using the following properties:

URL Pattern

The pattern which the URL in the found tag should match. The pattern must match the entire URL.

Ignore Case

If this property is checked, the pattern matching will be case insensitive.

If

Specifies the exact condition to test for, either that the "Pattern Matches URL" or that the "Pattern Does Not Match URL". If the stated condition is satisfied, execution will be affected as indicated by the Do property; if the condition is not satisfied, execution will continue down the current branch.

Do

Determines what happens when the condition and the If property together indicate that execution should not continue down the current branch.

As Specified Under Error Handling

The Error Handling tab specifies in detail what to do.

Skip Following Steps

Execution down the current branch will stop.

Test Value

This action tests a Boolean value to determine whether execution should continue down the current branch or if something else should be done. The action is among other things useful if the value of a global variable should be checked. For example, the action can be used to check if a counter has exceeded a given value.

Properties

The Test Value action is configured using the following properties:

Condition

Contains the condition. The condition must evaluate to either true or false. Any other value will produce an error when the action is executed. The default way of specifying the condition is by entering an expression. However, one the other options of the Value Selector can also be used.

If

Specifies the exact condition to test for, and thus enables inversion of the stated condition. The default is to test if the "Condition is Not Satisfied". If this is selected and the condition evaluates to false, execution will be affected as indicated by the Do property; however if the condition evaluates to true, execution will continue down the current branch. It is possible instead to test if the "Condition is Satisfied", which reverses the outcome.

Do

Determines what happens when the condition and the If property together indicate that execution should not continue down the current branch.

As Specified Under Error Handling

The Error Handling tab specifies in detail what to do.

Skip Following Steps

Execution down the current branch will stop.

Example

In the following examples, the condition is specified using an expression. This is the default way of specifying the condition.

Test if a text has a specific length:

```
ScratchPad.shortText1.length() == 28
```

Test multiple values at once:

```
PersonInput.isMale && PersonInput.isMarried
```

Test Variables

This action tests the value of one or more variables to determine whether execution should continue down the current branch or if something else should be done. The action is useful for checking that the value in an extracted variable are valid. For example, the action can be used to check that an extracted variable matches the value in an input variable.

To specify the test, add one or more *variable conditions*. Each variable condition compares the value of a selected variable against another value. Depending on the outcome of the comparisons, and on your selection in the If property, execution will either continue down the current branch or be affected as indicated by the Do property.

Properties

The Test Variables action is configured using the following properties:

Conditions

Contains the list of variable conditions. See below for more on how to configure a variable condition.

If

Determines what happens when the variable conditions have been tested.

> **No Conditions are Satisfied**
>
> Execution will continue down the current branch if one or more variable conditions are satisfied; if no conditions are satisfied, execution will be affected as indicated by the Do property.
>
> **Any Condition is Not Satisfied**
>
> Execution will continue down the current branch only if all variable conditions are satisfied; if one or more conditions are not satisfied, execution will be affected as indicated by the Do property.
>
> **Any Condition is Satisfied**
>
> Execution will continue down the current branch if no variable conditions are satisfied; if one or more variable conditions are satisfied, execution will be affected as indicated by the Do property.
>
> **All Conditions are Satisfied**
>
> Execution will continue down the current branch if one or more variable conditions are not satisfied; if all variable conditions are satisfied, execution will be affected as indicated by the Do property.

Do

Determines what happens when the condition and the If property together indicate that execution should not continue down the current branch.

As Specified Under Error Handling

The Error Handling tab specifies in detail what to do.

Skip Following Steps

Execution down the current branch will stop.

Configuring a Variable Condition

A variable condition compares the value of a selected variable against another value. A variable condition is configured using the following properties:

Basic tab

Variable

The variable (or variable attribute) whose value to test.

Operator

The operator to use when comparing the value of the variable to the other value. See below for a description of the available operators.

Value

The value to compare the variable value against. The value can be specified in several ways using the Value Selector.

Ignore Case

If this option is selected, the comparison will be done in a case-insensitive way.

Advanced tab

Always Satisfied if Variable is Empty

If this option is selected, the variable condition will be satisfied whenever the selected variable has no value, regardless of the outcome of the comparison.

Always Satisfied if Value is Empty

If this option is selected, the variable condition will be satisfied whenever the other value is empty, regardless of the outcome of the comparison. This option is useful if comparing the value of an extracted variable against a value in an input variable. If the input variable has no value for a specific attribute, the test of whether the extracted variable attribute value matches the attribute value in the input variable should usually be skipped. This is achieved by selecting this option.

The following operators are available in the Operator property:

Operator	Description
=	Tests whether the two values are equal.
<>	Tests whether the two values are not equal.
<	Tests whether the first value is less than the second value.

Operator	Description
<=	Tests whether the first value is less than or equal to the second value.
>=	Tests whether the first value is greater than or equal to the second value.
>	Tests whether the first value is greater than the second value.
contains	Tests whether the first value contains one or more occurrences of the second value. The test is done on the text representation of the values. **Note** If the first value is empty, the test is never satisfied. Also, if the first value is non-empty and the second value is empty, the test is always satisfied.
does not contain	Tests whether the first value does contain any occurrences of the second value. The test is done on the text representation of the values. Note: If the first value is empty, the test is always satisfied. Also, if the first value is non-empty and the second value is empty, the test is never satisfied.
is contained in	Tests whether the first value occurs one or more times in the second value. The test is done on the text representation of the values. Note: If the second value is empty, the test is never satisfied. Also, if the second value is non-empty and the first value is empty, the test is always satisfied.
starts with	Tests whether the first value starts with the second value. The test is done on the text representation of the values. Note: If the first value is empty, the test is never satisfied. Also, if the first value is non-empty and the second value is empty, the test is always satisfied.
ends with	Tests whether the first value ends with the second value. The test is done on the text representation of the values. Note: If the first value is empty, the test is never satisfied. Also, if the first value is non-empty and the second value is empty, the test is always satisfied.

Note The exact meaning of the operators '<>', '<', '<=', '>=', '>' depends on the type of the selected variable/attribute. For example, if the type is Short Text or Long Text, the comparison is done lexicographically, while it is done numerically if the type is Number or Integer.

Test Window

This action tests whether a specific window exists, in order to determine whether execution should continue down the current branch or if something else should be done.

Note that the window must exist when configuring the Test Window action, in order to be able to select the window.

Properties

The Test Window action can be configured using the following properties:

Window

The window to check the existence of (see how to identify a window).

If

Specifies the exact condition to test for, either that the Window Exists or that the Window Does Not Exist. If this condition is satisfied, execution will be affected as indicated by the Do property; if the condition is not satisfied, execution will continue down the current branch.

Do

Determines what happens when the condition and the If property together indicate that execution should not continue down the current branch.

As Specified Under Error Handling

The Error Handling tab specifies in detail what to do.

Skip Following Steps

Execution down the current branch will stop.

Transform XML

The Transform XML action transforms an XML document contained in an XML variable using an XSLT stylesheet. The stylesheet is specified as part of the action. The result of the transformation is stored in a variable which can be of type XML, HTML, or Long Text.

Note The Transform XML step action supports XSLT version 1.0.

The output that the stylesheet generates must be of a type that can be stored in the selected output variable. That is, if the output is to be stored in an XML variable, the stylesheet should specify <xsl:output method="xml">. If the output is to be stored in an HTML variable, the stylesheet should specify <xsl:output method="html">. If the output is to be stored in a text variable, the output method can be anything as XML, HTML, and text can all be stored as text.

A common use-case is to use an Extract Target action to store XML from a website in an XML variable, and then use a Transform XML action to transform the data and store it in the same XML variable. Finally, the Create Page action can be used to create a page displaying the XML document by choosing HTML Converted from XML Variable. This enables easy extraction of data from the transformed document using the standard extraction actions.

Properties

The Transform XML action can be configured using the following properties:

Input Variable

Select the XML variable that contains the input to the transformation. A HTML or Long Text variable can also be chosen, but it must contain valid XML.

XSLT Stylesheet

Specify the XSLT stylesheet to use for the transformation. In most cases the stylesheet will be specified as fixed XML, but you can also create the stylesheet dynamically by choosing XML from Expression or XML from Variable.

Output Variable

Select the variable where the result of the transformation should be stored. Variables of types XML, HTML, and Long Text are allowed. The XSLT stylesheet must create output that can be stored in the selected variable. Note, that the result can be stored in the same variable as the one chosen as XML input.

Transpose Table

The Transpose Table action transposes a table. Transposing a table means turning rows into columns and columns into rows while preserving the ordering between rows and between columns.

The action has no properties.

Example

As an example, consider the following table with 3 rows and 4 columns (including headers):

NAME	John	Michael	Ross
HEIGHT	1.80	1.56	2.09
WEIGHT	75	93	64

Transposing this table yields the following table with 4 rows and 3 columns (including headers):

NAME	**HEIGHT**	**WEIGHT**
John	1.80	75
Michael	1.56	93
Ross	2.09	64

Note Transposing a table twice will not necessarily result in the original table.

Try

The Try step is used when it is necessary to try several alternative approaches to get a particular thing done.

The Try step is similar to a branch point because it may have several branches going out from it. It differs from a branch point because branches beyond the first one are executed only if a step on the preceding branch activates the error handling option Try Next Alternative (or, in legacy robots, "Send Backwards").

Unhide Tag

This action unhides the found tags.

The unhiding is done by configuring the style attribute of the tags such that the tags becomes visible on the page.

Continue when

Add a wait criterion for the step. If you have several wait criteria, execution stops when any wait criterion is met. You can have several met wait criteria such as if you are waiting for two HTML elements that appear on the same load, or if you are waiting for an element on the main frame, and *Initial Page Load Completes* is set. For more information, see Use Wait Criteria.

Options

The robot's options can be overridden with the step's own options. An option that is marked with an asterisk in the Options Dialog will override the one from the robot's configuration. All other options will be the same as specified for the robot.

View as CSV

This action opens downloaded CSV content in a CSV view.

The step action only works on downloaded CSV content.

Properties

The View As CSV action can be configured with the following properties:

Use headers

This property is checked if the first row of the CSV document defines column names.

Separator Character

The separator used to parse the CSV document.

Quote Character

The quote used to parse the CSV document.

Additional Escape Character

The CSV-parser interprets the double quote character (") as an escape character if it is followed by another double quote character. If an additional escape character is used, the user may provide it here.

Encoding

The encoding used for this CSV document.

Skip Top Lines

Sets the number of lines to skip from the top of the CSV-file. This is particularly useful if a system has automatically appended a number of lines at the beginning of the CSV-file that are not part of the actual CSV-data.

Skip Bottom Lines

Sets the number of lines to skip at the bottom of the CSV-file. This is particularly useful if a system has automatically appended a number of lines at the end of the CSV-file that are not part of the actual CSV-data.

View as Excel

This action opens downloaded Excel content in an Excel view.

The step action only works on downloaded Excel content.

Properties

The action has no properties.

View as JSON

This action opens downloaded JSON content in a JSON view.

The step action only works on downloaded JSON content.

Properties

The View As JSON action can be configured with the following properties:

Encoding
The encoding used for the JSON document.

View as XML

This action opens downloaded XML content in an XML view.

The step action only works on downloaded XML content.

Properties

The action has no properties.

Wait

This action waits for a specified period of time. Wait is only performed during runtime execution, not during execution in Design mode in Design Studio.

Properties

The Wait action can be configured using the following properties:

Seconds
The number of seconds to wait. The value of seconds must be non-negative and can be specified in several ways using the Value Selector.

Write File

This action writes a new file or appends to an existing file.

Note that the file is only written during execution in Design mode in Design Studio, if the option Execute in Design Mode has been selected.

To test whether a given file exists, use the Test File Existence action.

If a CSV (Comma-Separated Value) file should be written, it can be done in two ways: Either write the file one line at a time with Write File, using an Add To CSV data converter to create each line. Alternatively, build up the desired file contents one line at a time in a global variable (again with the aid of the Add To CSV data converter) and use Write File in the end (that is, in an additional branch) to write the file in one piece.

To modify the global variable that holds the file contents, use the Assign Variable action that takes its input from a Get Variable data converter pointing to the variable itself. This is then followed by the Add To CSV data converter.

Properties

The Write File action can be configured using the following properties:

File Name

The name of the file. The name can be specified in several ways using the Value Selector. The name must be an absolute file name, including the drive name, if any, and the directory path to the file.

File Content

The contents to write to the file. The contents can be specified in several ways using the Value Selector. The contents can be either binary data or text. In the latter case, the character encoding selected in the File Encoding property will be used for encoding the text as bytes. See the Data Converters section for how to use converters to provide file contents for CSV, XML, and JSON files.

File Encoding

If the contents to write to the file is text, this property specifies the character encoding to use for encoding the text as bytes.

Append to File

Specifies whether a new file should always be created (overwriting any existing file with the same name), or whether the contents should be appended to the file if it already exists. Note that Append to File only adds new content at the end of an existing file, regardless of the file format. Therefore, this option should only be used with text variables.

Create Directories

Specifies whether to create the necessary directories on the file path before creating a new file. If not selected, then the action will fail if any directory on the path does not exist.

Execute in Design Mode

If this is enabled, the action will be executed even in Design Mode inside Design Studio. If this is disabled, the action will do nothing when you navigate the robot in Design Mode.

Write Log

This action provides a opportunity to write information to the log system.

Exactly what the log system is, depends on how the robot is being run. If the robot is running in design mode in Design Studio, the log information will be shown in the Log Window, which can be opened from the View menu. If the robot is running in debug mode, the log will be shown in the Log tab. If the Robot is run using RoboServer, the log information might be sent to a message database or somewhere else, depending on the configuration of Kofax RPA.

Properties

The Write Log action can be configured using the following properties:

Message

This field contains a message that is written to the log. The value of this can be specified in several ways using the Value Selector.

Data Converters

This section provides an overview of the available data converters.

The descriptions of the data converters below refer to the concepts of patterns and expressions, both of which are central text manipulation concepts in Kofax RPA. For a description of these two concepts, see Patterns and Expressions.

Standard

This category contains the most commonly used data converters. The most common way to process a text is to use Extract. It allows you to extract a piece of text using a pattern. To evaluate an expression, use Evaluate Expression. To add a text, use Add Text. To convert a text using a list of conversion rules, use Convert Using List. To get the value of a variable, use Get Variable.

Action	Description
Extract	This data converter extracts a piece of text from the input text using a pattern. The part that you wish to extract should be marked with a pair of parenthesis.
Evaluate Expression	This data converter outputs the result of an expression. The input text to the data converter can be used in the expression by using the INPUT notation.
Add Text	This data converter adds a text before or after the input text.
Convert Using List	This data converter converts the input text to an output text using a list of conversions.
Get Variable	This data converter fetches the value of a variable. The input text is ignored.

Extraction

This category contains data converters for extraction. The most commonly used one is Extract. It allows you to extract a piece of text using a pattern. If you need more advanced extraction features, use Advanced Extract. To apply the functionality of Advanced Extract multiple times on the same text, use Extract List.

Action	Description
Extract	This data converter extracts a piece of text from the input text using a pattern. The part that you wish to extract should be marked with a pair of parenthesis.
Advanced Extract	This data converter matches the input text against a pattern and outputs the result of an expression.
Extract List	This data converter finds all matches of a pattern, and for each match, evaluates an expression. The output text is a list of the results of the expression.

Text Formatting

This category contains data converters for various kinds of text formatting. To add a text, use Add Text. To search and replace text, use Replace Text. To search and replace text using a pattern, use Replace Pattern. To remove spaces from the input text, use Remove Spaces. To remove all special characters, use Remove Special Characters. To remove all non-printable characters, use Remove Non-Printable Characters. To change case, use Convert to Lower Case, Convert to Upper Case, or Capitalize.

Action	Description
Add Text	This data converter adds a text before or after the input text.
Replace Text	This data converter finds and replaces matching text in the input text.
Replace Pattern	This data converter replaces every match of a pattern with the result of an expression.
Remove Spaces	This data converter removes spaces from the input text.
Remove Special Characters	This data converter replaces all special characters in the input text with spaces.
Remove Non-Printable Characters	This data converter removes all non-printable characters.
Convert to Lower Case	This data converter converts all characters in the input text to lower case.
Convert to Upper Case	This data converter converts all characters in the input text to upper case.
Capitalize	This data converter capitalizes the input text, i.e. makes the first character in every word upper case, and the remaining characters lower case.
Unquote Text	This data converter unquotes text that has been enclosed in double or single quotes.

Number Handling

This category contains data converters for handling numbers. To extract a number from a text, use Extract Number. To format a number that is already in the standard number format, use Format Number.

Action	Description
Extract Number	This data converter extracts a number from the input text and outputs the number in the standard number format.
Format Number	This data converter reformats a number that is in the standard number format.

Date Handling

This category contains data converters for handling dates. To extract a date from a text, use Extract Date. To extract a year from a text, use Extract Year. To format a date that is already in the standard date format, use Format Date. To get the time between two dates, use Get Time Between Dates. To modify a date, use Modify Date.

Action	Description
Extract Date	This data converter extracts a date from the input text and outputs the date in the standard date format.
Extract Year	This data converter extracts a year from a date in the input text.
Format Date	This data converter reformats a date that is in the standard date format.
Get Time Between Dates	This data converter allows you to find the difference between two dates. It compares the date in the input text to a given date and calculates the difference, measured in the selected unit.

Action	Description
Modify Date	This data converter modifies the input date by adding or subtracting an amount from a selected part of the date. If this causes that part of the date to overflow or underflow, the other parts of the date will be updated accordingly. A time zone, in which the modifications are performed, can be specified.
To Excel Date	Converts a date from a standard date format to an Excel format.
From Excel Date	Converts a date from an Excel format to the standard date format.

HTML Handling

This category contains data converters for handling HTML. To remove all HTML tags from a text, use Remove Tags. To convert HTML to plain, structured text, use Convert HTML to Text. To reformat (pretty-print) HTML, use Format HTML. To count the number of times that a specific tag appears in the input text, use Count Tags.

Action	Description
Remove Tags	This data converter removes HTML tags from the input text.
Convert HTML to Text	This data converter converts the input HTML text to plain text, and structures the text similarly to how it would appear in a browser.
Format HTML	This data converter reformats (pretty-prints) the input HTML text.
Count Tags	This data converter counts tags with the name matching exactly the name given in the Tag Name field. If you want the converter to count only tags, that has a corresponding end tag within the input string, check the 'Require End Tag' checkbox.

Output Format Handling

This category contains data converters for handling output formats. Each data converter can format an object into a specific output format and add it to the input text (supposed to be a previously created partial result). Alternatively, any piece of text can be added to the input text according to the rules of the chosen output format.

Action	Description
Add to XML	This data converter extends a piece of XML with another piece of XML, optionally creating it first from a variable.

Encoding and Decoding

This category contains data converters for encoding and decoding.To decode or encode ampersands, use Ampersand Decode or Ampersand Encode. To encode or decode URLs, use URL Encode or URL Decode. To Base64 encode or decode binary data, use Base64 Encode or Base64 Decode.

Action	Description
Ampersand Encode	This data converter encodes characters with ampersand encodings, which is replacing certain characters with "&<something>;".
Ampersand Decode	This data converter decodes all HTML ampersand encodings into their real characters.
URL Encode	This data converter encodes characters with URL encodings.
URL Decode	This data converter decodes all URL encodings into their real characters.

Action	Description
Base64 Encode	This data converter encodes data using Base64 encoding.
Base64 Decode	This data converter decodes Base64 encoded data.
Convert Binary to Text	Converts a binary variable to text. The input text is ignored.
Convert Text to Binary	Converts a text variable to it binary Hex representation. The input text is ignored.

Other

This category contains various other data converters.

Action	Description
Evaluate Expression	This data converter outputs the result of an expression. The input text to the data converter can be used in the expression by using the INPUT notation.
Convert Using List	This data converter converts the input text to an output text using a list of conversions.
Convert Using JavaScript	This converter allows you to define the conversion using JavaScript. The input is available in the INPUT variable, and the result of the conversion must be assigned to the OUTPUT variable.
If Then	The If Then data converter allows you to specify a list of if-then rules that determine the output of the converter.
Get Variable	This data converter fetches the value of a variable. The input text is ignored.
Get Property	This data converter fetches the value of a property from a property list contained in a variable.
Make URL Absolute	This data converter makes a URL absolute.
Make URL Relative	Make URL Relative
Compute MD5 Checksum	This data converter computes an MD5 checksum of the input text.

Deprecated

This category contains data converters that have been replaced by newer versions or have otherwise become obsolete. They are available only for backwards compatibility with robots written using earlier versions of Design Studio. Do not use these data converters in new robots.

Add Text

This data converter provides the ability to add text before or after the input text.

Properties

The Add Text data converter is configured using the following properties:

Text to Add

This field contains the text to add to the input text.

Add Where

Select whether to add the text before or after the input text.

Description

Type in a description to be shown in the list of data converters. If there is no type in a description, one will be generated.

Add To CSV

This data converter converts a variable to CSV format and adds it to the input text.

See the Write File action for help on how to use this data converter to create and write the contents of a complete CSV file.

Properties

The Add To CSV data converter is configured using the following properties:

Variable to Format

Specify a variable, which will be formatted as CSV before being added at the end of the input text. The CSV will include all of the attributes of the variable, except those not marked as "storable".

Create Header

If this is checked, a header row is created that matches the selected variable. The storage names (or by default the names) of the variable's attributes are used as column titles in this header. If this property is left unchecked, a data row is created containing the values of the variable's attributes.

Date Format Locale

When creating a data row, the desired locale for dates needs to be specified just as for the Format Date data converter.

Date Format Pattern

When creating a data row, the desired date format pattern needs to be specified just as for the Format Date data converter.

Field Separator

The desired separator between individual fields in a row. This can be either a comma or a TAB character. When comma is selected, field values can optionally be put in quotes (" characters). In this case, quote characters in field values are doubled (so that " becomes ""). For quoted, comma-delimited values you can also specify whether the comma should be followed by a space or not.

Description

Type in a description to be shown in the list of data converters. If there is no type in a description, one will be generated.

Advanced Extract

This data converter allows manipulation of the input text in a flexible way using a pattern and an expression. The pattern can be used to extract text pieces from the input text, and these text pieces can then be used in the expression to construct the output text.

Properties

The Advanced Extract data converter is configured using the following properties:

Pattern

Contains a pattern that is matched against the input text. Note that the pattern must match the entire input text. Otherwise, the converter will fail, producing an error.

Ignore Case

If this option is selected, the pattern matching is case-insensitive, i.e. the pattern is matched against the input text without regard for character case.

Output Expression

Contains an expression whose result becomes the output of the data converter. The expression can refer to the submatches of the pattern using the $n notation. For example, $1 can be entered in the expression to get the first submatch in the pattern (i.e. the text that matches the contents of the first pair of parentheses in the pattern).

Description

Type in a description to be shown in the list of data converters. If there is no type in a description, one will be generated.

Ampersand Decode

This data converter decodes all HTML ampersand encodings into their real characters.

Properties

The Ampersand Decode data converter can be configured using the following properties:

Convert NBSP To Regular Space

Specifies that &nbsp; should be converted to a regular space instead of a non-breakable space.

Description

Type in a description to be shown in the list of data converters. If there is no type in a description, one will be generated.

Example

If the input text is

```
      &alt; &gt; &amp; &quot;
```

this data converter will output:

```
      < > & "
```

Ampersand Encode

This data converter encodes characters with HTML ampersand encodings.

For more information, consult:

http://www.w3.org/TR/html401/charset.html#h-5.3.2

Properties

The Ampersand Encode data converter can be configured using the following properties:

Description

Type in a description to be shown in the list of data converters. If there is no type in a description, one will be generated.

Example

If the input text is

```
     < > & " á ç a
```

this data converter will output:

```
     &alt; &gt; &amp; &quot; &aacute; &ccedil; a
```

Base64 Decode

This data converter decodes Base64 encoded text data into binary.

Properties

The Base64 Decode data converter can be configured using the following properties:

Description

Type in a description to be shown in the list of data converters. If there is no type in a description, one will be generated.

Example

If the input text is

```
     yv66vg==
```

this data converter will output:

```
     CA FE BA BE
```

Important The base64 Decode converter does not support dash (`"-"`) and underscore (`"_"`) characters, therefore it is necessary to replace those characters before doing the Base64 decoding. Use `"+"` instead of `"-"` and `"/"` instead of `"_"`.

Base64 Encode

This data converter encodes binary data into a string using Base64 encoding.

Properties

The Base64 Encode data converter can be configured using the following properties:

Description

Type in a description to be shown in the list of data converters. If there is no type in a description, one will be generated.

Example

If the input text is

```
     CA FE BA BE
```

this data converter will output:

```
     yv666vg==
```

Boolean Converter

The Boolean converter converts any pattern and returns a Boolean value: true or false if there is any pattern found. If no pattern found, no value is returned and instead, a warning message is displayed on top of the Test Input window.

Using the Boolean Converter

You can access the converter as follows:

- Extracting boolean data using the Extract action

  On the **Action** tab, select **Extract** > **Extract** then click the plus sign under **Converters** and select **Boolean Handling** > **Extract Boolean**.
- Using the right-click menu in the Page view

  Right-click the text you want to extract from and select the appropriate option on the **Extract** menu.

Properties

The Boolean converter is configured using the following properties:

Formats

Displays defined patterns and expected results: true or false.

Pattern

The Pattern list contains default suggested patterns like yes, no, and so on. You can also click the drop-down box to the right of the **Pattern** list and choose from Value, Variable, Expression, or Converters.

Ignore Case

If this option is selected, the action ignores the case of the pattern's content.

Value

Select true or false from the list.

Test Input

Enter an input value for testing.

Test Output

A corresponding output is displayed according to the defined formats.

Converting rules

- By default, if the input text is a boolean value itself, the output is the same as the input. Note that it is case sensitive. This rule only applies to lower case true and false.

- The order of the formats matters, that is the action starts searching for matching formats from the top. When any matching pattern is found, the result is returned regardless of any other patterns located below the found one.

Capitalize

This data converter capitalizes the input text. This means that the first character in every word will be made upper case, while the remaining characters will be made lower case.

Properties

The Capitalize data converter can be configured using the following properties:

Words that Should not Be Capitalized

Contains the words that should not be capitalized. The words are separated by commas. For example, if the words "in" and "the" should not be capitalized, set the property to "in, the".

Minimum Length of Words to Capitalize

The minimum length of the words that should be capitalized. For example, if this property is set to 3, all words with fewer than 3 characters will not be capitalized.

Description

Type in a description to be shown in the list of data converters. If there is no type in a description, one will be generated.

Example

If the input text is

```
    'hello WORLD'
```

then the output text becomes:

```
    'Hello World
```

Compute MD5 Checksum

The Compute MD5 Checksum data converter computes an MD5 checksum of the input text.

This data converter is useful for creating a digital fingerprint of a text or binary data, in order to easily detect changes in the data at some later point. As MD5 checksum can only be applied to binary data, any string input must be encoded before the MD5 can be applied. If the checksum generated from text is to be compared against a checksum generated by an external MD5 service, ensure that the external service used the same encoding as the robot.

Any text can be given as input, including the hexadecimal text representation that will result if it is read from an attribute containing binary data. The output of the data converter will be a 128-bit MD5 checksum of the text, represented as a 32-digit hexadecimal number. For all practical purposes, this number can be considered a unique identification of the input text. That is, in practice, two texts will never occur that result in the same MD5 checksum.

Properties

The Compute MD5 Checksum data converter can be configured using the following properties:

Encoding

The encoding used to encode the text before applying the md5.

Description

Type in a description to be shown in the list of data converters. If there is no type in a description, one will be generated.

Example

If the input text is

```
    The quick brown fox jumps over the lazy dog
```

then the output will be:

```
    9E107D9D372BB6826BD81D3542A419D6
```

Even small changes in the input text will result in a completely different output. For example, if you change the 'd' in "dog" to an 'h', so that the input becomes:

```
    The quick brown fox jumps over the lazy hog
```

then the output will be:

```
    5681F8C64F7CA70B12E0B80435265203
```

Convert Binary to Text

Converts the contents of a binary variable to text, using the selected encoding. The input text is ignored. This action is used when HTML, XML, or text files are passed as input to a robot through a binary variable.

Properties

The Convert Binary to Text data converter is configured using the following properties:

Variable

The variable whose value to get. This must be a binary attribute.

Encoding

The encoding used to decode the bytes in the binary variable. Binary content which can be converted into text is most often encoded in UTF-8, Latin-1 (ISO-8859-1), or Latin-1 (Windows-1252). If the text generated by this conversion contains the character '#', the selected encoding is wrong or the character is non-displayable.

Description

Type in a description to be shown in the list of data converters. If there is no type in a description, one will be generated.

Convert from Excel Date

This data converter converts the date from an Excel number to a standard format.

Properties

The Convert from Excel Date data converter can be configured using the following properties:

Description

Type in a description to be shown in the list of data converters. If there is no type in a description, one will be generated.

Example

If the input text is

```
1
```

then the output text becomes:

```
1900-01-01 00:00:00.0
```

Convert HTML to Text

This data converter converts the HTML input text to plain text, and structures the text similarly to how it would appear in a browser.

Properties

The Convert HTML to Text data converter can be configured using the following properties:

Include Aligned Tables and Images

Specifies that the tables and images that are aligned to the left or right of the text are included in the output text. Disabling this can sometimes result in removing the desired content.

Include URLs

Specifies that the actual URLs in link tags will be included in the output text.

Include Image Text Alternatives

Specifies that the text representation of images will be included in the output text.

Include Form Fields

Specifies that the text representation of form fields will be included in the output text.

Insert This Before a Heading

Specifies that this data converter should guess at the location of headings and insert the specified text before them.

Insert This After a Heading

Specifies that this data converter should guess at the location of headings and insert the specified text after them.

Keep Ampersand Encodings

Specifies that ampersand encodings will not be decoded. Text in script and style sheet will be respected.

Description

Type in a description to be shown in the list of data converters. If there is no type in a description, one will be generated.

Convert Text to Binary

Converts the contents of a text variable into its binary Hex representation, using the selected encoding. This step can be used to encode text as binary for file upload.

Properties

The Convert Text to Binary data converter is configured using the following properties:

Encoding

The encoding used to encode the text.

Description

Type in a description to be shown in the list of data converters. If there is no type in a description, one will be generated.

Convert to Excel Date

This data converter converts the date to an Excel number that represents the same date in a spreadsheet.

Properties

The Convert to Excel Date data converter can be configured using the following properties:

Description

Type in a description to be shown in the list of data converters. If there is no type in a description, one will be generated.

Example

If the input text is

```
1900-01-01 00:00:00.0
```

then the output text becomes:

```
1.0
```

Convert to Lower Case

This data converter converts all characters in the input text to lower case.

Properties

The To Lower Case data converter can be configured using the following properties:

Description

Type in a description to be shown in the list of data converters. If there is no type in a description, one will be generated.

Example

If the input text is

```
    "HELLO World"
```

then the output text becomes:

```
    "hello world"
```

Convert to Upper Case

This data converter converts all characters in the input text to upper case.

Properties

The To Upper Case data converter can be configured using the following properties:

Description

Type in a description to be shown in the list of data converters. If there is no type in a description, one will be generated.

Example

If the input text is

```
    "hello World"
```

then the output text becomes:

```
    "HELLO WORLD"
```

Convert Using JavaScript

This data converter provides the opportunity to specify the conversion using JavaScript. This data converter may for instance be useful when working with advanced text manipulation such as URL rewriting, or performing complex calculations.

The input to the converter is available in an implicitly defined INPUT variable. The result of performing the conversion must be assigned to an OUTPUT variable. Helper functions can be defined and called if needed. Note that access to the browser state from the JavaScript is not possible in this converter.

Properties

The Convert Using JavaScript data converter is configured using the following properties:

Script

The JavaScript to execute. This may be specified literally or created in several different ways using the Value Selector.

Description

Type in a description to be shown in the list of data converters. If there is no type in a description, one will be generated.

Example: JavaScript Conversion

Example 1

To calculate the average of a comma-separated list of numbers, we configure the Convert Using JavaScript data converter with the following script that performs the calculation:

```
var sum = 0;
var numbers = INPUT.split(",");
for (var i = 0; i < numbers.length; i++) {
    sum += parseInt(numbers[i]);
}

OUTPUT = sum / numbers.length;
```

The JavaScript reads the list of numbers, e.g. "23,22,25,31,24" from the INPUT variable, splits it at each comma using the built-in JavaScript split() function, iterates through the numbers to calculate the sum (note that the parseInt() function is used to convert from string to integer; otherwise we would be concatenating the strings rather than calculating the sum), and finally calculates the average and assigns the result to the OUTPUT variable.

In the case where the input to the data converter is the string "23,22,25,31,24", the output of the converter will thus be 25.

Example 2

To calculate the maximum of a comma-separated list of amounts, e.g. "$10.50,$13,$21.75,$7", we configure the Convert Using JavaScript data converter with the following script that performs the calculation:

```
function getNumber(amountWithDollarSign) {
    return parseFloat(amountWithDollarSign.substring(1));
}

var amountsWithDollarSigns = INPUT.split(",");
var max = getNumber(amountsWithDollarSigns[0]);
for (var i = 1; i < amountsWithDollarSigns.length; i++) {
    max = Math.max(max, getNumber(amountsWithDollarSigns[i]));
}

OUTPUT = max;
```

In the above JavaScript, we have defined a helper function named getNumber() that removes the preceding dollar sign from the amount and converts the rest into a floating-point number. This function is called repeatedly in script. The built-in JavaScript function Math.max() used to find the maximum of two numbers; in each iteration, the highest number found so far is compared with the next number.

Finally, the highest number found is stored in the OUTPUT variable and becomes the output of the data converter.

String Manipulation

The following methods are useful when converting String objects. Notice that strings are written inside "" whereas regexp are written within //. A global g attribute at the end of a regexp indicates that the method should apply to all matches.

Method	Description
string.charAt(n)	Returns the character with index n.
string.charCodeAt(n)	Returns the Unicode value of the character with index n.
string.concat(value1, value2, ...)	One or more values are concatenated to string.
String.fromCharCode(c1, c2, ...)	Creates a new string from integers specifying the Unicode encodings of the characters.

string.indexOf(substring)
string.indexOf(substring, start)
	Returns the index of substring within string.start specifies the index at which the search should start (0 being the first character in the string and string.length-1 being the last. Default is 0).

string.lastIndexOf(substring)
string.lastIndexOf(substring, start)
	Returns the position of the last occurrence of substring within string.start specifies the index at which the search should start (0 being the first character in the string and string.length-1 (which is default) being the last).

string.length	Character length of string.
string.match(regexp)	Searches string for matches with a regular expression regexp. Returns only the first match unless regexp has the global attribute specified (g), by which an array containing all results from the match is returned.

string.replace(regexp, replacement)
string.replace(substring, replacement)
	Searches string for matches with a substring or a regular expression and replaces the first occurrence with replacement. If regexp has the global attribute specified (g), all occurrences are replaced with replacement.

string.search(regexp)	Returns the index of the first character of the first match with a regular expression.
string.slice(start, end)	Returns a string containing all characters from start through end-1.
string.split(delimiter, limit)	delimiter is a string or regular expression which specifies the places at which to split string. Returns an array of strings. The array is no longer than limit
string.substr(start, length)	Returns a copy of the substring starting from index start and being of the length.
string.substring(from, to)	Returns the substring starting at position from and ending at to-1.
string.toLowerCase()	Returns a copy of string converted to lower case.
string.toUpperCase()	Returns a copy of string with all letters converted to upper case.

The Math Object

The following properties and methods are useful when doing mathematical computations. All angles are measured in radians.

Property / Method	Description
Math.E	Returns Euler's number.
Math.LN10	Returns the natural logarithm of 10.
Math.LN2	Returns the natural logarithm of 2.
Math.LOG10E	Returns the base-10 logarithm of E.
Math.LOG2E	Returns the base-2 logarithm of E.
Math.PI	Returns pi.

Property / Method	Description
Math.SQRT1_2	Returns the square root of 1/2.
Math.SQRT2	Returns the square root of 2.
Math.abs(x)	Returns the absolute value.
Math.acos(x)	Computes arc cosine.
Math.asin(x)	Returns arc sine.
Math.atan(x)	Computes arc tangent.
Math.atan2(y, x)	Computes the angle to a point. The input represent the usual (x,y)-coordinates, but the order has been reversed.
Math.ceil(x)	Rounds up a number.
Math.cos(x)	The cosine function.
Math.exp(x)	Takes e to the power of x.
Math.floor(x)	Rounds down a number.
Math.log(x)	Computes the natural logarithm.
Math.max(x1, x2, ...)	Returns the largest of the numbers.
Math.min(x1, x2, ...)	Returns the smallest of the numbers.
Math.pow(x,y)	Computes x to the power of y
Math.random()	Returns a random number between 0 and 1
Math.round(x)	Rounds to nearest integer.
Math.sin(x)	The sine function.
Math.sqrt(x)	Computes the square root.
Math.tan(x)	The tangent function.

Numbers

It is possible to convert from a Number to a String using String(number) and vice versa using Number(string). Here are some of the useful methods of the Number object.

Method	Description
number.toExponential(digits)	Specifies the number of digits that will occur after the decimal point. Returns a string representation of number in exponential notation.
number.toFixed(digits)	Specifies the number of digits that will occur after the decimal point. Returns a string representation of number that does not use exponential notation.
number.toPrecision(precision)	Specifies the number of significant digits. Returns a string representation of number with the specified number of significant digits.
number.toString(base)	Returns a string representation of the number using the specified base.

Convert Using List

This data converter converts the input text to an output text using a list of conversions. The data converter is useful when converting from the texts used on a particular website to the texts used here, or vice versa. For example, the data converter can be used to convert between country names and country codes.

Specifying the list of conversions

The list of conversions is specified in the Conversions field. For example, a list of conversions from country name to country code could look like this:

```
    "Australia" = "AUS"
    "Austria" = "AUT"
    "Belgium" = "BEL"
    "Brazil" = "BRA"
    "Canada" = "CAN"
    "China" = "CHN"
    "Denmark" = "DNK"
    "Egypt" = "EGY"
    "Finland" = "FIN"
    "France" = "FRA"
    "Germany" = "DEU"
    "Hungary" = "HUN"
    "Iceland" = "ISL"
    "India" = "IND"
    "Ireland" = "IRL"
    ...
```

With this list of conversions, the input text "Australia" will be converted to "AUS", the input text "Austria" will be converted to "AUT", etc.

The backslash character (\) can be used to enter special characters in the texts:

- \n for line break.
- \r for carriage return.
- \f for form document.
- \t for horizontal tab.
- \b for backspace.
- \" for double quote.
- \' for single quote.
- \\ for backslash itself.
- \uxxxx for the unicode character with encoding xxxx, where xxxx is four hexadecimal digits.

Quotes around a text can be omitted. In that case, all spaces at the start and end of the text will be removed, the text cannot be empty, and the backslash notation can not be used to enter special characters.

The list of conversions can contain empty lines and comment lines. A comment line starts with two forward slash characters (//).

Handling the case where no conversion matches

The If No Matching Conversion option determines what happens in the case where the input text does not match any of the conversions:

- *Generate Error* will cause an error to be generated by the data converter. The error will be handled according to the configuration of the step that uses the data converter.
- *Do Not Convert Text* will prevent the input text from being converted, i.e. the input text will be used as output text without any conversion.
- *Convert to Default Text* will cause the input text to be converted to the text specified in the Default Text field.

Note: The text in the Default Text field is specified *without* quotes. If there needs to be a quote character inside the text, it can be entered directly (do not use the \" notation).

Other Properties

The Convert Using List data converter can additionally be configured using the following properties:

Description

Type in a description to be shown in the list of data converters. If there is no type in a description, one will be generated.

Count Tags

The Count Tags data converter counts the number of times that a specific tag appears in the input text. It matches the tag name exactly.

Properties

The Count Tags data converter can be configured using the following properties:

Tag Name

In this field, you enter the name of the tag that you want to count, e.g. "tr".

Require End Tag

If this is checked, only the tags that have both start and end tags are counted.

Description

Type in a description to be shown in the list of data converters. If there is no type in a description, one will be generated.

Evaluate Expression

This data converter provides an opportunity to evaluate an expression.

The input text to the data converter can be used in the expression by using the INPUT notation, see the examples below.

Properties

The Evaluate Expression data converter is configured using the following properties:

Expression

Contains an expression whose result becomes the output of the data converter. The expression can refer to the input text by using the INPUT notation, see the examples later in this topic.

Description

Type in a description to be shown in the list of data converters. If there is no type in a description, one will be generated.

Examples

Get the current date and time can happen with this expression:

```
now()
```

If the input text is "The time is " and the current time should be added, this expression can be used:

```
INPUT + time(now())
```

Numeric calculations can also be performed. For example, if the integer attribute Item.price contains 95 and the number attribute Customer.discount contain 25.0 then the discount can be computed:

```
(Item.price * Customer.discount)/100
```

Likewise, if the input text is "10", it can be multiplied by 20 by using this expression:

```
 toInteger(INPUT) + 20
```

Extract

This data converter provides a possibility to extract a text piece from the input text in a simple way using a pattern. The piece to extract should be marked with a pair of parentheses (i.e. the extracted text piece is the first submatch in the pattern).

For more advanced extraction options, e.g. to extract more than one piece of text or to have manipulation options, use the Advanced Extract data converter instead.

Properties

The Extract data converter is configured using the following properties:

Pattern

Contains a pattern that is matched against the input text. The part of the input text to extract should be marked with a pair of parentheses. Note that the pattern must match the entire input text. Otherwise, the converter will fail, producing an error.

Ignore Case

If this option is selected, the pattern matching is case-insensitive, i.e. the pattern is matched against the input text without regard for character case.

Description

Type in a description to be shown in the list of data converters. If there is no type in a description, one will be generated.

Example

If the first word in the input text should be extracted, use the pattern

```
    (\S*)\s?.*
```

This extracts the first sequence of non-whitespace characters.

Extract Date

This data converter finds and extracts a date. The extracted date is output in the standard date format.

Note: If a date that is already in the standard date format should be reformatted, use the Format Date data converter instead.

Properties

The Extract Date data converter is configured using the following properties:

Basic
Formats

The date formats, in the order that they should be tried. The first date format that matches the input will be applied. If none match, the data converter will generate an error. Click the '+' sign at the top of the list to add a new date format. The data converter supports two kinds of date formats: Format patterns and relative dates. Format patterns allow specification of a date using patterns like MM/dd yyyy hh:mm. Missing month, date or year fields will be taken relative to today's date, depending on whether the date is expected to belong to the past or the future - see the description of the Direction in time property. A format pattern has the following properties:

Pattern

A pattern that specifies the format of the date to be extracted. See the Syntax of the Relative Date Pattern section later in this topic.

Locale

Specifies the locale that is used in the input. This is for instance used if the input contains the names of months or weekdays, as in 'Monday, 25 May 2009'.

Default Date

This option can be used to specify a date other than the current date to resolve incomplete dates. By default the option is set to Current Date.

Advanced

This tab contains options for specifying a future or a past date. The date that the input should be understood relative to. By default, this is the expression now(), which yields the current date and time.

Direction in time

Specify whether the date to be extracted is a past or future date. This allows the data converter to fill out the missing information if the month and/or year is missing from the format pattern, or when extracting from a relative date. For instance, if extracting from '3 hours ago', the direction in time should be set to 'Past date' in order for the 3 hours to be subtracted from the date specified in 'Relative to', while the direction in time should be set to 'Future date' if you are extracting from input like 'in 5 days'.

Default Time Zone

The default time zone of the date in the input text. If no time zone is selected, no default time zone is used. If a time zone is selected, this time zone is used when no time zone is found in the input text.

Result Time Zone

The time zone to which the date should be converted. If no time zone is selected, no conversion will be done. If a time zone is selected, the date found in the input text will be converted from its time zone (or the default time zone; see above) to this time zone. If the date in the input text has no time zone and no default time zone is selected, no conversion is done.

Constants

Specify language-dependent constants for extracting relative dates where some numbers may be written out rather than specified with numbers. For instance, to be able to extract a date from the input 'Updated an hour ago', a relative date format with the pattern 'HOURS hour[s] ago' must be specified and it is important to make sure that the constant 'an = 1' is defined.

Description

Type in a description to be shown in the list of data converters. If there is no type in a description, one will be generated.

Syntax of the Format Pattern

The following patterns can be combined to create the pattern in the Pattern property:

Pattern	Description
yy	Exactly two digit year
yyy	Any year
yyyy	Exactly four digit year
YYYY[2]	Exactly four digit week year
G	Era marker (AD or BC)
MM	One or two digit month, abbreviations or full names of months
dd	One or two digit date
EEE	Short weekday name (for example, Mon instead of Monday).
EEEE	Full weekday name (that is Monday, Tuesday, etc).
hh or HH	One or two digit hour
mm	One or two digit minute
ss	One or two digit second
a	AM or PM marker

[2] When using WEEK YEAR, the year of the week that includes the first day of the year is set as the new year.

Pattern	Description
Z	Time Zone identifier (e.g. "PST", "Central European Time" or "GMT+02:00")
*	Skip any number of characters
Space	Skip one or more white spaces
Any other character	Skip that exact character

If the weekday patterns ('EEE' and 'EEEE') are used and the date pattern ('dd') is *not* used, then the month and year patterns ('MM' and 'yy'/'yyy'/'yyyy') cannot be used. In this case, the date found is the next day with a name matching the pattern. For example, if the pattern is 'EEEE' and the input is the following text:

```
Wednesday
```

the date found is the next Wednesday.

If the weekday patterns are used together with the date pattern (and possibly the month and year patterns), the weekday is discarded. For example, if the pattern is 'EEE, dd/MM/yyy' and the input is the following text:

```
Mon, 16/03/2003
```

the date found is '2003-03-16 00:00:00.0' (ignoring whether this is a Monday or not).

Note: The 'EEE' pattern matches the short names of the weekdays (e.g. Mon, Tue etc.). If the pattern should match the *entire* weekday name, use the 'EEEE' pattern. For example, if the input is the following text:

```
Thus, let us meet on Wednesday
```

As the pattern 'EEEE' should be used, as the pattern 'EEE' would match 'Thu' causing the Date Extractor to find next Thursday.

Example: Format Patterns and Matching Dates

Format Pattern	Matching Dates
dd/MM-yyy	7/6-78 24/12-2001 1/jan-2001
dd. MM yyy	4. jan 1993 4. january 93
yyyy G	2000 AD

Syntax of Relative Date Pattern

The following date fields can be used in the pattern in the Pattern property of a relative date:

Date Field	Description
`SECONDS`	Seconds
`MINUTES`	Minutes

Date Field	Description
`HOURS`	Hours
`DAYS`	Days
`MONTHS`	Months
`YEARS`	Years

Note that time markers like "ago" are not automatically recognized by the step, therefore to extract a relative date in the past, select **Past date** in the **Direction in time** list on the **Advanced** tab. The robot then subtracts the extracted number from the current time.

To extract the date in the future, select **Future date** in the **Direction in time** list on the **Advanced** tab. The robot then adds the extracted number to the current time.

For example, if you want to get the exact time of the "123 seconds ago" string, specify the following:

- On the **Basic** tab, select `SECONDS sec[s] ago` in the **Pattern** field and `now()` in **Relative To**.
- On the **Advanced** tab, select `Past date` in the **Direction in time** list.

  The step then subtracts 123 seconds from the current time.

Example: Relative Date Patterns

Format Pattern	Matching Dates
`HOURS hour[s] ago`	4 hours ago an hour ago (if constant an = 1 is defined)
`HOURS hour[s] and MINUTES minute[s] ago`	3 hours and 5 minutes ago
`[HOURS hour[s] ]MINUTES minute[s] ago`	4 hours 1 minute ago 15 minutes ago

Extract List

The Extract List data converter filters texts according to a pattern, and returns a concatenated text of all matches.

Whereas the Advanced Extract data converter only matches the first instance of the pattern, this data converter will return each possible match in succession.

Note that the pattern must not necessarily match the entire input text, which is a requirement for the Advanced Extract data converter. In fact, most uses of this data converter will utilize patterns that start and end at well-defined points (i.e. the pattern will most likely not need to contain ".*" at the beginning and end of the pattern when performing a match for specific text embedded within a large input text).

Properties

The Extract List data converter can be configured using the following properties:

Pattern

Enter a pattern that is matched against the input.

Ignore Case

If this is checked, the matching against the pattern is done without regard for the character case, e.g. "KoFaX" is considered equivalent to "KOFAX" and "kofax".

Output Expression

Enter an expression that specifies the output text.

Output Delimiter

Enter an optional text to indicate the delimiter that should be used to separate successive pattern matches within the output text.

Description

Type in a description to be shown in the list of data converters. If there is no type in a description, one will be generated.

Extract Number

This data converter finds and extracts a number and outputs it in the standard number format.

Note: If a number that is already in the standard number format should be reformatted, use the Format Number data converter instead.

Properties

The Extract Number data converter is configured using the following properties:

Format Pattern

Contains a pattern that specifies the format of the number to be extracted. Either use one of the default patterns, or see below for details about specifying a pattern.

Decimal Separator

Contains the possible decimal separators in the number to be extracted, e.g. ".". More than one separator can be specified.

Thousands Separator

Contains the possible thousands separators in the number to be extracted, e.g. ",". More than one separator can be specified.

Minus Sign

Contains the character to use as minus sign in the number, typically '-'.

Multiply By

Specifies a multiplication factor that will be multiplied to the extracted number.

Convert to Integer

If this field is checked, the extracted number will be converted to an integer.

Constants

Contains definitions of *constants* which may occur before or after the number to be extracted. For each constant, the name (e.g. kilo) and the value (e.g. 1000) can be given as well as the position of the constant (before and/or after the number to be extracted). Note that the name of a constant must be precisely what comes before or after the number to be extracted. For instance, let the constants configured be kilo=1000.0 and double=2.0. From the input "2 kilo", the number 2000.0 will be extracted, but from the input "2 double kilo", only the number 2.0 will be extracted because no constants are named double kilo.

Description

Type in a description to be shown in the list of data converters. If there is no type in a description, one will be generated.

Specifying a format pattern

The format pattern provides a very flexible way of specifying the number format. However, the rules for specifying the pattern can be somewhat difficult to understand, so finding the default pattern that matches the required format in the best possible way, and then experimenting with changing that default pattern might be an easier solution.

In a pattern, the following special characters can be used:

Special Character	Description
0	A digit.
#	A digit, but zero is not shown.
.	The decimal separator, i.e. the character specified in the Decimal Separator field.
,	The thousands separator, i.e. the character specified in the Thousands Separator field.
-	The minus sign, i.e. the character specified in the Minus Sign field.
E	In scientific notation, separates the mantissa and the exponent.

Note In the pattern, the '.' character is always used to select the decimal separator, regardless of what is entered in the Decimal Separator field. The '.' character will then be replaced by the character in the Decimal Separator field when the number is formatted. The same applies to the thousands separator and minus sign.

Separate patterns can be specified for positive and negative numbers. This is done by specifying two patterns separated by semicolon (';'). For example, use the pattern "#,##0.00;(#,##0.00)" if you want negative numbers to be parenthesized instead of the default where the minus sign character is placed in front of negative numbers.

Note If the input uses scientific notation with a large exponent (e.g. the number 6.023E23), Convert to Integer should generally *not* be checked, as conversion of such large numbers to integers may give inappropriate results.

Example: Extracting Numbers

Consider this input:

```
    Price is USD 33,555.77.
```

With Format Pattern set to "###0.0",Decimal Separator set to ".",Thousands Separator set to ",",Minus Sign set to "-",Multiply By set to "1.0",Convert to Integer not checked, and no Constants configured, the number 33555.77 is extracted.

In the example above, if Convert to Integer is checked, the number 33556 is extracted.

Now, consider this input:

```
Price is USD 10.5 mill.
```

With Format Pattern set to "0.000",Decimal Separator set to ".",Thousands Separator set to ",",Minus Sign set to "-",Multiply By set to "1.0",Convert to Integer checked, and Constants set to mill.=1000000.0 and bill.=1000000000.0, the number 10500000 is extracted.

In the example above, if Convert to Integer is not checked, the number 1.05E7 is extracted.

Extract Year

The Extract Year data converter extracts a year from a date in the input text. The date may be incomplete, such as containing only the year.

Properties

The Extract Year data converter is configured using the following properties:

Locale

Specifies the locale that is used in the date.

Date Format Pattern

Contains a pattern that specifies the format of the date from which the year should be extracted. See the syntax description below.

Max. Months Ahead

The maximum number of months to look ahead. This field takes effect only in the absence of an explicit year when using the 'dd' and 'MM' patterns.

Max. Days Ahead

The maximum number of days to look ahead. This field takes effect only in the absence of an explicit month when using the 'dd' and 'yyy' patterns.

Description

Type in a description to be shown in the list of data converters. If there is no type in a description, one will be generated.

Syntax of the Date Format Pattern

The following patterns can be combined to create the pattern in the Date Format Pattern property:

Pattern	Description
yy	Exactly two digit year
yyy	Any year
yyyy	Exactly four digit year

Pattern	Description
YYYY[3]	Exactly four digit week year
MM	One or two digit month, abbreviations or full names of months
dd	One or two digit date
EEE	Short weekday name (e.g. Mon instead of Monday).
EEEE	Full weekday name (e.g. Monday).
hh or HH	One or two digit hour
mm	One or two digit minute
ss	One or two digit second
a	AM or PM marker
Z	Time Zone identifier (e.g. "PST", "Central European Time" or "GMT +02:00").
*	Skip any number of characters (this must be used in order to skip alphanumeric characters a-z and A-Z)
Space	Skip one or more white spaces
Any other non-alphanumeric character	Skip that exact character (use * to skip alphanumeric characters)

If the weekday patterns ('EEE' and 'EEEE') are used and the date pattern ('dd') is *not* used, then the month and year patterns ('MM' and 'yy'/'yyy'/'yyyy') cannot be used. In this case, the date found is the next day with a name matching the pattern. For example, if the pattern is 'EEEE' and the input is the following text:

Wednesday

the year found is the year of the next Wednesday.

If the weekday patterns are used together with the date pattern (and possibly the month and year patterns), the weekday is discarded. For example, if the pattern is 'EEE, dd/MM/yyy' and the input is the following text:

the year found is '2003'.

Note The 'EEE' pattern matches the short names of the weekdays (e.g. Mon, Tue etc.). If the pattern should match the *entire* weekday name, use the 'EEEE' pattern. For example, if the input is the following text:

Thus, let us meet on Wednesday

The pattern 'EEEE' should be used because the pattern 'EEE' would match 'Thu' causing the Extract Year data converter to find the year of the next Thursday.

[3] When using WEEK YEAR, the year of the week that includes the first day of the year is set as the new year.

Example: Date Format Pattern

Here are some examples of format patterns and matching dates:

Date Format Pattern	Matching Dates
dd/MM-yyy	7/6/1978 24/12-2001 1-Jan-01
dd. MM yyy	4. jan 1993 4. january 93

Format Date

This data converter reformats a date. The input text must be a date in the standard date format, e.g. "2001-02-25 14:32:49.0".

Note: If you want to convert a date to the standard date format, use the Extract Date data converter.

Properties

The Format Date data converter is configured using the following properties:

Locale

This field specifies the *locale* that the date is to be formatted to.

Format Pattern

Contains a pattern that specifies the format of the date. Either use one of the default patterns, or see below for details about specifying a pattern.

Description

Type in a description to be shown in the list of data converters. If there is no type in a description, one will be generated.

Specifying a format pattern

The format pattern provides a very flexible way of specifying the date format. However, the rules for specifying the pattern can be somewhat difficult to understand, so it might be easier to simply find the default pattern that matches the required format best, and then experiment with changing that default pattern.

In a pattern, the following special characters can be used:

Special Character	Description	Type	Example
y,yy,yyy	year	Number	96
yyyy	year	Number	1996
YYYY[4]	week year	Number	2009

[4] When using WEEK YEAR, the year of the week that includes the first day of the year is set as the new year.

Special Character	Description	Type	Example
M	month	Number	7
MM	month	Number	7
MMM	month	Text	Jul
MMMM	month	Text	July
d,dd	day in month	Number	10
(d)ddd	day in month	Number	(0)010
E,EE,EEE	day in week	Text	Tue
EEEE	day in week	Text	Tuesday
D,DD,(D)DDD	day in year	Number	(0)189
(F)F	day of week in month	Number	(0)2 (2nd Wed in July)
w,(w)ww	week in year	Number	(0)27
(W)W	week in month	Number	(0)2
(H)H	hour in day (0-23)	Number	(0)4 or (0)12
(k)k	hour in day (1-24)	Number	(0)4 or (0)12
(K)K	hour in am/pm (0-11)	Number	(0)0
(h)h	hour in am/pm (1-12)	Number	(0)5 or (0)12
(m)m	minute in hour	Number	(0)30
(s)s	second in minute	Number	(0)55
S,SS,(S)SSS	millisecond	Number	(0)978
a	am/pm marker	Text	PM
z,zz,zzz	time zone	Text	PST
(z)zzzz	time zone	Text	Pacific Standard Time
G	era designator	Text	AD
'	start/end of text not in input	Delimiter	'o''clock' -> o'clock
''	single quote	Literal	"EEEE" -> "Friday"

Any characters in the pattern that are not letters in the range A-Z or a-z will be treated as text. For instance, characters like ':', '.', ' ', '#' and '@' will appear in the resulting date even though they are not enclosed in single quotes.

If the pattern is empty, the default date format for the selected locale is used.

Example: Output Dates

Examples of output dates if the locale is "English (United States)":

Format Pattern	Example of output data
yyyy.MM.dd G 'at' hh:mm:ss z	1996.07.10 AD at 15:08:56 PDT

Format Pattern	Example of output data
EEE, MMM d, ''yy	Wed, July 10, '96
h:mm a	12:08 PM
hh 'o''clock' a, zzzz	12 o'clock PM, Pacific Daylight Time
K:mm a, z	0:00 PM, PST
yyyyy.MMMMM.dd GGG hh:mm aaa	1996.July.10 AD 12:08 PM

Format HTML

This data converter reformats (pretty-prints) the input HTML text.

Properties

The Format HTML data converter can be configured using the following properties.

Allow Missing End Tags

If selected, tags whose end tags are optional (e.g. <p>-tags) will be ended automatically. Selecting this option is not necessary for HTML outputted directly from a step action.

Description

Type in a description to be shown in the list of data converters. If there is no type in a description, one will be generated.

Format Number

This data converter reformats a number. The input text must be a number in the standard number format, e.g. "12378.64".

Note: If a number should be converted to the standard number format, use the Extract Number data converter.

Properties

The Format Number data converter is configured using the following properties:

Format Pattern

Contains a pattern that specifies the format of the number. Use one of the default patterns, or see below for details about specifying a pattern.

Decimal Separator

Contains the *decimal separator* to use in the number, i.e. the separator between the integer and fraction part of the number, for example '.' or ','.

Thousands Separator

Contains the *thousands separator* to use in the number, i.e. the separator between groups of thousands in the integer part of the number, for example ',' or a space.

Minus Sign

Contains the character to use as minus sign in the number, typically '-'.

Description

Type in a description to be shown in the list of data converters. If there is no type in a description, one will be generated.

Specifying a format pattern

The format pattern provides a very flexible way of specifying the number format. However, the rules for specifying the pattern can be somewhat difficult to understand, so simply find the default pattern that matches the required format best, and then experiment with changing that default pattern.

In a pattern, the following special characters can be used:

Special Character	Description
0	A digit.
#	A digit, but zero is not shown.
.	The decimal separator, i.e. the character specified in the Decimal Separator field.
,	The thousands separator, i.e. the character specified in the Thousands Separator field.
-	The minus sign, i.e. the character specified in the Minus Sign field.
E	In scientific notation, separates the mantissa and the exponent.

Note In the pattern, the '.' character is always used to select the decimal separator, regardless of what have been entered in the Decimal Separator field. The '.' character will then be replaced by the character in the Decimal Separator field when the number is formatted. The same applies to the thousands separator and minus sign.

Separate patterns can be specified for positive and negative numbers. This is done by specifying two patterns separated by semicolon (';'). For example, the pattern "#,##0.00;(#,##0.00)" can be used if negative numbers should be parenthesized instead of the default where the minus sign character is placed in front of negative numbers.

Get Property

This data converter fetches the value of a property from a property list contained in a variable.

The variable must be of the Properties variable type.

The input text to the data converter is ignored.

Properties

The Get Property data converter is configured using the following properties:

Properties Variable

The variable containing the list of properties.

Property Name

The name of the property to get.

Use Default Value If No Property

Determines what to do if the property does not exist. If this option is selected, a default value is used instead; otherwise an error is generated.

Default Value

The default value to use if the property does not exist, and a default value should be used instead.

Description

Type in a description to be shown in the list of data converters. If there is no type in a description, one will be generated.

Get Time Between Dates

This data converter provides the possibility to find the difference between two dates. It compares the date in the input text to a given date and calculates the difference.

The difference is measured in a selected unit, e.g. days or weeks. The input text must be a date in the standard date format, e.g. "2001-02-25 14:32:49.0".

Properties

The Get Time Between Dates data converter is configured using the following properties:

Other Date

Specify the date to compare the input date with. The date can be specified in several ways using the Value Selector. The date must be in the standard date format, e.g. "2001-02-25 14:32:49.0".

Get Difference as

Select what unit to get the difference in.

Get Integer Difference

Determines whether the difference should be rounded to an integer.

Get Signed Difference

Determines whether the difference should be with or without sign. If the difference should be signed, it will be positive if the input date is after the other date, and negative in the opposite case.

Description

Type in a description to be shown in the list of data converters. If there is no type in a description, one will be generated.

Example

If the input to the converter is "2008-03-01 12:00:00.0", and the other date is set to "2008-02-28 00:00:00.0", and the difference should be in days and with fraction, the result will be 2.5. Notice how the converter accounts for the leap year.

Get Variable

This data converter fetches the value of a variable. The input text is ignored.

Properties

The Get Variable data converter is configured using the following properties:

Variable

The variable whose value to get.

Description

Type in a description to be shown in the list of data converters. If there is no type in a description, one will be generated.

If Then

The If Then data converter allows for specification of a list of if-then rules that determine the output of the converter.

The list can consist of multiple if conditions and has always one else block at the bottom, which provides the default value.

Basic Conditions

If Contains, If Does Not Contain, If Starts With, and If Ends With condition types provide a check into whether the input string contains, does not contain, starts with or ends with the given string accordingly.

If Matches Pattern and If Does Not Match Pattern condition types provide a check into whether the input string matches or does not match a pattern.

Properties for Basic Conditions

The basic conditions of the If Then data converter can be configured using the following properties.

If Contains

If Does Not Contain

If Starts With

If Ends With

A text value is entered which is matched against the input text.

If Matches Pattern
If Does Not Match Pattern

In these fields a pattern is entered which is matched against the input text. Note that the entire input text must match / not match the pattern.

Then

Specifies the output text if the value of the property above matches the input text. The value can be specified in several ways using the Value Selector (without converters).

Ignore Case

If this is checked, the matching against the value of the first property is done without regard to the character case, e.g. "KoFaX" is considered equivalent to "KOFAX" and "kofax".

Properties for Else

The else statement of the If Then data converter can be configured using the following property.

Then

Specifies the output text if no conditions matched the input text. The value can be specified in several ways using the Value Selector (without converters). If this field is left blank, then the If Then converter returns an empty text.

In the If Matches Pattern the expression in the Then attribute can refer to submatches of the pattern in the preceding If Matches Pattern field using the $n notation.

In all the other conditions the INPUT keyword can be used to refer to the input text.

Other Properties

The If Then data converter can additionally be configured using the following properties:

Description

Type in a description to be shown in the list of data converters. If there is no type in a description, one will be generated.

Examples

Let us assume that the input text is "911" and we want the output text to be "Porsche 911". Alternatively, if the input text is anything other than "911", it should remain as is.

The If Then data converter should then be configured as follows:

- If Matches
  - If Matches: 911
  - Then (expression): "Porsche " + $0
  - Ignore Case: [unchecked]
- Else
  - Then (expression): $0

Make URL Absolute

The Make URL Absolute data converter converts a relative URL to an absolute one using the current URL of the robot.

Properties

The Make URL Absolute data converter can be configured using the following properties:

Description

Type in a description to be shown in the list of data converters. If there is no type in a description, one will be generated.

Examples

In these examples, the current URL of the robot is http://www.kofax.com

- If the input text is '~hello', then the output text becomes 'http://www.kofax.com/~hello'
- If the input text is 'hello', then the output text becomes 'http://www.kofax.com/hello'
- If the input text is 'test1/test2/../test', then the output text becomes 'http://www.kofax.com/test1/test'

Make URL Relative

The Make URL Relative data converter converts an absolute URL to a relative one using the current URL of the robot.

Properties

The Make URL Relative data converter can be configured using the following properties:

Base URL

The URL to make the input URL relative to.

Description

Type in a description to be shown in the list of data converters. If there is no type in a description, one will be generated.

Examples

In these examples, the current URL of the robot is http://www.kofax.com

- If the input text is 'http://www.kofax.com/~hello', then the output text becomes '~hello'
- If the input text is 'http://www.kofax.com/hello', then the output text becomes 'hello'
- If the input text is 'http://www.kofax.com/test1/test2/../test', then the output text becomes 'test1/test'

Modify Date

This data converter modifies a date by adding or subtracting from a selected part of the date.

If the addition/subtraction causes the selected part of the date to overflow or underflow, the other parts of the date will be updated accordingly.

The input text must be a date in the standard date format, e.g. "2001-02-25 14:32:49.0".

Properties

Amount

The amount to add or subtract from the date. The amount is specified using the Value Selector. The value must be an integer.

Part of Input Date to Modify

Select which part of the date to add or subtract from.

Function

Choose if the amount should be added or subtracted.

Time Zone

The time zone of the input date.

Description

Type in a description to be shown in the list of data converters. If there is no type in a description, one will be generated.

Example

If the input to the converter is "2008-02-28 10:45:00.0" and two days should be added, the result will be "2008-03-01 10:45:00.0". Notice how the month was updated and the leap day in 2008 was taken into consideration.

Remove Non-Printable Characters

The Remove Non-Printable Characters data converter removes all non-printable characters.

More specifically, the following characters are removed:

- All characters below ASCII 32, except tab (#x0009), line feed (#x000A), and carriage return (#x000D).
- All characters in the intervals #xD800-#xDFFF and #xFFFE-#xFFFF.

Properties

The Remove Non-Printable Characters data converter can be configured using the following properties:

Description

Type in a description to be shown in the list of data converters. If there is no type in a description, one will be generated.

Remove Spaces

This data converter removes spaces from the input text. Notice that non-breakable spaces (occur as &nbsp; in the HTML) are treated as spaces.

Properties

The Remove Spaces data converter can be configured using the following properties.

Remove All Spaces

Removes all spaces from the input text.

Remove Start and End Spaces

Trims the text so that there are no spaces at its start and end.

Replace Multiple Spaces with Single Spaces

Replaces all occurrences of multiple spaces within the input text with single spaces.

Description

Type in a description to be shown in the list of data converters. If there is no type in a description, one will be generated.

Example

If the input text is:

```
    "   hello   world    "
```

and Replace Multiple Spaces with Single Spaces is selected, then the output text becomes:

```
    " hello world "
```

With Remove Start and End Spaces selected, the output text becomes:

```
    "hello   world"
```

With both Replace Multiple Spaces with Single Spaces and Remove Start and End Spaces selected, the output text becomes:

```
    "hello world"
```

And with Remove All Spaces selected, the output text becomes:

```
    "helloworld"
```

Remove Special Characters

The Remove Special Characters data converter replaces all special characters with spaces in the input text. Any character that is not a letter, a digit, or a comma/point appearing before a digit, is considered to be a special character and is replaced by a space.

After applying the Remove Special Characters data converter, the Remove Spaces data converter can be used to remove undesired spaces.

Properties

The Remove Special Characters data converter can be configured using the following properties:

Description

Type in a description to be shown in the list of data converters. If there is no type in a description, one will be generated.

Remove Tags

The Remove Tags data converter removes HTML tags from the input text.

Properties

The Remove Tags data converter can be configured using the following properties.

Remove These Tags

Specifies the tags that should be removed.

- "All Tags" specifies that all tags should be removed. Optionally keep ampersand encodings in the text.
- "Selected Tags" specifies that only the selected tags should be removed. The tags names are separated using commas, e.g. "html,body". Optionally keep ampersand encodings in the text.

Description

Type in a description to be shown in the list of data converters. If there is no type in a description, one will be generated.

Replace Pattern

The Replace Pattern data converter replaces matches of a pattern with the result of an expression.

Properties

The Replace Pattern data converter can be configured using the following properties.

Pattern

Specifies a pattern to search for in the input text. Note that this pattern does not have to match the entire input text.

Ignore Case

If this is checked, then the pattern matching is case-insensitive.

Replace Expression

Specifies an expression, whose result will replace the part of the text matched by the pattern.

Replace All

If this is checked, then all occurrences of the Pattern will be replaced by the Replace Expression. If not, then only the first occurrence is replaced.

Description

Type in a description to be shown in the list of data converters. If there is no type in a description, one will be generated.

Replace Text

This data converter finds and replaces matching text in the input text.

Properties

The Replace Text data converter can be configured using the following properties.

Find this Text

The text to search for in the input text.

Replace with this Text

The text to replace with.

Ignore Case

If this is checked, then the text matching is case-insensitive.

Replace All

If this is checked, then all occurrences of the text will be replaced by the new text. If not, then only the first occurrence is replaced.

Match Whole Words Only

If this is checked, then text replacement will only take place for occurrences that are whole words and not parts of larger words.

Description

Type in a description to be shown in the list of data converters. If there is no type in a description, one will be generated.

Unquote Text

This data converter unquotes text that is enclosed in single or double quotes. Escaped quotes inside the text are unescaped.

Properties

The Unquote Text data converter can be configured using the following properties.

Description

Type in a description to be shown in the list of data converters. If there is no description, one will be generated.

Examples

If the input text is:

```
    "Bob"
```

then the output text becomes:

```
    Bob
```

If the input text is:

```
    "Robert \"Bob\" Jones"
```

then the output text becomes:

```
    Robert "Bob" Jones
```

If the input text is:

```
    'Bob'
```

then the output text becomes:

```
    Bob
```

If the input text is:

```
    Bob
```

then the output text becomes:

```
    Bob
```

URL Decode

This data converter decodes all URL encodings into their real characters.

The encoded characters are on the form %HH, where HH is the hexadecimal byte value. The encoded characters are first decoded to bytes from this notation. The bytes are then converted to characters using the selected character encoding.

Properties

The URL Decode data converter can be configured using the following properties:

Character Encoding

The character encoding to use for converting the bytes to characters.

Description

Type in a description to be shown in the list of data converters. If there is no type in a description, one will be generated.

Example

If the selected character encoding is UTF-8, and the input text is:

```
    x%2By%3Dz
```

the data converter will output:

```
    x+y=z
```

URL Encode

This data converter encodes characters with URL encodings.

The characters that need to be encoded are first converted to bytes using the selected character encoding, and the bytes are then represented on the form %HH, where HH is the hexadecimal byte value.

Properties

The URL Encode data converter can be configured using the following properties:

Character Encoding

The character encoding to use for converting the characters to bytes.

Description

Type in a description to be shown in the list of data converters. If there is no type in a description, one will be generated.

Example

If the selected encoding is UTF-8, and the input text is:

```
    x+y=z
```

the data converter will output:

```
    x%2By%3Dz
```

Patterns

A pattern is a formal way of describing a text. For example, the text "32" can be described as a text containing two digits. However, other texts also contain two digits, such as "12" and "00" and can therefore also be described as a text containing two digits. We can express this by the pattern \d\d, which is a formal way of expressing that a text must contain two and only two digits (\d is the symbol for a digit). We say that these texts match this pattern. Design Studio patterns follow the Perl5 syntax.

A pattern is composed of normal characters and special symbols. Each special symbol carries its own special meaning. For example, the special symbol "." (dot) means any single character and matches all single characters, such as "a", "b", "1", "2", ...

The table below provides an overview of the most commonly used special symbols.

Special symbol	Description
.	Any single character, such as "a", "1", "/", "?", ".", etc.
\d	Any decimal digit, such as "0", "1", ..., "9".
\D	Any non-digit, that is the same as ".", but excluding "0", "1", ..., "9".
\s	Any white space character, such as " " and line break.
\S	Any non-white space character, i.e. same as ".", but excluding white space (such as " " and line break).
\w	Any word (alphanumeric) character, such as "a", ..., "z", "A", ..., "Z", "0", ..., "9".
\W	Any non-word (alphanumeric) character, i.e. same as ".", but excluding "a", ..., "z", "A", ..., "Z", "0", ..., "9".

Examples

- The pattern ".an" matches all text lengths of three ending with "an", such as "can" and "man" but not "mcan".
- The pattern "\d\d\s\d\d" matches all text lengths five starting with two digits followed by a white space and ending with two digits, such as "01 23" and "72 13" but not "01 2s".
- If you want a special character, such as "." or "\", to act as a normal character, you can escape it by adding a "\" (backslash) in front of it. If you wish to match exactly the "." character, instead of any single character, you should write "\.".

  For example, the pattern "m\.n\\o" only matches the text "m.n\o".
- You can organize a pattern into subpatterns by the use of parentheses: "(" and ")".

  For example, the pattern "abc" can be organized as "(a)(bc)".

- All single characters are considered subpatterns.

  For example, in the pattern "abc", each single character "a", "b", and "c" is considered a subpattern.

Subpatterns are useful when applying pattern operators. The following table provides an overview of the available pattern operators.

Operator	Description
?	Matches the preceding subpattern, or the empty text.
*	Matches any number of repetitions of the preceding subpattern, or the empty text.
+	Matches one or more repetitions of the preceding subpattern.
{m}	Matches exactly m repetitions of the preceding subpattern.
{m,n}	Matches between m and n repetitions (inclusive) of the preceding subpattern.
{m,}	Matches m or more repetitions of the preceding subpattern.
a|b	Matches whatever the expression a would match, or whatever the expression b would match.

Examples

- ".*" matches any text, such as "Design Studio", "1213" and "" (the empty text)
- "(abc)*" matches any number of repetitions of the text "abc", such as "", "abc", "abcabc", and "abcabcabc", but not "abca"
- "(\d\d){1,2}" matches either two or four digits, such as "12" and "6789", but not "123"
- "(good)?bye" matches "goodbye" and "bye"
- "(good)|(bye)" matches "good" and "bye"

As with other special characters, you can escape the special characters that appear in pattern operators by adding a "\" backslash in front of the character.

Subpatterns are useful when you want to extract specific text pieces from a text. When you make a subpattern using parentheses, you can extract the part of the text that is matched by that subpattern. For example, consider the pattern "abc (.*) def (.*) ghi". This pattern has two subpatterns that are made by means of parentheses. If the pattern is matched against the text "abc 123 def 456 ghi", the first of those subpatterns will match the text "123", and the second subpattern will match the text "456". In an expression (see Expressions), you can refer to these subpattern matches by writing "$1" and "$2". For example, the expression "X" + $1 + "Y"+ $2 + "Z" will produce the result "X123Y456Z". This is a very important extraction technique in Design Studio, which is described in Experiment with Expressions.

By default, the repetition pattern operators (*, +, {...}) will match as many repetitions of the preceding pattern as possible. You can put a "?" after the operator to turn it into an operator that matches as few repetitions as possible. For example, consider the pattern ".*(\d\d\d).*". If the pattern is matched against the text "abc 123 def 456 ghi", the subpattern "(\d\d\d)" will match the second number in the text ("456"), as the first *-operator will match as many repetitions as possible. If you put a "?" after the *-operator, so that the pattern becomes ".*?(\d\d\d).*", the subpattern "(\d\d\d)" will match the first number in the text ("123"), as the *?-operator will match as few repetitions as possible.

We recommend that you experiment with patterns on your own. The best way to do this is to launch Design Studio and find a place where you can enter a pattern, such as in the Test Tag action. Then, click the **Edit** button to the right of the pattern field, to open the Pattern Editor window.

In the Pattern Editor you can enter a pattern and test whether it matches the test input text in the Input panel. When you open the window, Design Studio usually sets the test input text to the text that the pattern is matched against if the given step is executed on the current input robot state. However, you can also edit the test input text yourself, to try the pattern on other inputs. To test the pattern, click the **Test** button. The result of the matching appears in the Output panel.

The Symbol button in the Patter Editor is very useful when you want to enter a special symbol in the pattern. When you click it, a menu is shown, from which you can select the symbol to insert in the pattern. This way, you do not have to memorize all the special symbols and their meanings.

For more on the available special symbols and patterns, see Patterns.

Expressions

An expression typically evaluates to a text. For example, the expression

"The author of the book " + Book.title + " is " + Book.author + "." evaluates to the text "The author of the book Gone with the Wind is Margaret Mitchell.", if the variables Book.title and Book.author contain the texts "Gone with the Wind" and "Margaret Mitchell", respectively.

You can also do numeric calculations within the expression. For example, if the variable Book.price contains the price of a book, you can multiply it by 100 using the following expression:

Book.price * 100

The following table provides an overview of the most commonly used sub-expression types. For a complete overview of all available sub-expression types, see the reference below.

Commonly Used Sub-Expression Types

Sub-Expression Type	Notation	Description
Text Constant	"text" or >>text<<	Evaluates to the specified text, e.g. "Margaret Mitchell", or >>Margaret Mitchell<<.
Variables	variablename.attributename	Evaluates to the value of the specified variable, e.g. "Book.author" might evaluate to "Margaret Mitchell".
Current URL	URL	Evaluates to the URL of the current page.
Subpattern Match	$n	Evaluates to the text matched by subpattern in an associated pattern (if any). For example, this is used in the Advanced Extract data converter, as shown below. $0 evaluates to the text matched by the entire pattern.
Function	func(args)	Evaluates the specified function by passing it the specified arguments and converting its result to a text.

Note that you can specify a text constant using either the quote notation or the >>text<< notation, for example "Margaret Mitchell" or >>Margaret Mitchell<<. If you use the quote notation, and you want a quote character to appear inside the text, you have to write it as two quote characters. For example, write "This is some ""quoted"" text" to get the text "This is some "quoted" text". If you use the >>text<<

notation, anything can appear inside the text, except ">>" and "<<". Thus, you can write quotes directly, as in >>This is some "quoted" text<<. The >>text<< notation is useful for long texts that contain many quote characters, such as HTML.

The following table shows the most commonly used functions in expressions.

Function	Description
toLowerCase(arg)	Converts the argument to lowercase.
round(arg)	Rounds the argument to the nearest integer.

For example, the expression "The discount is " + round((Item.oldPrice - Item.newPrice) / Item.oldPrice) + "%." evaluates to "The discount is 10%." when the item's old price is $99.95 and the new price is $89.95.

Reference

Expressions

Expression Type	Notation	Description	Examples
Constant	"text"	A fixed text. You can use the backslash character (\) to enter special characters: \n for line break \r for carriage return \f for form document \t for horizontal tab \b for backspace \" for double quote \' for single quote \\ for backslash itself \uxxxx for the unicode character with encoding xxxx, where xxxx is four hexadecimal digits.	"This is some \"quoted\" text." "This a text with line break \n, tab \t and a unicode \u0035 character"
Constant	>>text<<	A fixed text. This notation can contain anything, including quote characters, except the end symbol (<<). The backslash (\) character cannot be used to enter special characters.	>>This is some "quoted" text.<<
Variables	variable.attribute variable	The value of a variable. If the variable is a complex type, an attribute name must also be supplied.	Book.title + Integer
Input Text	INPUT	The input text, if any, to the expression.	INPUT
Concatenation	expr1 + expr2	The concatenation of the two expressions expr1 and expr2.	"Title:" + Book.title

Expression Type	Notation	Description	Examples
Subpattern Match	$n	If n > 0, the text that matches subpattern n in the pattern. If n = 0, the text that matches the entire pattern. Note: This expression has meaning only if there is a pattern associated with the expression.	$1
Numeric Expression	Operators: +, -, *, /, %	The result of a numeric expression.	Book.price * 100
Boolean Expression	Operators: && (and), || (or)	The result of a boolean expression.	performTransactions && Book.price < 30
Conditional Expression	condition ? expr1 : expr2	If condition evaluates to true, the value of expr1 is used; otherwise the value of expr2 is used.	Book.price < 30 ? "cheap" : "expensive"
Equality	Operators: == (equal to), != (not equal to)	These operators work on operands of all types, but the type of the operands must be the same. For example, you cannot compare a Number to an Integer.	true == false style != "none"
Relational	Operators: < (less than), <= (less than or equal to), > (greater than), >= (greater than or equal to)	Relational operators determine if one operand is less than or greater than another operand. The operands must be numbers, that is of type Integer or Number. The types of the operands in an expression must be the same.	0 < 1 1.0 >= 0.0
Function	func(expr)	The function func applied to the result of the expression expr. See the Functions section that follows this table for the available functions.	capitalize(Book.title)
Current URL	URL	The current URL.	URL
Current Window	WINDOW	The unique ID of the current window.	WINDOW
Robot Name	Robot.name	The name of the robot.	Robot.name
Execution ID	Robot.executionId	The current execution ID of the robot.	Robot.executionId

Expression Type	Notation	Description	Examples
Execution Errors	Robot.executionErrors	The execution errors encountered in the previous branch of the nearest Try step. **Note** This expression can be used in the second and all subsequent branches of the Try step.	Robot.executionErrors

Functions

Function	Description
abs(arg)	Returns the absolute value of the number.
base64Decode(arg)	Decodes Base64 encoded data.
base64Encode(arg)	Encodes binary data with Base64 encoding.
binaryToText(data[, encoding])	Decodes binary data into text. Uses the specified encoding if present, or otherwise defaults to UTF-8 encoding.
capitalize(arg)	Makes the first letter of every word upper case and all other letters lower case.
ceil(arg)	Rounds the number up to the nearest integer.
collapseSpaces(arg)	Makes sure there are no two consecutive spaces.
contains(source, key)	Returns whether the source contains the specified key.
date()	Returns the current date in standard date format (yyyy-mm-dd).
day(args)	Returns the day of month of the date given as an argument.
endsWith(source, key)	Returns true if the source string ends with the specified key, or false otherwise.
floor(arg)	Rounds the number down to the nearest integer.
guid()	Returns a randomly generated, globally unique ID (GUID).
hexDecode(arg)	Decodes hex encoded data.
hexEncode(arg)	Encodes binary data with hex encoding.
indexOf(source, key)	Returns the first index of the key in the source, or -1 if not found.
length(arg)	Counts the number of characters in the text, or the number of bytes if given binary data.
max(a, b)	Returns the greater of the two numbers.
md5(arg)	Computes the MD5 checksum of the binary data given as an argument.
min(a, b)	Returns the smallest of the two numbers.
month(args)	Returns the month of the date given as argument.
now()	Returns the current date and time.

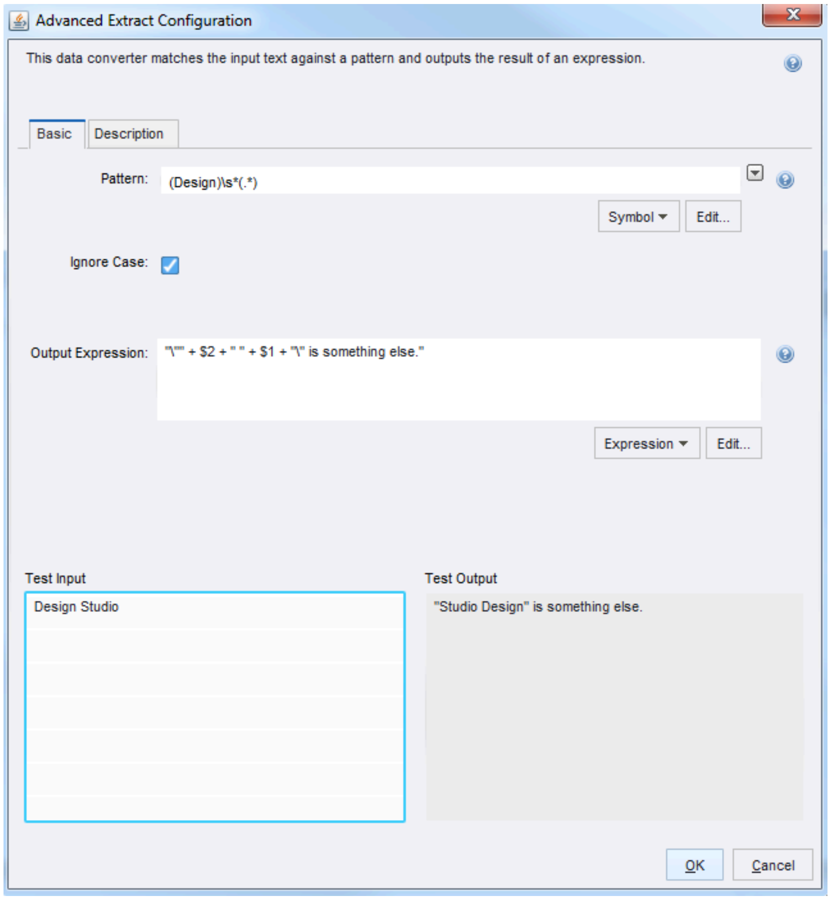
Function	Description
random()	Returns a random number between 0 and 1.
removeSpaces(args)	Removes all white space characters in the argument, such as SPACE, \t, \n.
replacePattern(source, pattern, newText)	Replaces every occurrence of the pattern "pattern" in text "source" with the text "newText". The pattern match is case insensitive.
replaceText(source, oldText, newText)	Replaces every occurrence of the text "oldText" in the text "source" with the text "newText". The match with "oldText" is case insensitive.
resolveURL(arg)	Converts a URL from relative to absolute using the current URL.
round(arg)	Rounds to the nearest integer.
shortTime(arg)	Returns the time without fractional seconds (hh:mm:ss) for the date given as an argument.
substring(source, startIndex)	Returns the portion of the source string starting at the startIndex and up to the end of the source string, with the first character of the string being at index 0.
substring(source, startIndex, endIndex)	Returns the portion of the source string starting at the startIndex and up to the endIndex, with the first character of the string being at index 0.
startsWith(source, key)	Returns true if the source string starts with the specified key, or false otherwise.
textToBinary(text[, encoding])	Encodes text to binary data. Uses the specified encoding if present, or otherwise defaults to UTF-8 encoding.
time(arg)	Returns the time (hh:mm:ss.fff) for the date given as an argument.
toInteger(args)	Converts the text to an integer. This can be useful if you want to include it in calculations.
toNumber(args)	Converts the text to a floating-point number. This can be useful if you want to include it in calculations.
toLowerCase(arg)	Converts all of the characters in the text to lowercase.
toUpperCase(arg)	Converts all of the characters in the text to uppercase.
trim(arg)	Removes all space from both ends of the text.
urlDecode(arg)	Decodes the URL encoded text.
urlEncode(arg)	URL encodes the text.
weekday(arg)	Returns the name of the weekday (in English) for the date given as an argument.
year(args)	Returns the year of the date given as an argument.
toColumn(arg)	Converts integer to Excel column text.
toIndex(arg)	Converts Excel column text to integer.
iteration(arg)	Contains the current iteration number.

Experiment with Expressions

We recommend that you experiment with expressions on your own. The best way to experiment with expressions is to launch Design Studio and open an existing robot.

1. In Design Studio, select the **Extract** action for the current step.
2. Add an Advanced Extract data converter.
3. Click the Configuration icon to configure the data converter.

   The Advanced Extract Configuration Window appears.



This data converter matches the input text against a pattern and outputs the result of an expression.

Pattern: (Design)\s*(.*)

Ignore Case: ☑

Output Expression: "\"" + $2 + " " + $1 + "\" is something else."

Test Input: Design Studio

Test Output: "Studio Design" is something else.

   In the shown example, note the use of the $n notation to extract parts of the input text.

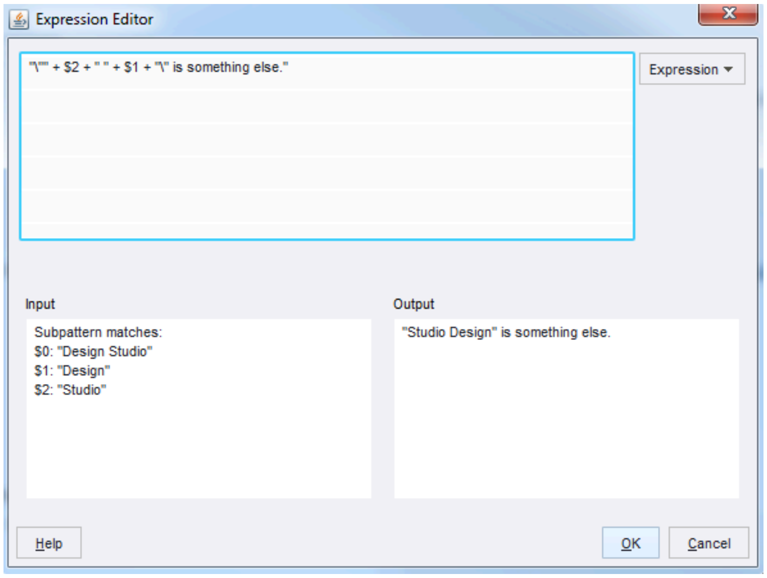
4. Change the input text in the text area to the left.
5. Next, change the Pattern property.
6. Change the Output Expression property.

   Review the results in the right area, while typing the expression.

Edit Expressions

1. On the Advanced Extract Configuration window, in the Output Expression field, click **Edit**.

   The Expression Editor appears.

   ```
   "\"" + $2 + " " + $1 + "\" is something else."
   ```

   Input

   ```
   Subpattern matches:
   $0: "Design Studio"
   $1: "Design"
   $2: "Studio"
   ```

   Output

   ```
   "Studio Design" is something else.
   ```

2. In the Expression field, enter an expression, or click **Expression** to select one from the list. Options include Constant, Variables, Operators, Special Character, Functions, Page Properties, and Robot Properties with additional sub-expression functions.

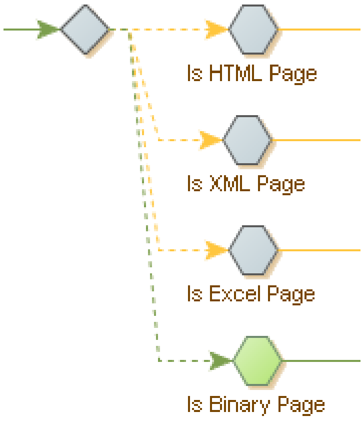
   Expression values appear in the Input and Output sections.

   Note Testing functionality is not available everywhere in Design Studio.

3. Click **OK**.

Determine the Page Type

You can create a Try step to identify the type of loaded page. Valid page types include HTML, XML, Excel, and Binary.



1. In the Designer, after a Load Page step, insert a **Try** step.
2. On the branch, Insert an action step and select **Test** > **Test Page Type**.
3. On the Action tab, Page Type field, select **HTML**.
4. On the Basic tab, change the Step Name to **Is HTML Page**.
5. Select the try step and click **Add Branch**.
6. Repeat 2 through 5, with Page Type set to **XML**, and the Step Name to **Is XML Page**.
7. Repeat 2 through 5, with Page Type set to **Excel**, and the Step Name to **Is Excel Page**.
8. Repeat 2 through 4, with Page Type set to **Binary**, and the Step Name to **Is Binary Page**.

   Once the robot runs, the page type is highlighted.

Use Tag Finders

Use Tag Finders to find a tag on an HTML/XML page. The most common use of a Tag Finder is in a step, where the Tag Finder locates the tag to which you want to apply an action. The list of Tag Finders for the current step is located in the Tag Finders tab of the Step View.

Tag Paths

A tag path is a compact text representation of where a tag is located on a page. Consider this tag path:

```
     html.body.div.a
```

This tag path refers to an <a>-tag inside a <div>-tag inside a <body>-tag inside an <html>-tag.

A tag path can match more than one tag on the same page. For example, the tag path above will match all of the <a>-tags on this page, except the third one:

```
<html>
   <body>
     <div>
       <a href="url...">Link 1</a>
       <a href="url...">Link 2</a>
     </div>
     <p>
       <a href="url...">Link 3</a>
     </p>
     <div>
       <a href="url...">Link 4</a>
       <a href="url...">Link 5</a>
       <a href="url...">Link 6</a>
     </div>
   </body>
 </html>
```

You can use indexes to refer to specific tags among tags of the same type at that level. Consider this tag path:

```
     html.body.div[1].a[0]
```

This tag path refers to the first <a>-tag in the second <div>-tag in a <body>-tag inside an <html>-tag. So, on the page above, this tag path would only match the "Link 4" <a>-tag. Note that indexes in tag paths start from 0. If no index is specified for a given tag on a tag path, the path matches any tag of that type at that level, as we saw in the first tag path above. If the index is negative, the matching tags are counted backwards starting with the last matching tag which corresponds to index -1. Consider this tag path:

```
     html.body.div[-1].a[-2]
```

This tag path refers to the second-to-last <a>-tag in the last <div>-tag in a <body>-tag inside an <html>-tag. So, on the page above, this tag path would only match the "Link 5" <a>-tag.

You can use an asterisk ('*') to mean any number of tags of any type. Consider this tag path:

```
     html.*.table.*.a
```

This tag path refers to an <a>-tag located anywhere inside a <table>-tag, which itself can be located anywhere inside an <html>-tag. There is an implicit asterisk in front of any tag path, so you can simply write "table" instead of "*.table" to refer to any table tag on the page. The only exception is tag paths starting with a punctuation mark ('.'), which means that there is no implicit asterisk in front of the tag path, so the tag path must match from the first (top-level) tag of the page.

With asterisks, you can create tag paths that are more robust against changes in the page, as you can leave out insignificant tags that are liable to change over time, such as layout related tags. However, using asterisks also increases the risk of accidentally locating the wrong tag.

You can provide a list of possible tags by separating them with '|', as in the following tag path:

```
     html.*.p|div|td.a
```

This tag path refers to an <a>-tag inside a <p>-, <div>-, or <td>-tag located anywhere inside an <html>-tag.

In a tag path, text on a page is referred to just as any other tag, using the keyword "text". Although text is not technically a tag, it is treated and viewed as such in a tag path. For example, consider this HTML:

```
<html>
   <body>
     <a href="url...">Link 1</a>
     <a href="url...">Link 2</a>
   </body>
 </html>
```

The tag path "html.body.a[1].text" would refer to the text "Link 2."

Tag Finder Properties

This topic describes the properties to use to configure a Tag Finder.

Find Where

Specifies where to find the tag relative to a named tag. The default value is "Anywhere in Page," meaning that named tags are not used to find the tag.

Tag Path

The tag path as described in Tag Paths.

Attribute Name

The tag must have a specific attribute, for example "align."

Attribute Value

The tag must have an attribute with a specific value. If the Attribute Name property is set, the attribute value is bound to that specific attribute name.

These values are case-sensitive.

- "Equals Text" specifies that the attribute value must match a specified text. Note that the text must match the entire attribute value.
- "Contains Text" specifies that the attribute value must contain a specified text.
- "Starts With Text" specifies that the attribute value must start with a specified text.
- "Ends With Text" specifies that the attribute value must end with a specified text.
- "Matches Pattern" specifies that the attribute value must match a specified pattern. Note that the pattern must match the entire attribute value.
- "Does Not Equal Text" specifies that the attribute value must not be equal to a specified text.
- "Does Not Contain Text" specifies that the attribute value must not contain a specified text.
- "Does Not Start With Text" specifies that the attribute value must not start with a specified text.
- "Does Not End With Text" specifies that the attribute value must not end with a specified text.
- "Does Not Match Pattern" specifies that the attribute value must not match a specified pattern.

Tag Pattern

A pattern that the tag must match (including all tags inside it), for example ".*<b>.*Stock Quotes.*</b>.*". Some caution should be observed in using this property, as it can have considerable impact on the performance of your robot. This is because the Tag Pattern may be applied many times throughout a page just to find the one tag that it matches. One way to try and avoid this is to choose Text Only for the Match Against property.

Match Against

The Tag Pattern should match only the text or the entire HTML of the tag. The default is to match only the text because this is normally much faster.

Tag Depth

Determines which tag to use if matching tags are contained inside each other. The default value is **Any Depth**. This value accepts all matching tags. If you select **Outermost Tag**, only the outermost tags are accepted, and similarly, if you select **Innermost Tag**, only the innermost tags are accepted.

Tag Number

Determines which tag to use if more than one tag matches the tag path and the other criteria. You specify the number of the tag to use, either counting forwards from the first tag or counting backwards from the last tag that matches. For example, if you set the tag path to "table," the Tag Attribute property to "align=center," and the Tag Pattern property to ".*Business News.*," then the Tag Finder would locate the first <table>-tag that is center aligned and that contains the text "Business News."

Configure Tag Finders

There are multiple ways to define the tag path for a tag finder. Design Studio builds the path automatically when you interact with the browser\HTML\DOM path views to create an action or right-click and select "use this tag" or "use only this tag." Alternatively, you can manually define the tag path.

In the Page View, click Tag Finder to see the tag found by the Tag Finder.

Configure Tag Finders using one of the following methods:

- To configure Tag Finders manually, enter details in the Finder tab.
- To configure Tag Finders automatically, select a tag in the Page View and click **Set Selected Node in Finder** . This configures the Tag Finder to find the selected tag using a tag path in simple mode.
- To configure Tag Finders from a context menu, in the Page View, right-click a tag. If you select **Use Tag** from the menu, the Tag Finder is configured to find the right-clicked tag using a tag path in simple mode. Similarly, if you choose another action from the menu, it selects a corresponding step action and configures the Tag Finder to find the right-clicked tag.
- To configure Tag Finders for a step action, select a new step action. Some actions, when selected, configure the Tag Finders so that they find the tags typically used for that action. For example, the Submit Form action uses one Tag Finder and sets its tag path to "form" to locate the first <form>-tag in the page.

1. In the Designer, right-click a node and select **Insert Step** > **Action Step**,
2. Select an Action from the list.
3. On the Actions tab, define attributes based on the Action selected.

   Required attributes are indicated with warning ⚠symbol.

Submit a Form

Submitting a form is a common task in a robot. For example, you may need to submit a search form to get the search results you want to extract, or you may need to submit an order form to make an order transaction. In some cases, you do not need to actually submit the form, but simply want to create a URL that represents the form submission, or modify the current values in the form.

The recommended and simplest way of submitting a form in Design Studio is similar to the way you submit a form in an ordinary browser.

1. Fill in the form details.

   You can use the following actions:

   - Enter Text
   - Enter Password
   - Press Key
   - Select Option
   - Select Multiple Options
   - Set Checkbox
   - Set Range Value
   - Select Radio Button
   - Select File

2. Click the form submission button.

   You can also loop through field values (text input) options or radio buttons by using the following actions:

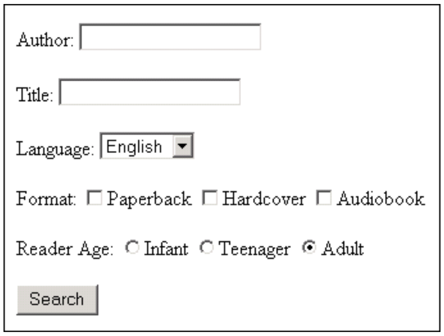
   - Loop Field Values
   - For Each Option
   - For Each Radio Button

Form Basics

Consider the following example of a book search form, first shown as HTML, then as it appears in a browser.

```
<html>
     <body>
       <form action="http://www.books.com/search.asp" method="get">
         Author:
         <input type="text" name="book_author">
         <p>
         Title:
         <input type="text" name="book_title">
         <p>
         Language:
         <select name="book_language">
         <option value="lang_0" selected>English</option>
         <option value="lang_1">French</option>
         <option value="lang_2">German</option>
         <option value="lang_3">Spanish</option>
         </select>
         <p>
         Format:
         <input type="checkbox" name="book_format" value="format_pb">Paperback
         <input type="checkbox" name="book_format" value="format_hc">Hardcover
         <input type="checkbox" name="book_format" value="format_ab">Audiobook
         <p>
         Reader Age:
         <input type="radio" name="reader_age" value="age_inf">Infant
         <input type="radio" name="reader_age" value="age_teen">Teenager
         <input type="radio" name="reader_age" value="age_adult" checked>Adult
         <p>
```

```
      <input type="submit" value="Search">
    </form>
  </body>
</html>
```



Author:

Title:

Language: English

Format: □ Paperback □ Hardcover □ Audiobook

Reader Age: ○ Infant ○ Teenager ● Adult

A form contains a number of fields. For example, the first <input>-tag in the example form defines a field named "book_author". Note that the name of a field is usually different from what the user sees in a browser. For example, the "book_author" field will appear to be named "Author" in the browser, not "book_author".

A field can be defined by more than one tag. For example, the "book_format" field is defined by three <input>-tags in the example form. Tags that use the same field name and are of the same field type (text field, radio button, check box, etc.) define the same field.

A field can be assigned one or more values. For example, the "book_format" field can be assigned the value "format_pb" to select paperback format. Note that, like the field name, the value assigned to a field is usually different from what the user sees in a browser. For example, the user will see the text "Paperback", not the value "format_pb", when choosing the paperback format. Depending on the field type, some fields can be assigned more than one value at the same time. For example, as "book_format" is a check box field, we could assign both the value "format_pb" and the value "format_hc" to the "book_format" field to select both the paperback format and the hardcover format.

Most fields have a default value. The default value is the value that is initially assigned to the field in the form. For example, the "book_language" field has the default value "lang_0", because of the "selected" attribute.

A form is submitted by sending the current values of the fields to the web site. Only fields that have one or more current values are sent. For example, if none of the check boxes of the "book_format" field in the example form are checked, no value is sent for that field.

In a browser, the submission of a form usually happens when the user clicks a submit button. There are two kinds of submit buttons: normal submit buttons and image submit buttons. Normal submit buttons are defined using a <button>-tag or an <input>-tag, in both cases with the "type" attribute set to "submit". If a normal submit button has a field name and value, that field is sent with the specified value when the button is clicked.

Image submit buttons are defined using an <input>-tag with the "type" attribute set to "image". An image submit button defines two fields, named "button name.x" and "button name.y", where button name is the name contained in the "name" attribute of the <input>-tag. If the <input>-tag has no "name" attribute, the fields are named "x" and "y". When an image submit button is clicked, these two fields are assigned the x-

and y-coordinates of the position in the image where the mouse was clicked. Some web sites use this for creating image maps with different behavior depending on where the user clicks.

Some forms use JavaScript. For example, the <form>-tag may have an "onsubmit" attribute that contains JavaScript to be executed before the form is submitted. Similarly, an <input>-tag may have an "onclick" attribute that contains JavaScript to be executed when the user clicks on the field. The robot will automatically execute this JavaScript.

For performance reasons, you may decide to ignore the JavaScript execution when submitting the form. To do this, you must clear the "Execute JavaScript" option in the options in the form submitting step.

Determine the Step Action

The simplest way to submit a form is to fill in the form using the appropriate actions. For more complex submissions, you can loop through a form to get the desired result.

Consider the book search example form.

- To search for books in all available languages and for all reader ages, the site may not allow such a general search. You can loop through the languages and reader ages, making a form submission for each combination of language and age. To do this, use the Loop Form actions:
  - Loop Field Values
  - For Each Option
  - For Each Radio Button
- The Loop Form actions does not submit the form, so this must be done separately in a subsequent Click action that clicks one of the submit buttons of the form.
- To loop over a combination of field values, place several steps with Loop Form actions after each other prior to the Click action that submits the form.
- To create a URL that represents a submission of the form, use the Extract URL action on the form's submit button.

Use Loops in Forms

There are three Loop actions you can use in forms: Loop Field Values, For Each Option and For Each Radio Button. The three actions correspond to the three kinds of form controls for text input (INPUT elements with type "text" and TEXTAREA elements), options (SELECT elements) and radio buttons (INPUT elements with type "radio"). Watch the video below or read on to learn how to use these loops.

To loop over a form, you need to decide which form controls to loop over and in which order (this determines the order in which output values are generated). Next, insert a step for each with the loops with the corresponding Form action. This can be done by right-clicking the control in the Page view and choosing **Loop > <form action>** from the context menu, where `<form action>` is the appropriate action. For example, if the control is a text input control, you would choose **Loop** > **Loop Field Values**.

Every time a Loop Form action is executed, a value is changed in a form control element in the HTML page. This corresponds to what you would have done manually in a browser. If the form control has a JavaScript event attached to it, this event would be fired and some JavaScript executed. In some cases this JavaScript may change the form, such as the options of a SELECT element. In this case you must be careful and choose the right order in which to loop over the controls to ensure that the right options are available to the robot when it needs them. Normally, if you follow the order that you would use when doing this manually in a browser, it should work just fine.

Once all the Loop Form action steps have been inserted in the robot, you should add a step with a Click action that clicks one of the submit buttons of the form.

Upload Files

Some forms contain file fields that allow you to upload files. A file field is defined by an <input>-tag of type file, such as the following:

```
    <INPUT type="file" name="attachedFile">
```

In the Select File action, there are two ways to upload a file using a file field.

1. The first way is to upload a file from the file system. To do this, select **File in Local File System** from the list and enter the file name. When the form is submitted, the specified file is loaded from the file system and uploaded as part of the form submission.

   > **Note** The file name must be an absolute file name, including the drive name, if any, and the directory path to the file.

2. The second and most common way to upload a file is to specify the file contents to upload, instead of loading the file from the file system. To do this, select **File Contained in Variable** from the list. Then, you may select the variable that holds the file contents from the File Content list. Typically, you get the contents from either a binary variable in which you have downloaded the file earlier using the Extract Target action, or from a variable containing text that you have extracted earlier.

Optionally, you can specify the content type and the file name of the file. The content type should be the MIME type of the contents, optionally followed by a charset. You may use one of the predefined content types, acquire it from an attribute or specify a custom content type. For example, the content type could look like this for an image:

```
    image/gif
```

and like this for a plain text:

```
    text/plain; charset=iso-8859-1
```

Note that when downloading files using Extract Target, you can store the content type and file name of the downloaded data in other variables. You can then use this information when uploading the file with the Select File action.

Use the Context Menu on the Page View

You can use the context menu in the Page View as a shortcut for selecting and configuring the Submit Form and Loop Form actions.

> **Note** This is an obsolete functionality that is available in old robots only.

1. To select the **Submit Form** or **Loop Form** action in the current step, right-click inside a <form>-tag in the Page View.
2. In the context menu, Forms submenu, select **Use Submit Form** or **Use Loop Form**.

3. If the current step contains a Submit Form or Loop Form action, in the Page View, Forms submenu, right-click a field and select **Add Assignment to Field** in the Forms submenu.

   A dialog box appears.
4. Assign a field value.
5. If the current step contains a Loop Form action, in the Forms submenu, right-click a field and select **Add Looping over Field** to loop through the field.
6. A dialog box appears where you can configure a **One field with values to loop through** field group for the field.
7. If the current step contains a **Submit Form** or **Loop Form** action, in the Page View, right-click a submit button.
8. In the **Forms** submenu, click **Select Submit**.

Loop Through Tags on a Page

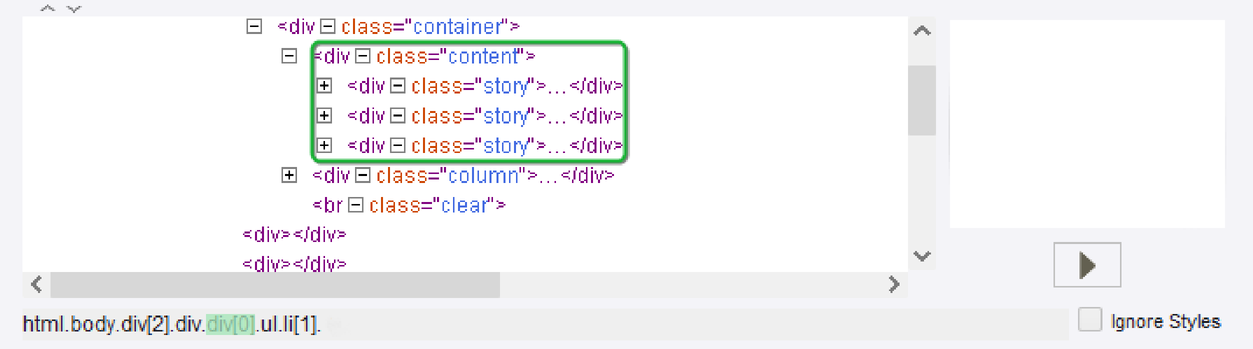
A robot often needs to loop through elements on a page to perform some action on each element. For example, you might be interested in extracting certain properties from each result in a search or from each row in a table. The following topics explain how to do this:

- Loop Through Tags with the Same Class
- Loop Through Tags with Different Classes

Keep in mind, with multiple different types of loop steps, there are many ways to handle the same situation.

Loop Through Tags with the Same Class

There are a couple of ways to set up a loop. The first way is the easiest if it is viable, and involves looping over tags that all share a class attribute.



```
⊟ <div ⊟ class="container">
   ⊟ <div ⊟ class="content">
      ⊞ <div ⊟ class="story">...</div>
      ⊞ <div ⊟ class="story">...</div>
      ⊞ <div ⊟ class="story">...</div>
   ⊞ <div ⊟ class="column">...</div>
     <br ⊟ class="clear">
<div></div>
<div></div>
```

html.body.div[2].div.div[0].ul.li[1].

Note Each div element has the attribute `class="story"`.

To determine whether looping through tags with the same class is possible, find the elements in the HTML view. In the case shown above, it is possible to loop through the three div tags with the attribute `class="story"`.

1. Right-click the first tag and select **Loop** > **For Tags With Class** > **story**.

   This creates a For Each Tag Path step in the robot, which loops through all elements on the page with the given class.
2. On the Loop step, use the arrows to ensure that the correct tags are included in the loop.

   There might be other tags on the page that you do not wish to include in the loop but that use the given class. These can be excluded from the loop with an easy fix.
3. To exclude tags with the selected class, in the **Action** tab of the editor select **Loop** > **For Each Tag Path**.

   Review the HTML view.

   The For Each Tag Path has automatically included the entire page as the found tag.
4. Change the found tag to force the robot to only loop through tags within another given tag.

   Once the loop has been successfully created, any steps added after the **For Each Tag Path** step is repeated for each iteration of the loop.

   Steps after the loop step are executed for each iteration.

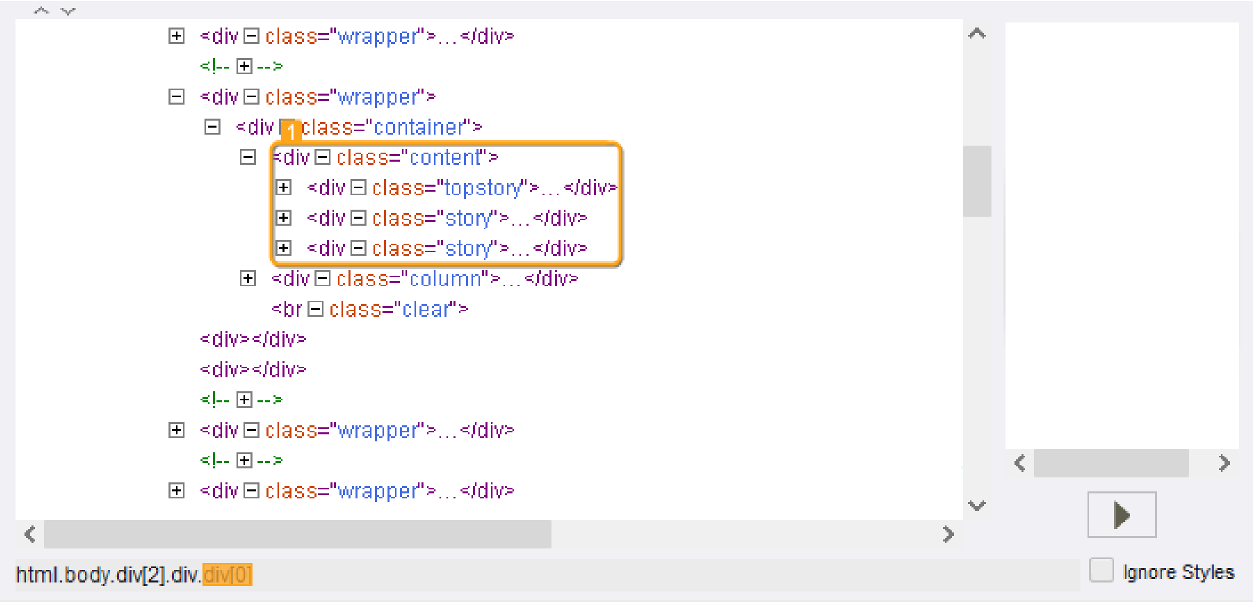
   In the example above showing the robot view, the robot extracts two pieces of text, a title and a preview, and return those values, for each iteration of the loop.

Loop Through Tags with Different Classes

Looping through tags with the same class is a common scenario, but not the most simple one you might run into. Often it is necessary to loop through tags that do not all have the same class. An alternative scenario will be explored here.

The For Each Tag step is very efficient in most cases where For Tags With Class fails. The For Each Tag step loops over all types of tags, which are directly inside the found tag. It does take a little more configuration than just right-click and insert. Here is how to use it.

Use **For Each Tag** to loop over each tag of a given type, which is directly inside the found tag.



```
⊞ <div ⊟ class="wrapper">...</div>
  <!-- ⊞ -->
⊟ <div ⊟ class="wrapper">
    ⊟ <div ⊟ class="container">
        ⊟ <div ⊟ class="content">
            ⊞ <div ⊟ class="topstory">...</div>
            ⊞ <div ⊟ class="story">...</div>
            ⊞ <div ⊟ class="story">...</div>
        ⊞ <div ⊟ class="column">...</div>
          <br ⊟ class="clear">
  <div></div>
  <div></div>
  <!-- ⊞ -->
⊞ <div ⊟ class="wrapper">...</div>
  <!-- ⊞ -->
⊞ <div ⊟ class="wrapper">...</div>
```

html.body.div[2].div.div[0]

Note that the found tag contains three div tags, but they do not all have the same class. In this scenario, use For Each Tag to handle the differences.

1. Insert an empty New Step and select the **For Each Tag** action.
2. In the Page View, select the Found tag.
3. In the **Step Action View**, Tag field, select the tag type.
4. Add steps after the For Each Tag step.

   These steps are repeated for each iteration of the loop.

Loop Through HTML Pages

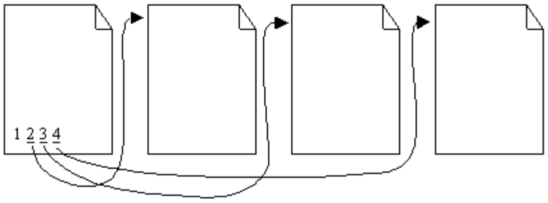
A robot often needs to loop through pages. For example, many web sites that present the results of a search request will do so over several pages, each containing e.g. 20 results from the search. To get the search results, you need to loop through the pages and process one page at a time. The following topics explain how to do this.

- First Page Links to All Other Pages
- Each Page Links to Next

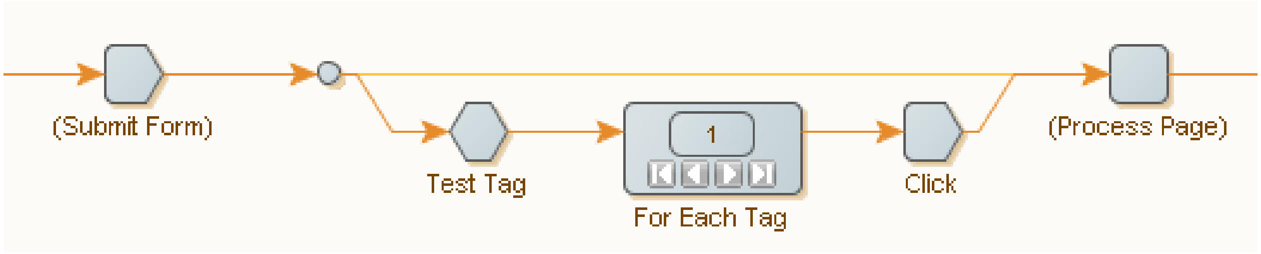
First Page Links to All Other Pages

When the first page contains direct links to all other pages, you can follow a link to access any page directly from the first page.

Note The first page can also contain a link to itself.



In this example, you can easily loop through pages using a For Each Tag step, as shown in this excerpt from a robot.



In this illustration, the robot loops through the result pages from a search request, symbolized by the Submit Form step.

The first result page is processed directly, shown by the connection from the form submission step directly to the Process Page step.

The remaining pages loop through using the For Each Tag action in the second branch.

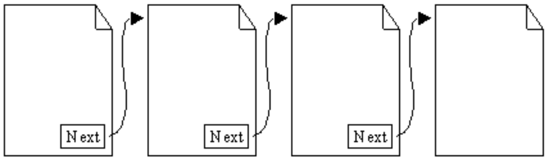
The Test Tag step checks to confirm there is more than one page.

- If the first page links to multiple pages:
  1. Loop through the tags containing the links to the pages.
  2. Load each page using a Click action.
  3. Continue to page processing.
- If the first page links to itself:
  1. Configure the For Each Tag action to skip this first link.

     The first page is not processed twice.

Each Page Links to Next

Pages are linked to a subsequent page, typically with a link at the bottom to the next page.



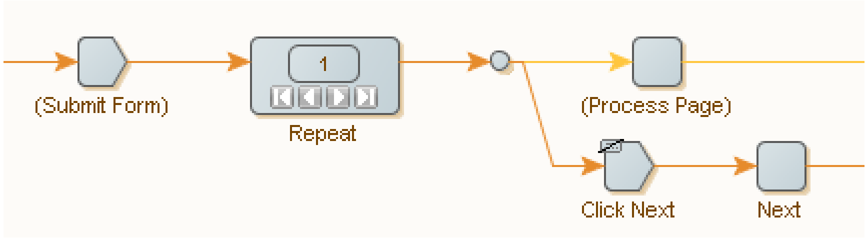
Use the Repeat action to loop through such pages. This action loops through the pages that are supplied to it by another action named Next.

Repeat and Next must be used collaboratively to have any effect.

1. On the first page add a Repeat step.
2. Insert additional actions as required.
3. Insert a Next step.

   When the robot execution reaches the next step, it reverts back to the repeat step and executes another iteration of the steps. The page is transferred to the repeat step and a new page is loaded with each iteration.

   > **Note** You can add other loops between the repeat and next steps to extract additional information from the page, as needed.

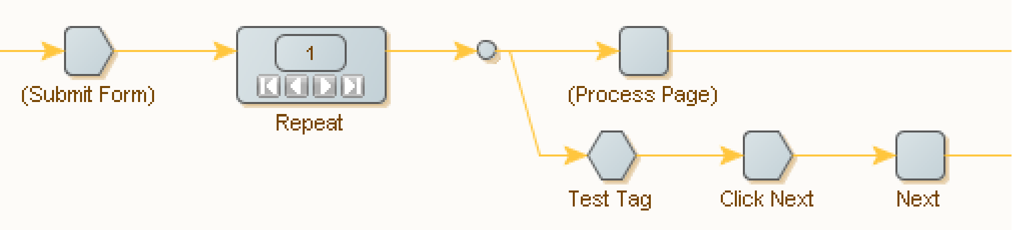
   Here, like before, we are looping through the result pages from a search request, symbolized by the Submit Form step.

   The form submission step will output the first result page, which we give to the Repeat action. In the first branch from the Repeat action, we process the current page. In the second branch, we load the next page by clicking its link. The Next action sends the page back to the Repeat action, which will output it in the next iteration. When the last page is reached, the Click action generates an error. To do this, the Click step is configured to terminate the loop. In the Click step, this is done in the Error Handling tab by setting the Then property to Break Loop.
4. To terminate the loop, In the Error Handling tab, set the **Then** property to **Break Loop**.

   If the process does not find a Next page, the process terminates.

   See Handling Errors for more information.

   An alternative way of handling the last page is shown in the following robot excerpt:

   You can use a Test Tag action in a second branch to detect when the last page has been reached. The Test Tag action checks that the page contains a next-page link, for example by looking for an <a>-tag containing the text Next. If the page contains such a link, we load this page and give it to the

Next action. When the last page is reached, the Test Tag action stops execution down the second branch, and no new page is given to the Repeat action, causing the loop to end.

Finding the link to the next page can be tricky. A common mistake is to find the previous-page link on some pages instead of the next-page link, because the layout of the pages changes slightly between the first page, the subsequent pages, and the last page. Another common mistake is to not detect the last page reliably. You may have to configure the tag finders of the steps carefully to make things work (see Using Tag Finders).

Note When you are working with a robot in Design Studio, you may not always be able to step correctly back and forth between iterations of a Repeat action. If you are not sure whether Design Studio has gotten it right, click Refresh to update.

Use Wait Criteria

Wait criteria in the *Continue when* option are powerful instruments to help you build robots in a fast and reliable manner.

Note Wait criteria are available when using the Default browser engine.

Every robot step, which requires the browser to start running, can be configured with a set of criteria to pinpoint when the processing of an action (such as a click or a page load) has completed enough for the robot to continue.

The *Continue when* feature, combined with the built-in algorithm, allows for browser steps to run only as-long-as-needed for the robot to be able to continue. Also the user can point and click an element on the page and create a new stop criterion for the browser step.

Wait criteria can be specified in the following step actions:

- Click
- Close Window
- Create Page
- Enter Password
- Enter Text
- Execute JavaScript
- For Each Option
- For Each Radio Button
- Insert Tag
- Load Page
- Loop Field Values
- Move Mouse From
- Move Mouse To
- Press Key
- Replace Tag
- Scroll

- Scroll To
- Select File
- Select Multiple Options
- Select Option
- Select Radio Button
- Set Checkbox

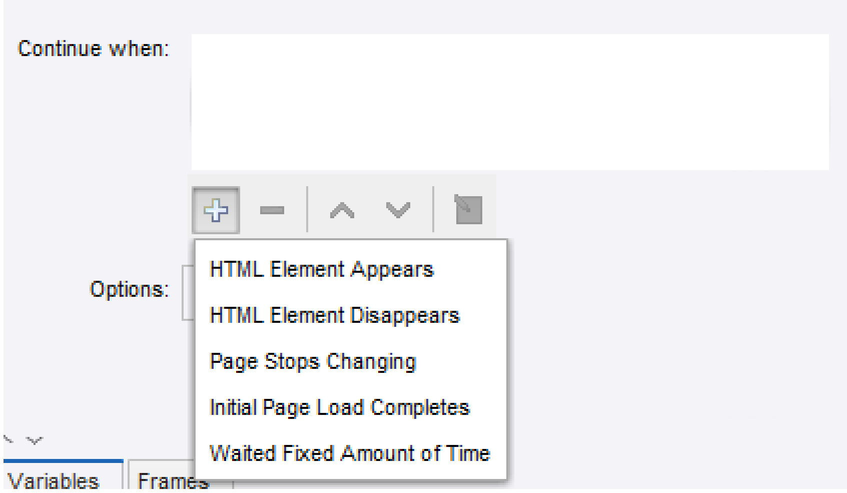
All robots created in Kofax RPA version 9.6 and later have their default waiting set to *Page Stops Changing for 500 ms*. You can see this setting on the Advanced tab of the Robot Configuration window. All wait-criteria-enabled steps have a default *No Page Changes for 500 ms* wait criterion with the enabled **Resume** button, which can be seen in the Result View. This wait criterion is always grayed out in the Wait view and always met. All other browser steps have a default *Initial Page Load Complete* wait criterion with the disabled **Resume** button. This criterion is also always grayed out in the Wait view and always met.

If you migrate the Default browser engine robot to a Classic browser engine robot, the following rules are applied:

- If a step in the Default browser robot has a specified wait criteria, this step in the Classic browser robot is set to *Wait Real-Time for Timer Events=true*
- If a step in the Default browser robot has Legacy timing, in addition to *Wait Real-Time for Timer Events=true*, this step in the Classic browser robot has *Max Wait for Timer Events* set to the time specified in Legacy timing.
- If you migrate the same robot back to the Default browser robot, wait criteria are not restored.

Adding Wait Criteria

To specify a wait criterion for a step, click "+" in the **Continue when** field and select a criterion:

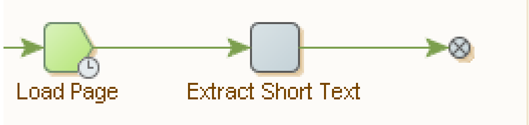


To add Wait criteria from the Browser and Source view after execution of a wait-criteria-enabled step, right-click the browser or source view, choose **Wait for** from the menu, and select a criterion. The step is re-executed after the criterion is added, which is shown by the **Re-execute step** button in the wait criterion configuration window.

Once the criteria are added, you can add, remove, move up, move down, and edit wait criteria using the panel under the list. Right-clicking a wait criterion in the **Continue when** list, opens an option menu to copy, cut, and paste a criterion.

You can add more than one wait criterion to a step. If you have several wait criteria, execution stops when any wait criterion is met. You can have several met wait criteria such as if you are waiting for two HTML elements that appear on the same load, or if you are waiting for an element on the main frame, and *Initial Page Load Completes* is set.

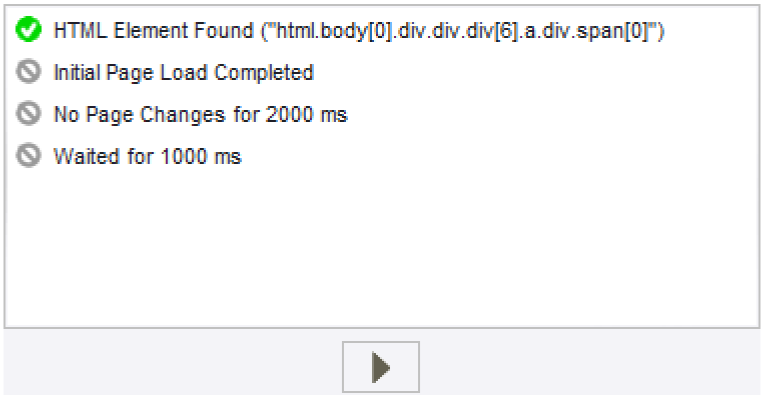
If adding a wait criteria to a wait-criteria-disabled step from a shortcut menu, the criteria is added to the previous wait-criteria-enabled step. For example, if you try to add a wait criterion after the Extract step as in the following example:



the criterion will be added to the Load Page step.

Wait View

The Wait view shows the results of wait criteria execution as well as disabled wait criteria.



✓ HTML Element Found ("html.body[0].div.div.div[6].a.div.span[0]")

⃠ Initial Page Load Completed

⃠ No Page Changes for 2000 ms

⃠ Waited for 1000 ms

Right-clicking a criterion in the list opens a shortcut menu. You can enable, disable, and delete a criterion as well as open properties for the selected criterion. For **HTML Element Appears**, you can select an element found in the DOM in the Browser view.

This view also shows whether the page was completely loaded. If the page is loaded completely, the **Resume** button is disabled. If the page is not completely loaded, because of the short timeout, the **Resume** button is disabled and you need to extend the timeout.

The **Resume** button in the Wait view is used to resume the browser operation after a wait criterion is met. If you have several wait criteria and one of them is met, clicking the **Resume** button marks the met wait criteria with a grey sign and the browser continues working until the next wait criterion is met. Once all criteria are met, clicking the **Resume** button starts loading the page during the time specified in the *Timeout for Each Attempt* option in the Options dialog box.

> **Note** The **Resume** button is enabled for wait-criteria-enabled steps.
>
> For default wait criteria you can click the **Resume** button as many times as you want. For non-default wait criteria, you can click the **Resume** button only once.
>
> If a wait criterion is displayed without an icon, the criterion is not met.

Wait Criteria Properties

Each wait criterion except *Initial Page Load Completes* has its settings. To configure a wait criterion, either double-click a criterion in the **Continue when** or **Wait** view, click ✎ in the **Continue when** view or right-click a criterion in the **Wait** view and select **Properties**.

Disabling Wait Criteria

Wait criteria can be disabled and enabled in its configuration window. By default, all criteria are enabled. To disable a wait criterion, clear the **Wait Criterion Enabled** check box. When a wait criterion is disabled, it is not taken into account during the step execution.

> **Note** You can right-click a wait criterion and use the shortcut menu to disable and enable a criterion.

After you disable or enable a wait criterion, the previous step is re-executed.

Ignore All Pending Loads When Met

Each wait criterion except *Initial Page Load Completes* has the Ignore All Pending Loads When Met option that stops loading a page when a wait criterion is met. This option can help in cases when a wait criterion is already met, but timers continue execution and loading does not stop. This option is not selected by default. If the browser was stopped by this option, a warning sign is added to the green icon in the Wait view.

HTML Element Appears

This criterion is met when a specified HTML element is present in the DOM tree. This criterion configuration is similar to the Tag Finders tab in "Step Configuration", except for additional property **Found Element must be** that contains two options:

- **Enabled**: If this option is selected, execution must stop when `result = !element.disabled;`
- **Visible**: if this option is selected, execution must stop when `result = style.display !== "none" && style.visibility !== "hidden";`

If HTML Element Appears criterion is met, it is marked in the Browser and Source views when you use the **Select in Browser View** command on the **Wait** view option menu.

HTML Element Disappears

This criterion is met when a specified HTML element disappears from the DOM tree. This criterion configuration is similar to the Tag Finders tab in "Step Configuration", except for an additional property: **Initial Element Detection** that contains two options:

- **Element is Found on Page**: the robot waits until the element appears on the page and only after that the robot waits for the element to disappear from the DOM tree.
- **Wait for Fixed Amount of Time** the robot waits the specified time and then verifies if element exists in DOM:
  - If the element is present in the DOM tree, the robot waits until the element disappears.
  - If the element is not present in the DOM tree, the wait criteria is met and the robot proceeds to the next step, even if the element has not appeared in DOM as the beginning of page load.

Page Stops Changing

This criterion is met if the DOM tree is not changing for the specified time. To set the time, open the criterion properties and specify a timeout in milliseconds in the **Timeout (ms)** text box.

Initial Page Load Completes

This wait criterion is met when the initial page load completes similar to the Javascript onload event.

> **Note** Though this criterion does not have the **Stop All Loads When Met** option, by default the loading stops when all loads are met.

Waited Fixed Amount of Time

This wait criterion is met when the execution is waited for a specified time. To set the time, open the criterion properties and specify a time in milliseconds in the **Wait (ms)** text box.

Old Robots in Kofax RPA 9.6 and Later

When you open your Default Browser robots created in Kofax RPA version prior to 9.6, you can see the *Default Waiting* settings on the *Advanced* tab of the *Robot Configuration* dialog box.

Default Waiting settings for Robots from previous releases is *Use Pre 9.6 Default Waiting*. For such robots you can change this setting to *Page Stops Changing for 500 ms*.

- If *Default Waiting* is set to *Use Pre 9.6 Default Waiting*, adding a new wait criterion to a step produces the following warning.

  Default Waiting should be set to Page Stops Changing for 500 ms in the robot settings when using wait criteria.
- If *Default Waiting* is set to *Use Pre 9.6 Default Waiting* and the step is set to use the *Legacy Timing* wait criterion, changing the Default Waiting to *Page Stops Changing for 500 ms* produces the following error.

  'Default Waiting' must be set to 'Use Pre 9.6 Default Waiting' in Robot Settings when using legacy wait criteria.

Upgrading Existing Robots to 9.6

If you open your robots created in Kofax RPA version prior to 9.6 (9.3, 9.4, 9.5), depending on the **Max. Wait for Timer Events** and **Wait Real-Time for Timer Events** settings, you must perform different steps to use new wait criteria in your robots:

Max. Wait for Timer Events and Wait Real-Time for Timer Events settings are not default

If *Max. Wait for Timer Events* and *Wait Real-Time for Timer Events* settings are not set as default in your robots, Design Studio adds a non-editable *Legacy Timing* criteria to your robot.

This wait criterion is always green after step execution. If you add any new wait criterion, *Legacy Timing* is automatically removed and you can take advantage of the new Wait criteria.

Max. Wait for Timer Events and Wait Real-Time for Timer Events settings are default

If *Max. Wait for Timer Events* and *Wait Real-Time for Timer Events* settings are set as default in your robots, you need to perform the following to switch to the new default settings when updating an existing robot to 9.6: go to the **File** > **Configure Robot** dialog box, click the **Advanced** tab and clear the **Use Pre 9.6 Default Waiting** check box. Now you can take advantage of the new Wait criteria in your robots.

Fixing Page Load

In some cases when a robot uses the *Use Pre 9.6 Default Waiting* or *Legacy Timing* options, after loading the page you get the *Page loading completed* message, but the page is not completely loaded. This might happen when previous page loads were aborted. To fix page load, remove the *Use Pre 9.6 Default Waiting* option by changing it to the *Page Stops Changing* option and use a new wait criterion on the step.

Extract Content from HTML

Design Studio has six step actions for extracting content from a tag in an HTML page:

- The **Extract** action is used to extract text content from the tag, optionally including the HTML tags.
- The **Extract URL** action is used to extract a URL from a tag attribute containing a URL, and make that URL absolute.
- The **Extract Tag Attribute** action is used to extract the value of a tag attribute.

- The **Extract Target** action is used to extract binary data such as images and PDF files, but it handles any kind of binary data.
- The **Extract Form Parameter** action is used to extract a form parameter from a form URL in the found tag and then store its value in a variable.
- The **Extract Selected Option** action is used to extract the selected option from a <select>-tag and then store it in a variable.

To reformat (or normalize) the extracted content, use the Extract and Extract Tag Attribute actions and configure data converters in the list.

There are two actions to extract data from various binary data formats, for example, PDF or Flash. These are different from the preceding actions in that they extract the data and produce an HTML page that contains the data in a structured form that lets your robot access the data. These actions are used in an initial step before the actual data extraction, in which you may loop over the produced HTML and extract text.

- The **Extract from PDF** action is used to extract text from a PDF document contained as binary data in a selected attribute.
- The **Extract Flash Content** action is used to extract data from a Flash object in a found tag.

Extract Text

1. On the Action tab, select **Extract**.
2. To extract short text, such as a product name or a price, extract as **Only Text**.

   This extracts the text between the tags.
3. To extract longer text with sections, headings, and so on, you can select as plain text. If you want the text to appear close to how it appears in a browser, extract the text as **Structured Text**.
4. To extract with special markup, such as brackets surrounding the headings, select **Structured Text**.

   Structured Text has rudimentary support for special markup.
5. If the markup requirements cannot be fulfilled using the Structured Text option, select **Advanced Structured Text**.

   This option allows you to set mappings from the HTML tags into your proprietary markup.

Extract Binary Data

Extract binary data using the Extract Target action.

On the Action tab, select **Extract Target**.

The URL data is loaded and stored in a variable, or directly into a file.

Typically, binary variables are used to store the loaded data. The available types of binary variables include Binary, Image, PDF, and Session. They are all equivalent except that the Image, PDF, and Session types allow you to preview the data.

Use the Context Menu in the Page View

1. Right-click the text or tag to extract from, or right-click the link to load from.
2. On the Extraction context menu, select the appropriate option.

Perform Common Tasks

Extracting Only Part of a Text

To extract only a part of the text in a tag, you can use patterns on the text in the tag. For example, you might want to extract the name "Bob Smith" from the following text: "The article is written by Bob Smith." To do this, use the Extract data converter (do not confuse this with the Extract step action) and configure it as described in this topic.

In this example, the pattern used is ".*by\s(.*)\.", which means that the text between "by" and the period will be matched by the subpattern. For more information, see Patterns.

1. Open Extract Configuration, and select the **Basic** tab.
2. In the Pattern field, enter the text pattern to extract.

   Configure the Pattern property to match the entire text, with the text to extract matched by a subpattern, enclosed by parentheses.

Converting Content

To normalize content, use Conversion, such as replacing text with another text. For example, to normalize country codes to their natural language description, such as normalizing "US" to "United States."

- For plain text conversions, use the Convert Using List data converter.
- For conversions based on patterns or expressions, use the If Then data converter.

Extracting and Formatting Numbers

1. To extract a number from content, add an Extract Number data converter.
2. To perform additional number formatting, use the Format Number data converter.

Extracting the Date from Text

Extracting dates should be done in the same fashion as extracting numbers.

1. To extract a date from text, add an Extract Date data converter to your robot.

   Extract Date uses patterns to extract the date. The pattern does not have to match the entire text, only the date. The extracted date is converted to standard date format.
2. To perform additional date formatting, use the Format Date data converter.

Extracting Only a Subset of the Tags in a Found Tag

Sometimes, you want to extract from a range of tags rather than a single tag.

For example, consider the case of extracting the body text of an article, where the body text is made up of individual sections, each in their own tag, and where information about the article title and author is contained in some other tags. To extract only the body text without the article title and author, use the Extract action to extract the text, and configure the action so that only the range of tags spanning the body is extracted.

1. On the Action tab, select **Extract**.

2. Specify the first tag in the range.
3. Specify the last tag in the range.

Local Files Usage in Robots

You can use Robots to load many types of files including HTML, Excel, CSV, and regular text files. This enables robots to extract data from a variety of sources.

- The following file types can be loaded natively by robots: HTML, XML, Excel, and JSON.
- Other file types can be loaded but are converted to HTML before being handled by the robot: plain text, CSV and PDF.

There are two procedures to load file types. If the file is located on the Internet, it is loaded using the Load Page action, specifying the URL of the file or using the Click action, clicking a link to the file. This automatically loads the file up in the page view. If the file is located on your system, load the file in the following way to ensure that the file is also available upon uploading the robot to the Management Console to be scheduled or added to a Kapplet.

All file types, except PDF, are loaded by the same procedure. To add a file to the robot, perform the following:

1. In the Add Variable form, add a binary type variable to the robot.

   Note Other variable types such as PDF and HTML can also be used, but are not as flexible as the binary type and may not permit user input.

2. Enter a name.
3. In the **Type and Initial/Test Values**, select an option from the list.
4. Select the **Global** and **Use as Input** options as required.

   Note The difference between checking and not checking Use as Input only matters if the robot is to be scheduled or used in a Kapplet in the Management Console. An input variable is definable by the user, and so the file will be interchangeable each time the robot is run. On the other hand, if the file should be the same each time the robot is run, there is no need to use an input variable.

5. Click **Load** to load a test file, and then click **OK**.

   If you have not selected Use as Input, this test file is the final one.

   A variable with an attribute of the type binary is added to a robot. It is defined as an input variable to allow users to input other files in Kapplets and Schedules.

   A test file is loaded into the attribute.

Load an Excel Page from a Variable

Even though the most common way to load a page in Design Studio is using a Load Page step, a robot may also receive Excel documents as input in a Binary attribute. To load an Excel document into the robot to loop and extract data from Excel documents, perform the following steps:

Important Design Studio may become unresponsive when working with large Excel files. To avoid that, try working with smaller Excel files. Alternatively, increase the physical memory (RAM) of the machine or allocate more memory in the `memory.conf` (this file is used to configure memory usage).

1. In the robot workflow, insert a **Create Page** action step.
2. In the **Contents** list, select the binary variable.

   This is used to load the file into the Page View.

   To select a variable, set the **Contents** list value selector to Variable, then select the binary variable from the list.
3. Click **More** to open **Options** dialog.
4. On the Page Loading tab, Page Content Type, select **Same for All Pages**.
5. In the Content Type field, select **Excel** and click **OK**.

   The robot can now recognize the Binary data from Excel pages.

Extract Content from Excel

Design Studio has three steps for extracting content from a spreadsheet:

- The Extract Cell step is used to extract text content from the found range.
- The Extract Sheet name step is used to extract the sheet name of the sheet of the found range.
- The Extract As HTML step is used to extract the found range of a spreadsheet as an HTML page containing a table with the cells of the range into a variable.

For the Extract Cell and Extract As HTML steps you can specify what to extract from the cells. This is controlled by the value of the Extract This option. The choice here is the same as the View Modes for the Spreadsheet View. The possible options are described in this topic.

Formatted Values

The extracted values are what you see in Excel and the values of dates and numbers are extracted formatted, which means that numbers may have fewer decimals than the actual values of the cells.

Plain Value

The extracted values are the actual values that Excel would show if the values of the cells were not formatted. For example, numbers would not have rounding of decimals.

Formulas

If a cell contains a formula, it is extracted or otherwise, it is the same value as for the Plain Values option is extracted.

If you create the steps by right-clicking the Spreadsheet View, the value of **Extract This** is set to the value of the selected **View Mode**. If you set the View Mode to Formulas and then right-click in the page view and select **Extract** > **Extract Text** from the context menu (into a text variable), the Extract This option of the Extract Cell action step is set to Formulas.

You may need to reformat (or normalize) the extracted content, and the Extract Cell action allows you to do this by configuring a list of data converters.

To do so, in the Spreadsheet view, right-click to create a step. Select the desired extract step and specify necessary parameters.

Shared formulas in Excel files

The built-in Excel driver does not support documents with shared formulas. A shared formula is a cell with a formula that is automatically copied to other cells. Any operation that changes the structure of an Excel document containing shared formulas, such as adding or removing rows, may lead to errors in this document.

This limitation is only observed in Excel documents created outside Design Studio. An Excel file created with a robot cannot contain shared formulas.

Workaround: Ensure that your Excel document does not contain any shared formulas. When copying a formula cell to many cells, do not copy it to multiple cells at once; instead, copy the formula to one cell at a time.

Alternatively, you can use the convertSharedFormulas.snippet file included in the Snippets folder of your Kofax RPA installation to convert any shared formulas in your Excel document. The snippet performs these steps:

1. Takes an Excel document containing shared cells.
2. Loops over all formula cells.
3. For each of these cells, extracts the formula and set it on the cell again.

Extract Values from Cells

Use the Extract Cell step to extract the content of a cell or a range of cells into a variable.

1. On the Action tab, select **Extract Cell**.
2. Select an option in the Extract This field.

   If the found range is a single cell, the value of this cell is extracted. If the found range contains more than one cell, the values of the cells are extracted as text in which the cells are tab separated and rows are separated by new lines. In both cases, the extracted value stored in the variable is created by applying the converters to the extracted value.

   The value extracted from a cell is essentially the content of the cell in Excel taking the value of the Extract This option into account. For a blank cell, the value is the empty string, and if a cell is part of a merged cell such as C4:D6 (created in Excel by merging cells), the extracted value is blank unless the cell is the top left cell C4 of the merged cells.

Extract a Sheet Name

The Extract Sheet Name step is used to extract the name of a sheet. This step is useful when combined with a Test Value step to skip a sheet with a given name while looping over all sheets. This step is also useful to extract a sheet name to an attribute of a variable of complex type so that it becomes part of a returned value.

1. On the Extract Sheet Name step, Action tab, select **Extract Sheet Name**.
2. In the Variable field, select **text**.

   This action extracts the name of an Excel page into a variable.

Extract as HTML

The Extract As HTML step is used to extract part of a spreadsheet document as HTML source code stored in a structured text variable, such as HTML type. The extracted code contains the extracted range (in a header tag), such as Sheet1:A1:H17, which means that the name of the sheet is contained in the code. The cells of the found range are placed in a table in the generated code. This step is mainly for obtaining an HTML version of part of a spreadsheet so that it may be returned for the robot and presented in a browser. It is also possible to use the step in a robot to create an HTML page with the extracted code using a Create Page step. We do not recommend using the Extract As HTML step to convert a spreadsheet into an HTML page to access its content in that way, because it might result in poor performance of the robot.

Test Cell Types in Excel

To test the content of a cell in an Excel page, first extract the cell content, and then use a Test Values step to perform the actual test. This is essentially the same as what you would do in other page types, such as HTML. To determine the cell type of a cell would not be straightforward or even possible by just extracting the content of a cell and subsequently performing a test on it; for example, there is no way to determine whether a cell is blank or contains an empty text. Fortunately, Design Studio contains a step to perform such a test: the Test Cell Type step.

You can test six different cell types. They correspond directly to what you can test for in Excel using functions such as ISTEXT or ISNUMBER.

Blank

Corresponds to the Excel function ISBLANK.

Text

Corresponds to the Excel function ISTEXT.

Number

Corresponds to the Excel function ISNUMBER. This type also includes dates since they are represented as numbers in Excel.

Logical

Corresponds to the Excel function ISLOGICAL, which correlates to the type Boolean in Design Studio.

Error

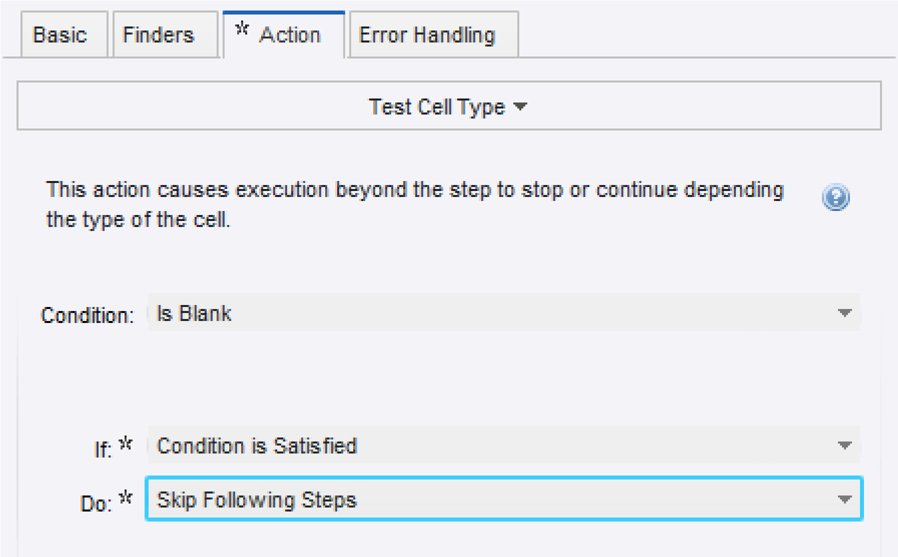
Corresponds to the Excel function ISERROR.

Formula

Corresponds to the Excel function ISFORMULA.

The Test Cell Type works like any other test step. It tests that the cell type in the found range matches a specified type, and based on the result, determines whether to continue along the branch or skip the following steps. The step is described in further detail in Test Cell Type.

An important property of the Test Cell Type step is that it can test the type of many steps simultaneously. For example, consider how you would test an entire empty row. This test could be useful when looping over a document containing several identically structured tables separated by blank lines. The following figure shows how to configure the Test Cell Type step. In this example, the branch following the step is skipped, if the cells in the found range are all blank.



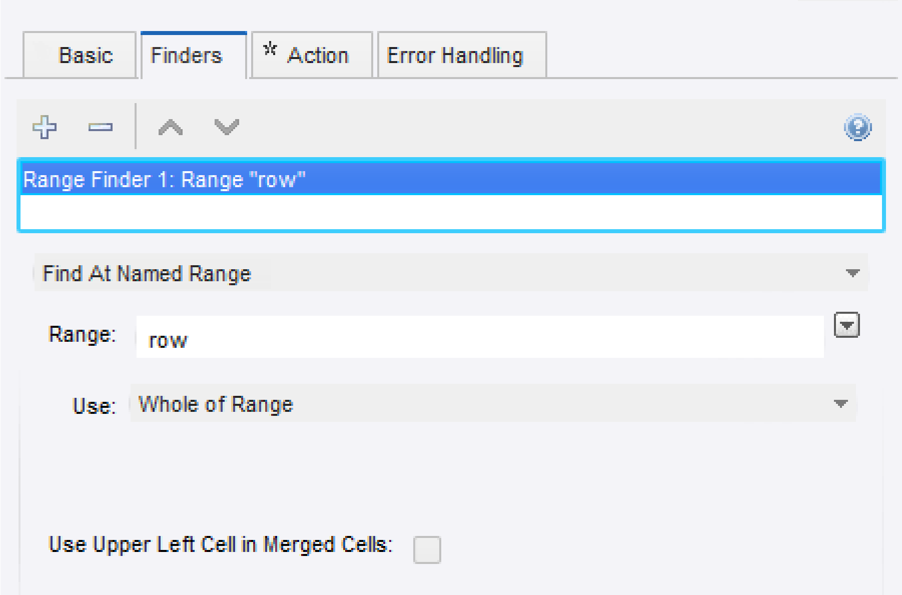
This action causes execution beyond the step to stop or continue depending the type of the cell.

Condition: Is Blank

If: * Condition is Satisfied

Do: * Skip Following Steps

The following figure shows how to configure the Range Finder such that it finds an entire row. In this case we have a named range called "row" that is set by a Loop in Excel step looping over rows and occurring before the Test Cell Type step. We have specified that the result should be the entire row by selecting Whole of Range for the Use property.



Range Finder 1: Range "row"

Find At Named Range

Range: row

Use: Whole of Range

Use Upper Left Cell in Merged Cells: □

Loop in Excel

Looping in Excel is in many ways similar to looping in HTML, but much simpler due to the simpler structure of Excel. Essentially you may loop over all the sheets in a document or you may loop over the cells of a sheet either by looping over the rows, columns or cells of a found range. To loop in Excel you use the Loop in Excel step. This step has many options in common with steps that loop in HTML, such as "First index" and "increment," which are described in detail in the reference documentation.

You can insert a loop step that loops through all the rows in a table.

1. In the "Using a robot which loads from an Excel document," click the upper left corner of the Excel view to select the entire spreadsheet.
2. Right-click inside the selected area.

   A list of options appears.
3. Select **Loop** > **Loop Rows in Selection** > **Exclude First Row**.

   This excludes the header row of the spreadsheet from the search. The Loop in Excel step now sets the first cell in the loop as the named range.

   It is now possible to extract from the named range, and because of the loop, corresponding values are extracted from the other rows.
4. Right-click the top cell in a column just below the header and select the information to extract. For example, to extract a series of identifies, right-click the first cell in the ID column and select Extract, Extract Number, ID.

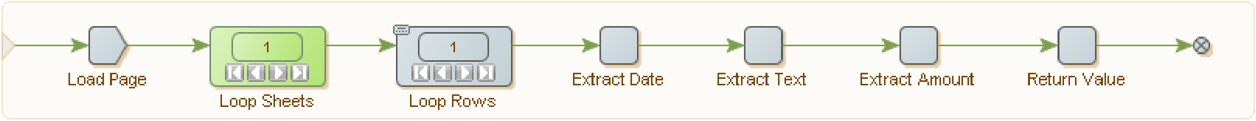
   A wizard appears with the Format Pattern correctly configured.
5. Click **OK**.

   The wizard closes.
6. Repeat steps 4 and 5 for each Named Value to extract.
7. Click **Debug** to switch to debug mode.
8. On the toolbar, click **Run**.

   The values appear in the results.

Loop Over Sheets and Rows

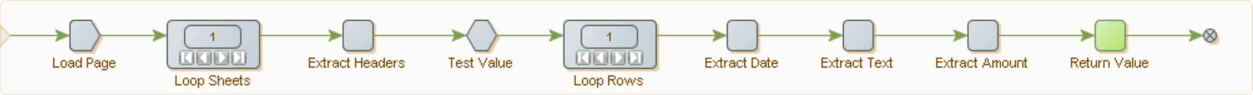
You can create a robot to loop in an Excel document with multiple sheets containing tables and with the same type of data. For example, each sheet in the Excel spreadsheet to display account information for a separate month of the year. In this case, you would have your robot first loop over the sheets and then over the rows of each sheet. You may also like to handle situations where the document contains a sheet that does not contain data of the same type as the other sheets, such as a blank sheet. The following image shows the structure of such a robot.



The first step in this robot is a Load Page step that loads the Excel document from a URL. The robot then contains a Loop in Excel step that loops over all the sheets of the document. For each iteration of this first loop step, the robot executes another Loop in Excel step that loops over each row of the sheet. The Error

Handling property `Then` of the step that loops over rows is set to **Next Iteration**, which means that if the range finder of the step fails to match a range with the size of the table, it goes to the next iteration.

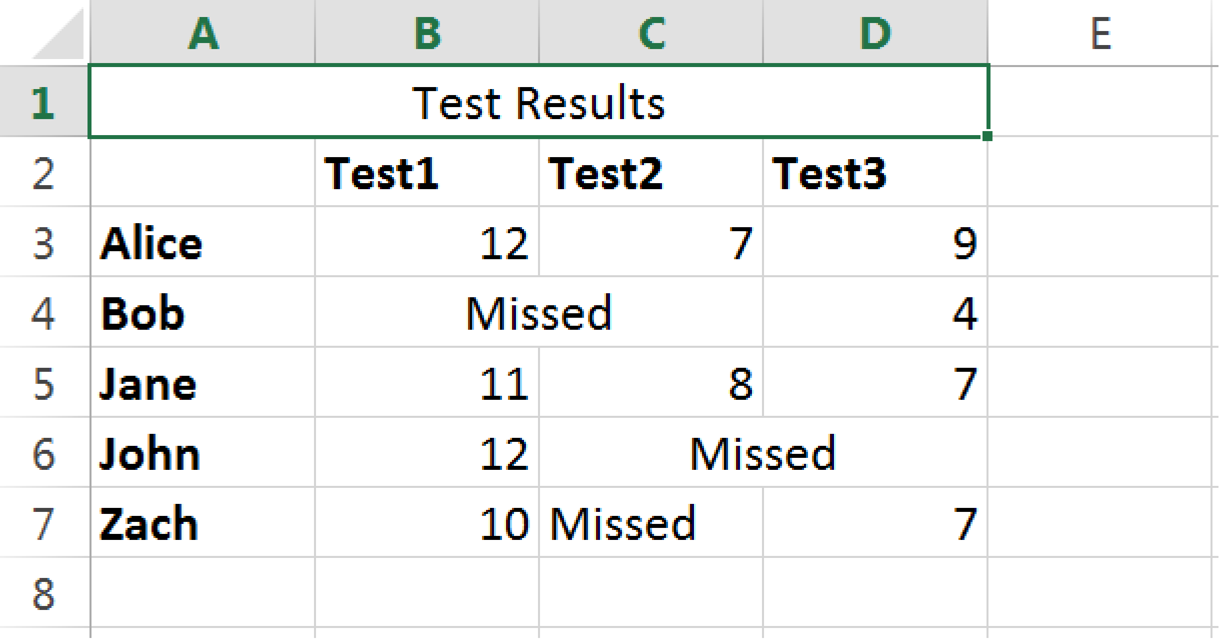
This simplified error handling will handle the simple situation where a sheet is blank, but not situations where a sheet contains a table with entirely different types of data. In general, you would have to insert a step to extract part of the sheet followed by a step to test the structure. One example could be extracting the column headers and testing that they have some given structure. The following image shows the error handling added to a robot.



In this example, the Extract Cell step named Extract Headers, extracts the first row of the sheet into a variable and the Test Value step has a condition that tests the value. If the value matches, the robot executes the next step (the Loop Rows step). If not, the robot skips the following steps; the Do property of the Test Value step will **Skip Following Steps**.

Loop Over Merged Cells

A merged cell in Excel is two or more adjacent cells merged into one cell and shown as one. You can configure your robot to loop over merged cells. The content of a merged cell is stored in the upper left cell of the cells and all other cells are blank. Looping over a table that contains merged cells can cause extraction problems. For example, if you look at the following sheet that shows test results for students, notice that some student have missed their test and in some cases, two tests are shown using a merged cell.



	A	B	C	D	E
1	Test Results				
2		Test1	Test2	Test3	
3	Alice	12	7	9	
4	Bob	Missed		4	
5	Jane	11	8	7	
6	John	12	Missed		
7	Zach	10	Missed	7	
8					

Looping over the rows to extract the student test results may fail to extract the results correctly when a student has missed a test because the text "Missed" is not a number. To correct this, you can insert a test to search for the term "Missed" and then store the value 0 for a failed result. This test does not work for situations where the cell has merged. In the preceding example, this would work fine for the cell B4 because it contains the value "Missed," but it would fail to work for C4 because the content would be a blank value.

Instead of having yet another test for blank cells, you can use an **If Then** data converter on all Range Finders to identify a single cell inside a merged cell, and return the upper left cell of the merged cell.

1. On the Finders tab, description field, enter `Range Finder 1: Column at +2(in range named "row")`.
2. In the Range field, select **row**.
3. In the Use field, select **Column At Position**.
4. In the Column field, select **By Index**.
5. In the Offset field, enter the integer 2.
6. In the Height field, select **Same As Range** and **Height is to the bottom of the named range**.
7. Select **Use Upper Left Cell in Merged Cells**.
8. In the Action tab, Extract This field, select **Formatted Values**.
9. In the Converters field, enter an If Then statement. For example, `if contains "Missed" then "0" Else INPUT`.

   The Extract cell tests for the text "Missed" and uses 0 for the result. If Missed is not found it uses the extracted value.

Create and Reuse Snippets

A snippet can be created in three ways.

1. From a selection of steps:

   (must be steps that can be grouped and not a single Group step)

   a. Select one or more steps and click **Create Snippet from Selection** on the Edit menu.

   b. Enter a name for the new snippet.

   c. Create a snippet of that name containing the select steps.
2. Turn a group step into a snippet step:

   a. Select a Group step and click **Convert Group to Snippet** on the Edit menu.

   b. Enter a name for the new snippet.

   c. Create a snippet of that name containing the group steps.
3. Create a snippet from a new snippet:

   a. On the File menu, select **New Snippet**.

   b. Enter a name for the new snippet.

   An empty snippet appears in your project and the Snippet editor opens.

   > **Note** You cannot edit the snippet contents (the steps inside the snippet) inside this editor.

   c. Edit the description and referenced variables list as needed.

Variables and Snippets

Just like steps anywhere in a robot, the steps in a snippet can use variables. The steps of snippets are always edited inside a robot. In that context the variables defined on the robot can be used in the snippet. Reusing the snippet in another robot requires you to define the variables used by the steps in the snippet on each robot that uses the snippet.

A snippet can define its own variables. Open the snippet in its own editor to define variables on the snippet. If the snippet already contains steps using variables that existed in the robot where the snippet was edited, the steps are marked with a red flag.

If a snippet defines variables, using the snippet in a robot automatically adds the snippet variables to the set of variables for the robot.

A robot should not contain variable definitions by the same name as variables defined in the snippets it uses. If it does, the variable types must match.

Removing a snippet from a robot also removes the variables imported by that snippet.

Snippet Best Practices

Consider the following snippet best practices.

- Put the non-default robot configuration used to execute the steps inside the snippet, on the steps of the snippet. This way you do not need to remember to set them on each robot using the snippet.
- When inserting a snippet into a robot, take care that names of variables defined on the snippet do not conflict with variables defined on the robot. Design Studio cannot handle a situation where a variable defined on the snippet has the same name as a variable defined on the robot. It is a good practice to define variables on the snippet when they need to work in another context. This makes reuse of the snippet easier.
- In the snippet description, document the context in terms of named tags and/or windows required by the snippet.
- Use caution when including snippets inside snippets. A snippet may contain snippet steps referencing other snippets. However, a snippet cannot contain a circular reference (such as a cyclic reference where the snippet contains itself). If a snippet contains a circular reference, Design Studio reports an error.

Reuse Sessions

A session is the result of browsing on a website, and consists of the page, the page URL and the cookies and authentications obtained in the course. However, obtaining a session where the desired information is easily reached can require a number of navigation steps such as logging in.

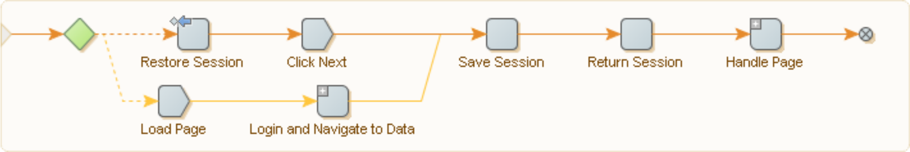
If a robot is run frequently enough, and the response time needs to be very small, getting to a suitable session in the robot can require more time than is available. However, if the session is obtained once, and then shared between robots and robot runs, then great time savings is achieved.

Two step actions are used for session reuse:

1. The Save Session action: Saves a session in a variable.
2. The Restore Session action: Restores a session from a variable.

Example

Assume that we have a robot that logs in to a web site to collect and return data. However, the data that we seek to collect is distributed over many linked pages, such as with a next page link. We want the first invocation of the robot to log in to the site and return the data of the first page, and each subsequent invocation should then return a new chunk of data (the next page). We want to share the session of a logged-in user between robot invocations, but we also want to remember how much data we have returned. The robot could look something like the following example.



When the robot is called, it will first try to restore a session from an input variable. If one exists, that session is used and the next step clicks a next page link to get a fresh page of data. If no session is passed to the robot, the step fails, and the second alternative is executed, which does the logging in and also navigates to the relevant page on the site where the data may be found.

If the robot execution gets through one of the two alternative branches, it reaches the Save Session step. This saves the session to use the next time the robot is called. But for this to be possible, we need to return the session to the caller of the robot. This is handled by the Return Session step, which is a normal Return Value step that returns the value of a variable containing the session (the variable is of a type that has an attribute of attribute type Session in which our Save Session step stored the session). Finally, if the robot reaches the end of the data (no next page link exists on the page), then the Click Next set produces an error. This is ignored by the robot, because we set Error Handling to Skip Following Steps, but if we have a check mark in API exception, the caller would get an exception. For example, if the robot is called from Java, uses the check mark to know that the end of data has been reached.

After the session is saved, the remaining steps of the robot extract the data from the page, such as by looping over a table and returning a value for each row.

Note that in Design Studio, robot execution is not controlled by the natural flow of a robot run. It is controlled by the user interaction.

1. To store the session, select the step following the Save Session step.
2. Select the Restore Session action.

Modify an Existing Type

If you need to change a type after writing robots using variables of that type, be cautious. Your robots might stop working if you do something wrong.

You should take care when performing any of the following changes to a type that is already used by variables in existing robots; otherwise, you cannot use those variables (however, the robots may still be loaded without the variables):

- Change the name of a type.
- Delete a type.
- Remove or rename an attribute in a type if, in a robot, one or more variables of that type have assigned values different from the default for that attribute.
- Change the attribute type of an attribute if, in a robot, one or more variables of that type have assigned values that are not compatible with the new attribute type.

If your robot is open while you make any of the preceding changes, you see a red status bar at the top of the Robot Editor with a text explaining the problem. The status bar also contains a button that you can click to reload the robot with the compromising variables removed. You can also solve the problems by making the appropriate changes to your types. If you do this, you can return to your robot and continue working.

The following changes to a type can be automatically carried over to robots without having to remove any variables of that type (you may have to reload), but some errors might be generated when the robots are executed (you can subsequently fix the errors):

- Change the name of an attribute.
- Change the Required property of an attribute from false to true.
- Add a new attribute that has the Required property set to true.
- Delete or rename an attribute that is assigned a value in a variable.
- Change the attribute type of an attribute that is assigned a value in a variable.

You can always make the following changes, without affecting existing robots:

- Change the Required property of an attribute from true to false.
- Change a comment (no matter where).
- Add a new attribute that has the Required property set to false.

Work with Variables in the Applications View

The Applications View shows part of the current robot state, such as a loaded HTML page or a JSON document. Variables or attributes of certain simple types (XML, JSON and Excel) can also be shown in a tab in the Applications View. When a variable is shown in the Applications View, you may operate on it in the same way as other documents loaded in the Applications View. For example, you can extract, test, and loop over, and in most cases you can also modify the variable.

As an example you may want to call a web service that takes some XML as input, and as output also returns some XML. You may then create the input XML using an XML variable that you modify using a step action that works on the content of the window showing the variable. When it has the desired form, you may feed it as input to a web service step action. You can have this web service step action store the response in another XML variable, which you may then loop over and extract data from.

Open a Variable

To work with a variable (or an attribute) in a window, you must first open the variable in a new window. You do this with an Open Variable step action.

The easiest way to do this is to right-click the variable in the Variables View and select the menu option **Insert Step** > **Open Variable**.

1. In the Variables View, right-click the variable and select **Insert Step** > **Open Variable**.

   When this step is executed, a new window shows the content of the variable. In this way the Open Variable step action behaves much like the Load Page step action.

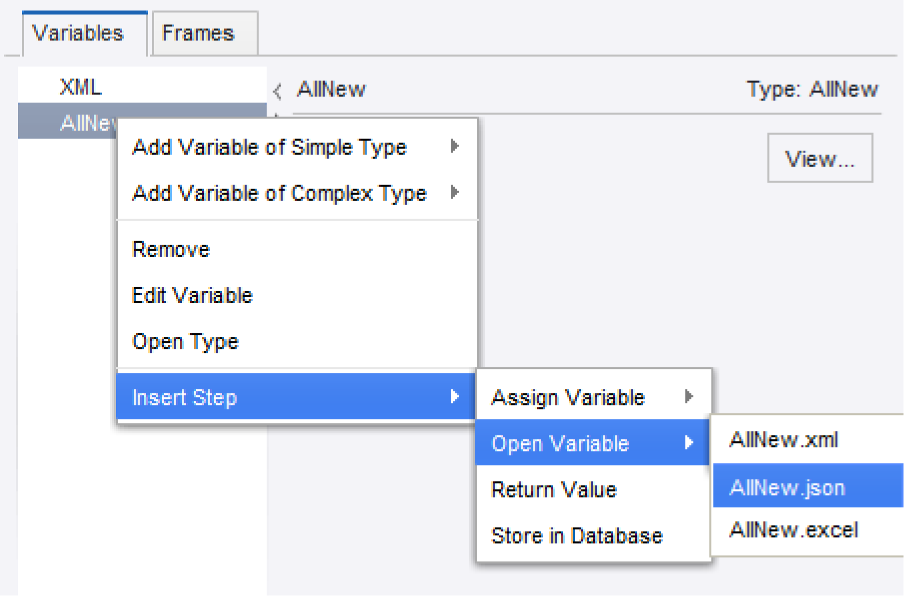
   If the variable is already open, no new window is opened; but the window containing the variable becomes the new current window. In this way the Open Variable step action behaves differently from the Load Page step action and more like the Set Current Window step action.

   Even though the step action is call Open Variable, it also works on attributes of variables if they are also of a type that may be opened in a window.

   Once a variable (or attribute) is open, you work on it just as you would on a document (such as an XML document) loaded from a URL.

   You can insert a step action to operate on the variable by right-clicking in the view. The insert steps will work on the current window whether it is loaded (opened) from a variable or a URL. The only real difference is that document loaded from a URL may not be modified, and it is considered immutable. To modify a document you must first extract it into a variable and then modify it.
2. On the Variables tab, right-click XML or All New and select configuration options.

   The following figure shows how to open an attribute of a JSON variable of complex type.



   An Open Variable step is inserted in your robot before the current step.
3. Right-click the variable and insert a step action to operate on the variable.

Modify a Variable

You can modify XML and JSON variables. Both variable types have a range of dedicated step actions that may be used to modify them. For example, the Set Attribute sets the value of an existing attribute or adds a new attribute to an XML tag, the Set Property Name step action changes the name of a property on a JSON object, etc. You can also modify a variable by using a step action that operates directly on the variable, such as Assign Variable. In that case, the view will reflect the changes.

Step actions that modify XML variables through the current window:

- Set Tag
- Set Content
- Set Text
- Set Tag Name
- Set Attribute
- Insert Content
- Remove Tag
- Remove Content
- Remove Attribute

Step actions that modify JSON variables through the current window:

- Set JSON
- Set Property Name
- Insert JSON
- Remove JSON

When a variable of type XML or JSON is shown in the current window, the menu option for inserting the step actions are available in the context menu (right-click menu) in the window. Only those relevant for the given type are shown. Some may be disabled, if the current choice in the view is not relevant. For example, Remove Attribute will not be enabled if the selected tag does not have any attributes.

Work with JSON

JSON (JavaScript Object Notation) is a lightweight data-interchange format that resembles JavaScripts literal notation such as { "x" : 5 , "y" : 7 }.

JSON is a text format, but in robots the JSON structure is represented and viewed similar to the way XML is represented. JSON is treated as its own data format (exactly as HTML, XML and Excel) with its own Page Type. It is not transformed into XML as it was in previous versions of Design Studio. The Test Page Type step action can verify that the content of the current windows is JSON.

In Applications View, JSON is loaded from a URL or from variables/attributes of simple type JSON. A dedicated view is available to view JSON variables opened in both the Applications View and the Variables View, along with dedicated step actions that work only on JSON.

The following is an example of a JSON text:

```
{ "answer" : 42,
   "people" : [ { "firstName" : "Arthur",
                  "lastName" : "Dent" },
                { "firstName" : "Ford",
                  "lastName" : "Prefect" } ] }
```

JSON Terminology

A JSON text is either an object, `{ "a" : 5 }` or an array e.g. `[1, 2, 3]`. A JSON value is either a JSON text or JSON Simple type where a JSON Simple type is either a JSON literal, a number, a string. A JSON literal is false, null or true. The literals false and true are called Booleans. A number may be either an integer or a floating point number. There is no limit on the precision or size of numbers, but as soon as they are converted to another representation, the limitation of that representation must of course be respected. For example, if an integer is extracted to an integer variable then the value must be between -2^{63} and $2^{63}-1$; otherwise the extraction step will produce an error. JSON strings must start and end with a double quotation mark (") and may contain any Unicode character except `",  \` or control character (these characters may be escaped using \, such as \", \\ and \r. The JSON format is described in RFC 4627 on the https://www.ietf.org website.

JSON Syntax

JSON Text = JSON Object | JSON Array

JSON Object = {} | { Properties }

JSON Array = [] | [items]

Properties = Property,Properties

Property = String :JSON Value

Items = JSON Value,Items JSON Value = JSON Text | String | Number | false | null | true

String = "" | "Characters "

Characters = Character Characters

Character = any Unicode character except ", \ or control character | \" | \\ | \/ | \b | \f | \n | \r | \t | \u 4 hex digitsNumber = a number very much like a C or Java number

JSON MIME Type

The MIME media type for JSON text is as follows:

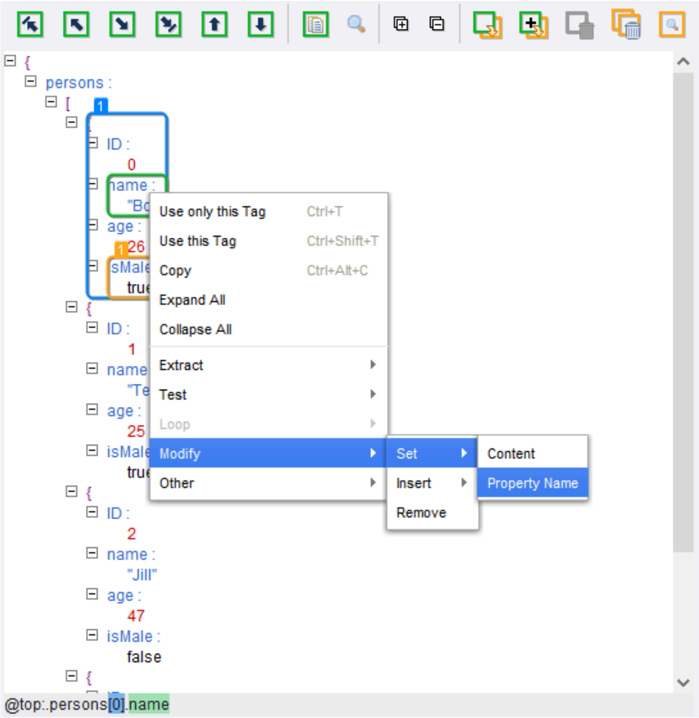
```
application/json
```

Strictly speaking, not all JSON values are valid for this MIME type. It might be that implementers of services that accept or return JSON may be more liberal and accept and return JSON values. Kofax RPA has chosen to follow this more liberal approach to JSON. To that end, a JSON variable may contain a JSON value and the JSON view can display that value.

When data is loaded from a URL and the MIME-type is application/json, the loaded JSON is shown in the JSON Page View. If this is not the case, you can specify that the data represents JSON. To do this, on the Load Page step, set the Page Content type to JSON. You can also use this method when the data is not loaded from a source where a MIME-type is available, such as in Create Page step action.

JSON and Step Actions

A number of step actions work only on JSON; the data presented in the current window must be JSON (and not JSON in the legacy format where JSON has been translated into XML). These step actions are found in the step action category called JSON in the Step Action Selector on the Action tab of the Step View. But the easiest way to select them is to use the context menu (right-click menu) in the Applications View when the current window contains JSON. See the following sample image.



Two step actions can extract from a JSON value:

- Extract JSON. This step action always extracts a JSON value. For exampl, if the selection in the view is a property, it is the value of the property that is extracted. It is in many ways similar to the Extract step that extracts from HTML and XML, except that it is simpler because of the simpler data format; no distinction exists between markup and text.
- Extract Property Name. This step action extracts the name of a property.

Two step actions can loop over a JSON text:

- For Each Property. This step action loops over each property of a JSON object
- For Each Item. This step action loops over each JSON value of a JSON array.

Both of step actions will for each iteration set a part of the JSON value in question as named JSON (similar to a named tag). This cannot be global when iterating over a variable, as changing the value of a variable during the iteration may change the value in such a way that iteration may fail, such as if an item is removed from the list iterated over.

Four step actions can modify JSON (only if the JSON is in a variable):

- Set JSON. Replaces the selected part of a JSON value with a new JSON value.
- Set Property Name. Sets the property name to a new name on a selected property.
- Insert JSON. Inserts a new property in a JSON object or a new item (JSON value) in a JSON array. There are several options on where to insert the new property or item, such as first or last. Consult the reference documentation for the step action for a full list.
- Remove JSON. Removes the selected part of a JSON value, such as a property from a JSON object or an item from a JSON array.

Finally, two more step actions work on JSON:

- Test JSON. This step action tests the "type" of a JSON value to determine whether it is an object, array, string, etc.
- Set Named JSON. This step action is similar to its corresponding step action for other types of data, such as Set Named Tag and Set Named Range. It defines a named reference to a part of a JSON value so that it can be used as a reference when finding other parts of a JSON value in subsequent steps. Shown as blue boxes in the view.

JSON as a JavaScript Object

Consider a converter stack in a step action that contains a Convert Using JavaScript converter. This converter gets access to the output from the previous converter as a variable named INPUT to use in the JavaScript used by the converter. The value of the INPUT variable is always a String.

The following table shows possible conversion values for the INPUT variable.

INPUT Value	JavaScript (OUTPUT =)	Result (OUTPUT value)
5	OUTPUT = INPUT	5
5	OUTPUT = INPUT + 3	53
5	OUTPUT = eval(INPUT)	5
5	OUTPUT = eval(INPUT) + 3	8
5	OUTPUT = eval(INPUT + 3)	53
5	OUTPUT = eval(INPUT + " + 3")	8
[1,2,3]	OUTPUT = INPUT[0]	[
[1,2,3]	OUTPUT = eval(INPUT)[0]	1
{ “a”: 5 }	OUTPUT = eval(INPUT).a	“Syntax Error”
{ “a”: 5 }	OUTPUT = eval("var x=" + INPUT + "; x;").a;	5

Note the following when converting JSON to JavaScript:

- INPUT is a variable bound to a string value. Therefore, any operation that you perform on INPUT is a string operation. For example, + is string concatenation. That is why INPUT + 3 becomes 53 in the example above.
- The function "eval" only accepts correct JavaScript as input and {"a":5} is not a syntactically correct JavaScript line, but var x = {"a":5} is, which is why the last example above is the one that works.

Browser Window Actions

A *window* holds HTML, XML or other content in a robot. One or more windows are always open, and one window is the *current window*, i.e. the window containing the page that a step-action works on. In Design Studio, each window is displayed in a tab, and the current window is marked with a yellow rectangle.

Windows allow you to handle multiple pages simultaneously. Note, however, that a step can only work on a single page at a time, so you need to change the current window whenever you want to work on another page than the current one.

Using the appropriate step actions in a browser window, you can:

- Open a new window using the New Window action
- Set the current window using the Set Current Window action
- Close a window using the Close Window action

In Design Studio, an easy way to insert a Set Current Window step is to right-click on the window's tab and choose Set as Current Window.

When loading a page that loads other pages (e.g. a page containing a `<frameset>`-tag), each page will automatically be loaded into a separate window.

Each window can have one or more named tags or ranges. Note that each named tag or range belongs to a specific window.

Identifying a Window

Some step actions (for example the ones mentioned above) are configured to operate on a particular window. The window may be identified in three ways:

- by its name as displayed in the window's tab or by the number of the window's tab
- by the found tag
- by a pattern matching against the windows name.

The name is the more stable of the two alternatives in face of robot changes, and also (most subtly) when a step may be reached via different paths that open different windows. Thus the name is the preferred way to identify a window. Matching does not work on windows showing variables, because the names of these are fixed.

In some cases, however, the name is not the same every time the robot is run. For example, some Web sites are based on frames but name these frames differently each time (while keeping the structure of the frame set). Because the window name is derived from the frame's name, the window name is not much use in such a case, and the windows must be referenced by their numbers. In these situations, it is important to make sure that every path through the robot that can lead to the step action in question results in the same window structure and window numbers.

You can also use tags to identify a window. The found tag must be a FRAME, IFRAME, OBJECT or EMBED element. In Design Studio, the list of frames is displayed as a tree in the Frames View. Use the **Window in Found Tag** option in the Set Current Window action to set the current window by the found tag.

There are two alternative ways of identifying a window:

- Specifying a pattern for the window name
- Specifying a pattern for the (text or HTML) content of the window

In both cases, the pattern must be precise enough that only the name of a single window matches it.

Chapter 5

Desktop Automation

Introduced in Kofax RPA 10, Desktop Automation helps you automate any work process involving computer applications such as:

- Native Windows applications
- Native Java applications
- Legacy terminal applications
- Other applications presenting a GUI on a Windows system, such as Citrix clients

See Introduction to Desktop Automation for more information.

Note Desktop Automation service relies on Windows UI Automation API. Do not run any UI Automation API clients on the same computer simultaneously with Desktop Automation Agent.

Also, the *Getting Started with Desktop Automation Guide* is available in the Kofax RPA documentation set that walks you through the process of using the Desktop Automation to build a robot.

Note The Desktop Automation Service cannot automate, remove the focus from, or generate input, such as keyboard input or a mouse click, for an application that has the High integrity level as defined by Windows. The Desktop Automation Service runs as a process with the Medium integrity level and cannot generate input for applications with a higher level. Example of applications running at the High integrity level is the Task Manager, System Properties, or applications ran as administrator.

Also, the Desktop Automation Service cannot generate a tree for applications with a higher integrity level.

Workaround: Run the Desktop Automation Service as administrator to increase the integrity level to High. Or, install the Desktop Automation Service with the virtual input driver and set the environment variable KOFAX_RPA_VIRTUAL_INPUT to Y. For more information on the latter, see "Install the virtual input driver" in the *Kofax RPA Installation Guide*.

A note for Windows 10 users

Windows Start menu is not selectable and does not appear as a tab in the Recorder View when running Windows 10 as a remote device with Kofax Kapow version 10.3.0.1 and later.

Symptom
A robot is significantly slower when Desktop Automation Service (DAS) is running on Windows 10 (or corresponding server versions) compared to Windows 7. Additionally, it is possible that the robot using DAS running on Windows 10 (or corresponding server versions) is not able to identify some elements (such as a window, a popup or other) that can be identified if the DAS runs on Windows 7.

Details

Kofax Kapow version 10.2.0.3 and to version 10.3.0.0 included a patch to fix an issue observed with Universal Windows Platform (UWP) applications also known as Metro-style applications on Windows 8.1 and Windows 10. While looping (enumerating) the open applications on a Windows desktop (which is done five times per second), the traditional method would skip those applications.

With this patch the robots use a method provided by the Microsoft UI Automation API that includes the Metro-style applications, but also has two adverse effects:

- In general this method consumes more CPU power and takes longer, and as the Desktop Automation Service (DAS) is single threaded, it leads to performance degradation across all installations.
- For specific applications, at least in one instance we have seen this method leading to 100% CPU usage for both the DAS (the node.exe process) and the application itself.

Resolution

For versions 10.2.0.3 to 10.3.0.0 there is a method of switching back to the original enumeration algorithm by setting the following environment variable:

```
KAPOWHUB_APPLIST_VERSION=1
```

Follow these steps to add this environment variable:

1. Stop DAS (right click the DAS system tray icon and select the option from the shortcut menu).
2. Open Task Manager and make sure there are no node.exe processes running (stop them, if there are any).
3. In Task Manager, stop all DesktopAutomationServiceControl.exe. This is very important because this process is the DAS Controller (includes the sys tray icon). When DAS is started, the Controller is getting its environment variables from the system and the Desktop Automation Service is getting its variables from the Controller. So for a new environment variable to be used in DAS, the Controller has to be restarted too.
4. Set the Environment variable.
5. Restart Desktop Automation Service (from Start > Programs, it will restart both the system tray app and node.exe).

If this environment variable is added, the performance of DAS running on Windows 8.1 and 10 improves and Metro-style apps are not automated.

Kofax Kapow version 10.3.0.1 improves a few minor performance issues from the previous version and reverts back to using the original enumeration method as default. Therefore, in Kofax RPA version 10.3.0.1 and later there should be no performance problems between Windows 8.1 or 10 and Windows 7, but the Metro-style apps are not automated by default leading to disappearance of the Windows Start menu tab from the Recorder View.

If you want to automate Metro-style apps, set the following environment variable:

```
KAPOWHUB_APPLIST_VERSION=2
```

The steps to set this variable are the same as above. If this variable is set, please be aware this can lead to performance degradation.

Introduction to Desktop Automation

In Kofax RPA, you can create Desktop Automation robots that can automate work processes involving Windows and Java applications on your networked computers. Because Desktop Automation is substantially different from website and database automation, Design Studio has a dedicated workflow language and steps for this purpose.

Desktop Automation Robots

The workflow of a Desktop Automation robot is a sequence of steps that are executed one after the other. The steps model how a user would interact with the application that is being automated. To use a Desktop Automation robot, you need to call it from a Web Automation robot with a dedicated action step named Call Desktop Automation Robot.

A Web Automation robot can contain multiple Call Desktop Automation Robot steps, each with their own workflow. A single Desktop Automation robot can be reused by Web Automation robots, which significantly saves time when you work on multiple robots at the same time. The robot with a Call Desktop Automation Robot step can be executed as any other Kofax RPA robot from a schedule, via the API, via Kapplets or manually during development or testing.

Desktop Automation Workflow

The Desktop Automation workflow is edited in Design Studio. In Design Studio, you can see a view of the robot and the applications being automated along with details on the robot state and a dedicated toolbar with buttons to control the robot manually. See Edit Desktop Automation Robot for details.

On the menu, the Help button is available that includes links to the respective documentation and the Getting Started guides that walk you through the process of using Desktop Automation to build a robot.

Steps

Steps are the basic building blocks of a workflow in a Desktop Automation robot. In Desktop Automation, all steps have one entry point and one exit point, except for a few steps that have no exit point. Some steps are simple steps and merely perform one action such as moving a mouse or pressing a key. Other steps, called composite steps, may contain additional steps. Composite steps are used to group steps that belong together or to handle branching and other ways to control how execution proceeds. For the complete list of steps, see Desktop Automation Step Actions.

Steps in Desktop Automation are typically granular and handle smaller tasks. For example, there is no inherent error handling on every step type. Instead, dedicated steps exist specifically to handle errors during execution.

Devices

The purpose of Desktop Automation is automated control of applications. The applications run on devices (computers, servers or virtual machines) that can be remotely accessed over a network. A robot performs Desktop Automation by connecting to Desktop Automation Agents running on remote devices, unless the device is running a terminal, where the connection is direct from the robot. For details about handling devices and setting up the agents, see Configure Desktop Automation Service.

Application Tree

Kofax RPA provides several ways to populate the application tree. By default, Kofax RPA detects the type of application the robot is working with (such as a Windows application, terminal, built-in browser, and so on) and automatically forms the tree for this application. For some Windows applications, Kofax RPA provides extended support. For example, when working with Internet Explorer in Design Studio, Kofax

RPA turns on extended Internet Explorer support to retrieve DOM (Document Object Model) tree as it provides a more accurate result in the application tree.

To distinguish between attributes that Kofax RPA receives directly from the applications and attributes that Kofax RPA adds, a set of "derived attributes" is provided. This is done to prevent name clashes between different attributes. Kofax RPA adds the bounding box (x, y, width, height) as derived attributes. Derived attributes are displayed in the tree prefixed with "`der_`" and can be used in finders and for extraction.

The following table lists and describes derived attributes that can be used in the application tree.

Derived attribute	Description
For all elements	
der_x	X coordinate of the upper left corner of the element.
der_y	Y coordinate of the upper left corner of the element.
der_width	Width of the element (bounding box of the element).
der_height	Height of the element (bounding box of the element).
der_rendered	Set to "y" (for "yes") when the element is rendered on the page.
der_isOffscreen	Set to "true" when the element is not visible on the screen. The page should be scrolled to make the element visible.
For form control (input) elements	
der_value	Used for the "email," "text," "number," "range," "tel," "time," "url," "search," "date," "datetime-local," "week," "color," "month," and "textarea" input elements.
der_checked	Used for the “radio” and “checkbox” input elements. Can be "true" or "false," depending on whether the element is selected or cleared.

You can turn off extended application support if you experience issues working with a particular application.

Desktop Automation Robots Compared to Web Automation Robots

Kofax RPA was originally designed for accessing HTML at a time when HTML pages were mostly static. In those cases, the state of the application (web page) can be tracked internally in the robot. By contrast, the Desktop Automation functionality is designed to automate remote applications where the state resides in the application. In this case, the state is external to the robot.

Execution of steps in Desktop Automation move forward only. The state of the execution is on the remote device and it is not possible to undo by going back in the workflow, except for a group of Extract Value and Convert Value steps. As a consequence, when designing your workflow, newly inserted steps are not executed until you explicitly select to do so in the Edit Desktop Automation Robot .

Important Branching, as it is designed in Web Automation robots, does not exist in Desktop Automation robots. It only occurs as part of composite steps.

Branching only occurs as part of composite steps, such as the Conditional. The branches are alternative branches, so only one branch is chosen when a workflow is executed. This differs from Web Automation robots where branches are executed sequentially one after the other, and the state is reverted at the start of each branch.

In Desktop Automation, error handling differs because it is not specified for every step. Instead, a try-catch step explicitly catches errors occurring within its scope and defines how to handle them.

In general, when designing a Desktop Automation robot, think how a user interacts with the user interface of the application you are automating. For example, if you need to type some text in the text field, first click the field and then insert a step that types the text.

Desktop Automation has features that allow the robot designer to design the automation to gauge the external state of the application and react appropriately. For example, a click on a button can be made to wait until the button appears. Or a step can detect that an application is already started, to avoid starting another instance. When you design a robot workflow, guards and finders are used to wait for specific states of the application, ensuring that the robot finds the required elements and interacts with them as expected. Guards are described in the Guarded Choice and finders are described in Finders.

See Get Started to begin automating remote devices and Desktop Automation Step Actions for the list of action steps.

Get Started

The following instructions assume that you have downloaded and installed Kofax RPA on one of your computers. See "Quick Start Guide" in the *Kofax RPA Installation Guide* to start using Kofax RPA.

1. Download the Kofax_RPA_DesktopAutomation_11.1.0_x32.msi installation file from the Kofax download portal. Make sure your devices meet the requirements listed in "Desktop Automation requirements and prerequisites" in the *Kofax RPA Installation Guide*.
2. Install and configure Desktop Automation Service on a remote computer that run the applications you wish to automate. If you only need to automate terminal applications, skip this step.
3. Open Design Studio.
4. Create a Desktop Automation robot.
    a. Click **File** > **New Desktop Automation Robot**.
    b. Specify a name for the robot and select a project. Click **Finish**.

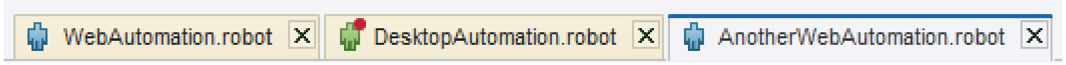
       The new robot appears on a new tab in the editor window. As opposed to Web Automation robots that are identified by a blue icon , Desktop Automation robots are identified by a green icon .

       > **Note** At this point, you can perform basic editing of the Desktop Automation workflow, such as add action steps, configure input values, set variables, and perform other actions that do not involve the use of external, real-time data and applications. To be able to automate external applications and interact with data as well as run a Desktop Automation robot, it needs to be called from a Web Automation robot.
5. Open an existing Web Automation robot or create a new one by clicking **File** > **New Web Automation Robot**.

6. To allow execution for the Web Automation robot, click **Prepare Execution** on the toolbar or in the Applications pane.

   Important Only one Web Automation robot at a time can have the execution privilege, so to take the execution privilege from one robot to another, open the tab with the required robot and click **Prepare Execution**. Also, a Web Automation robot cannot have the privilege to execute in both Design and Debug mode at the same time.

   When a Web Automation robot has the execution privilege, the editor tab of this robot is highlighted. When a Web Automation robot is calling a Desktop Automation robot, the tabs of both robots are highlighted for convenience as shown below. The robot where execution is currently located is marked with a red dot.

7. Insert a **Call Desktop Automation Robot** step in the Web Automation robot.

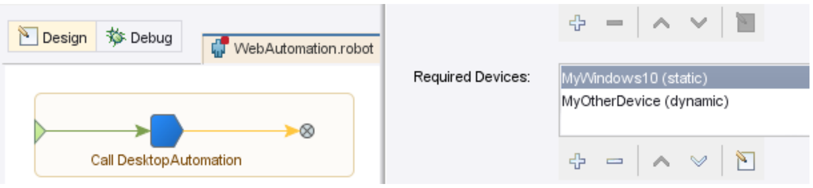
   a. In the **Desktop Automation Robot** drop-down list for the step, select the Desktop Automation robot created in Step 4.

   b. Configure input values, output mappings, and required devices. See Reference to Automation Device and Automation Device mapping. If you want to automate terminal applications only, skip the configuration for required devices.

8. When the execution is allowed, to start editing the Desktop Automation robot, execute the workflow to the Call Desktop Automation Robot step and then click **Step Into DA Robot** on the toolbar.

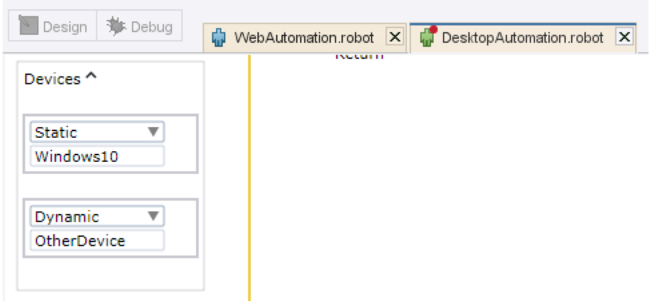
   The tab with your Desktop Automation robot is opened, and you can now perform full editing and execution of the robot.

9. If you are planning to automate external applications, specify the automation devices configured in Step 6 in the Devices box on the left. If you only need to automate terminal applications, skip this step.
   - The number of devices specified in the Web Automation robot and the Desktop Automation robot must match.

   Required Devices: MyWindows10 (static)
   MyOtherDevice (dynamic)

   - The devices names that you set in the Desktop Automation robot may differ from those in the Web Automation robot.

   Static: Windows10
   Dynamic: OtherDevice

10. You can start designing your automation workflow. You can also run the workflow to see it in action.
11. Once you finish designing your Desktop Automation robot, you can run it to automate the devices.
    - To step out of the Desktop Automation robot and switch to working on the Web Automation robot, execute the entire Desktop Automation workflow and then click **Step Out** on the toolbar. In the Web Automation robot, the Call Desktop Automation Robot step is now shown as executed.
    - To close the robot without executing it to the end or returning a result, click **Leave Robot** on the toolbar. In the Web Automation robot, the Call Desktop Automation Robot step is now shown as *not* executed.

Convert old Desktop Automation action step

In Kofax RPA versions 10.7 and earlier, a standalone Desktop Automation Editor was used to edit the Desktop Automation workflow contained in a Desktop Automation action step. Starting with Kofax RPA 11.0, you can execute Desktop Automation action steps created in version 10.7 or earlier, but to edit the workflow, you need to extract the Desktop Automation workflow from the step into a new Desktop Automation robot and change the Desktop Automation step to a Call Desktop Automation Robot step referring to the new robot.

The export can be performed with both Web Automation robots and snippets containing Desktop Automation steps.

Note The export process cannot be undone, so you cannot revert to the initial Desktop Automation step once you exported it to a Desktop Automation robot.

1. To convert the action step to a robot, in the Projects tree, right-click a Web Automation robot containing a Desktop Automation step and click **Export Desktop Automation Robots**. If you open a single robot and select the Desktop Automation step, you can preview the future Desktop Automation robot by clicking **Preview** in the step properties.
   - For convenience when working with multiple robots, you can export several Desktop Automation robots at once. In the Projects view, right-click any folder or select and right-click multiple Web Automation robots containing Desktop Automation steps and then click **Export Desktop Automation Robots**.

   A new dialog box appears listing all found Desktop Automation steps to extract from.
2. If required, you can assign a new descriptive name to the future Desktop Automation robot, preview the future robot, and see the Desktop Automation step in the containing Web Automation robot.
   - Click to rename the selected file.
   - Click to preview the selected Desktop Automation robot to be created after the export.
   - Click to show the selected Desktop Automation step in the containing Web Automation robot.
3. Click **Next** to select a location for the exported robots inside the current project.
4. Click **Finish** to close the dialog box and start the export.

   An export summary appears listing how many robots and snippets have been exported. Any required devices configured in the Web Automation robots automatically appear in the Desktop Automation robots.

Reference to Automation Device

Before editing a robot with a Call Desktop Automation Robot step, provide a device reference by clicking Add in the **Required Devices** field on the **Action** tab of the Call Desktop Automation Robot step. In the **Add Device** window, select **Static Reference**, **Dynamic Reference**, or **Trigger Reference**.

Note that `"local"` reference is always available by default whether any other references are provided or not. `local` reference is used for accessing websites in the built-in browser and working with Excel spreadsheets in the built-in Excel driver.

Note When automating terminal computers, do not install the Desktop Automation Agent and do not provide Automation Device reference.

Static reference

Static reference implies that you created one or more Automation Device mappings to choose from. The robot automates the devices associated with the selected mapping. Mapping information is required for automating Windows devices and it is not required for automating terminals, working with built-in browser and Built-in Excel Driver. If you change the mapping, click **Refresh** in Design Studio to renew the connection. Use static reference when using Local Desktop Automation.

Dynamic reference

This type of reference helps you connect to automation devices in Single User Mode, such as by using the Remote Desktop Protocol (RDP) connection. Specify a mapping name to use in the Connect to Device step. All other connection parameters are specified within the Desktop Automation workflow. Once the Dynamic Reference connection was used by the robot in the Call Desktop Automation Robot step and device is connected, the connection stays alive and you can use this reference (and this device) in the next Call Desktop Automation Robot steps in your robot.

Important You can only connect to the same device once during the execution of a workflow. For example, if you want to connect to a device in a loop, make sure your robot skips the connection step in the loop when connection is already established.

Trigger reference

Trigger reference implies that you created one or more Automation Device mappings to choose from. Attended automation robots connect to devices associated with the selected mapping. When creating a device mapping, specify host, port, and a token of the device. See Map Automation Device for details.

Automation Device Mapping

Mapping of an automation device makes it possible to access this device from a project in Design Studio. We recommend using the Management Console Based Device Mapping option, because this way you can always be sure that the robots you create will work on the dedicated Management Console with the specified Automation Devices despite changes in the network infrastructure. Note that the name of the mapping you create in Design Studio must match the name of the mapping in Management Console. The labels used in Design Studio must match the device you use at design time, and the labels in Management Console must match the device used in production environment.

Important Automation Device Mapping name must start with a letter or an underscore and contain only letters, digits and underscores and cannot be the words "false" or "true".

The Device Mapping option in the Device Mapping Configuration is recommended for developing and debugging your robot in Design Studio.

Note For automating terminal devices, do not install the Desktop Automation Service on your remote computer or create a device mapping in the Management Console.

Map Automation Device

1. Click **New Automation Device Mapping** on the **File** menu or right-click a project in the **Projects** list and select **New** > **Automation Device Mapping**.
2. If you started the Automation Device Mapping wizard from the File menu, provide a name and select a project that the device will be associated with. Otherwise, provide a name of the Automation Device. Click **Next**. For the Management Console based mapping, the name of the mapping you create in Design Studio must match the name of the mapping in Management Console.

3. In the Automation Device Mapping Configuration step, select either **Management Console Based Device Mapping** or **Device Mapping**.

   Management Console based device mapping
   This option helps you connect to Automation Devices using mappings in the Management Console. Configure the following.

   - **Management Console**: Select the Management Console to use the mappings from.
   - **Cluster Name**: Type the cluster name on the selected Management Console.
   - **Required Labels**: Type one or more labels for the Automation Devices. The labels you specify must match the device you use at design time. The labels must be separated by commas.

   Device mapping
   This option helps you connect to Automation Devices directly. Configure the following.

   - **Host**: Type the Automation Device host name or IP address.
   - **Port**: Type the Command port number to connect to the Automation Device.
   - **Token**: Type the token specified in the remote hub settings on the selected Automation Device.

Configure Device Mapping

To edit the Automation Device Mapping, either double-click the device mapping in a project, or right-click the device mapping and select **Configure**. In the **Configure Device Mapping** window, select either **Management Console Based Device Mapping** or **Device Mapping**.

Management Console Based Device Mapping
This option helps you connect to Automation Devices using mappings in the Management Console. Configure the following.

- **Management Console**: Select the Management Console to use the mappings from.
- **Cluster Name**: Type the cluster name on the selected Management Console.
- **Required Labels**: Type one or more labels of the Automation Devices. The labels must be separated by commas.

Device Mapping
This option helps you connect to Automation Devices directly. Configure the following.

- **Host**: Type the Automation Device host name or IP address.
- **Port**: Type the port number to connect to the Automation Device.
- **Token**: Type the token specified in the remote hub settings on the selected Automation Device.

Edit Desktop Automation Robot

When working on a Desktop Automation robot, the following panes are present in the editor. You can undock and move any of the editor's windows to make editing more convenient. Also, see Get Started.

- **Editor**: Contains a workflow of steps as well as variables, expressions, and tree mode settings. Buttons on the toolbar below the menu help you navigate the steps of the workflow. You can go forward in the workflow using the Start Execution, Step Into, Step Over, and Step Out buttons; you can also pause or reset the execution of the robot. Use the following key combinations to select several steps:

- Shift+click: selects a range of steps
- Ctrl+click: adds/removes selection of steps
- Ctrl+Shift+click: adds a range selection

Button	Description
Open project	Opens a project.
Save All	Saves changes in the workflow.
Undo	Reverts the last change.
Redo	Repeats the reverted action.
Copy	Copies a selected element or elements of the workflow to the clipboard. For example, several steps or a finder.
Cut	Cuts selected steps.
Paste	Pastes the content of the clipboard into the workflow at the selected element. For example, a flow point or a finder.
Delete	Deletes selected steps.
Leave Robot	Stops and leaves the Desktop Automation robot without executing it to the end or returning a result.
Start Execution	Starts executing the workflow from the current flow point.
Pause	Pauses automation workflow execution.
Step Into	Executes to the next flow point. In case the next flow point is inside a collapsed step, this step will be expanded. This is essentially related to single stepping case.
Step Over	Executes the step immediately following current flow point. If there is no such a step, it executes to the next flow point.
Step Out	Executes to the flow point immediately following the step containing the current flow point. When the workflow is executed until the end, clicking this button steps out of the Desktop Automation robot.
Go to Next Iteration	Enabled when the current flow point is inside a Loop step. Press the button to execute until the same flow point is reached again. The loop can be executed more than once if the flow point is skipped in some iterations. If there are no more iterations, the execution stops at the flow point outside the Loop step.
Reset	Resets the execution of the Desktop Automation robot.

Button	Description
Go To Current Flow Point	Brings to view the current flow point location.
Collapse	Collapses all steps and other elements in the editor pane.
Expand	Expands all steps and other elements in the editor pane.

Button	Description
Collapse all but selection	Collapses all steps and other elements except the selected ones.

Newly inserted steps are not executed unless you click **Step Into** or **Step Over** on the toolbar.

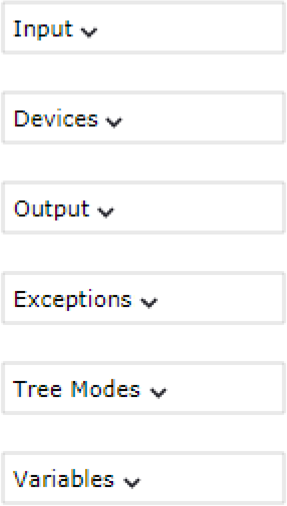
Zoom in the workflow view

For your convenience, you can zoom in and out in the workflow view the same way you zoom in a web browser.

- To zoom in use: Ctrl+plus sign (+) or Ctrl + mouse wheel scroll up
- To zoom out use: Ctrl+minus sign (-) or Ctrl + mouse wheel scroll down

- **Left Side**:

  Contains attributes of the Desktop Automation robot.

  - **Input**: Lists inputs that a Desktop Automation robot takes.
  - **Devices**: Use this element to provide a reference to one or several required devices. For more information, see the Reference to Automation Device section.
  - **Output**: Lists outputs that a Desktop Automation robot delivers to the Web Automation robot.
  - **Exceptions**: Lists unhandled exceptions that may be thrown when a Web Automation robot is calling.
  - **Tree Modes**: Use this element to configure the application tree type for different applications to deliver. For more information, see the Tree Modes section.
  - **Variables**: Use this element to configure the Desktop Automation robot variables. For more information, see the Variables section.
- **Recorder View**: Shows tabs with open application windows and a tree with available elements. You can select elements in the interface or select images and insert steps by right-clicking the selected

element or image. The bottom part of the Recorder view shows the coordinates of the mouse relative to the top left corner of the application window as well as the live streaming status of the device state.

When you select an element in the view, along with the window coordinates, it shows the coordinates relative to the top left corner of the selected element as well as the path to the element in the bottom bar.

All tag paths in the Recorder View are interactive. Left-click a tag in the path to make the node a selected tag, right-click a tag to open the context menu for the node.

Tags are marked with colored boxes in accordance with their current state:

- A green box around a tag: currently selected tag.
- An orange box around a tag: secondary tag.
- A blue box around a tag: ternary tag.

To switch between elements of different levels, use Select Outermost Tag, Select Tag One Level In, Select Tag One Level Out, Select Innermost Tag, Select Previous Sibling Node, Select Next Sibling Node buttons.

> **Note** Sometimes it is not possible to select cell elements in tables in the application view. You can select cell elements in the application tree and add step actions from the tree view.

Application-level actions

Application-level actions are actions applied to the entire application and available by right-clicking the application tab in the **Recorder View**. The list of application-level actions and steps includes the following:

- Enter Text
- Press Key
- Scroll
- Guard
- Trigger
- Tree Mode
- **Application Action**
  - Focus: This action is available for remote desktop applications.
  - For locally running applications, such as Excel, Document Transformation Browser, website browser, and terminal emulators the list of application actions varies. See the related topics for details.

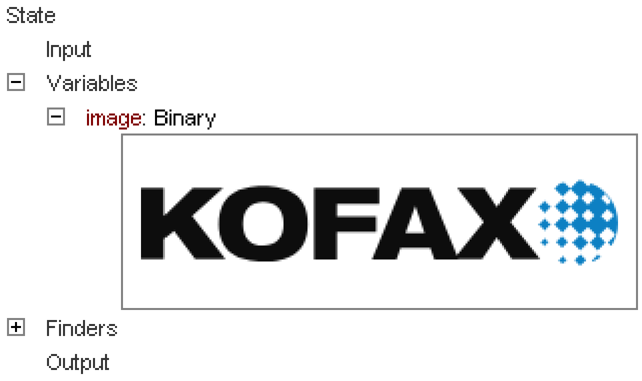
You can zoom in and out in the **Recorder View** either by selecting a zoom level in the toolbar or the same way you zoom in a web browser.

- To zoom in use: Ctrl + mouse wheel scroll up
- To zoom out use: Ctrl + mouse wheel scroll down

Button	Description
❚❚ Pause ▶ Resume	Pauses or resumes live streaming of the device state. Click to pause or resume streaming in the Recorder view. When streaming is active, the following is shown under the Recorder view: Stream LIVE. When streaming is paused, the following is shown under the Recorder view: Stream PAUSED.
Create finder for selection	Replaces the selected in the editor pane finder with a finder that matches the selection in the **Recorder View**.
Show next location found	Shows the next element that matches the finder. The button's tooltip also provides the number of matched elements.
Select Next Node Matching Click	Moves selection to the next node that matches the selection in the **Recorder View**.
Toggle between Simple and Nine Grid Image Finder	Changes image selection from simple to nine-grid and back.
x1 ▾ Select zoom level	Zooms in and out in the **Recorder View**.
Select Tag One Level Out	Changes selection to the node, which is parent to the selected one.
Select Tag One Level In	Selects the first child node of the selected one.
Select Outermost Tag	Changes selection to the top node in the application tree.
Select Innermost Tag	Selects the last child node of the selected one.
Select Previous Sibling Node	Selects the previous node located on the same level in the application tree.
Select Next Sibling Node	Selects the next node located on the same level in the application tree.
Copy Sub Tree As XML	Copies the subtree element selected in the tree view.
○ Auto Execute	Immediately executes and streams newly added action steps. When enabled, the circle on the button is red. Prior to adding a step or a number of steps, click this button. If a step opens a new application or a dialog, a respective tab appears in the stream view and becomes the active tab. To stop automatic execution, click this button again.

- **Desktop Automation State** view: Shows state of the workflow execution, such as variable values. When Kofax RPA detects that a binary variable contains an image, the image is displayed in the view.

Kofax RPA detects GIF, JPEG, BMP, TIFF, and PNG images. The image tooltip shows the MIME type of the image and its size (width and height).



- **Output Log**: Contains workflow execution messages.
- **Comment**: Shows comments for steps in the automation workflow.

  To write or change a comment, click a step or a Group step and add/change your notes in the Comment window. You can use the Undo and Redo buttons here. The comment is automatically saved when you click outside the window.

  A step that contains a comment is marked with a comment sign.

Editor Procedures

Put focus on errors in workflow
When a step is inserted from a flow point in the Desktop Automation robot, and that step contains an error or multiple errors, the location of the first error is indicated in the expanded step. If the robot execution is stopped due to errors in a step, that step is also expanded to show the location of the first error. If any steps were inserted in the workflow using the "Surround with" type of steps, those steps are expanded, too, for convenience.

Configure Desktop Automation Service

This chapter describes Desktop Automation Service configuration.

Desktop Automation Prerequisites

All Desktop Automation requirements and prerequisites are listed in the "Dependencies and Prerequisites" chapter of the *Kofax RPA Installation Guide*.

> **Note** Desktop Automation service relies on Windows UI Automation API. Do not run any UI Automation API clients on the same computer simultaneously with Desktop Automation Agent.

Configure the Desktop Automation Agent

Once your computers meet all the necessary requirements for Desktop Automation, you can install and configure the Desktop Automation Agent.

1. If you need to automate Java applications, install Java 32-bit (JRE or JDK) on remote devices and check that the Java Access Bridge is enabled on your devices. See Check Java Access Bridge for details.
2. Download and run the Kofax RPA Desktop Automation installer on your device.
3. Start the Desktop Automation Service from the Start menu. Once the service starts, you can see its status by looking at the icon in the notification area.

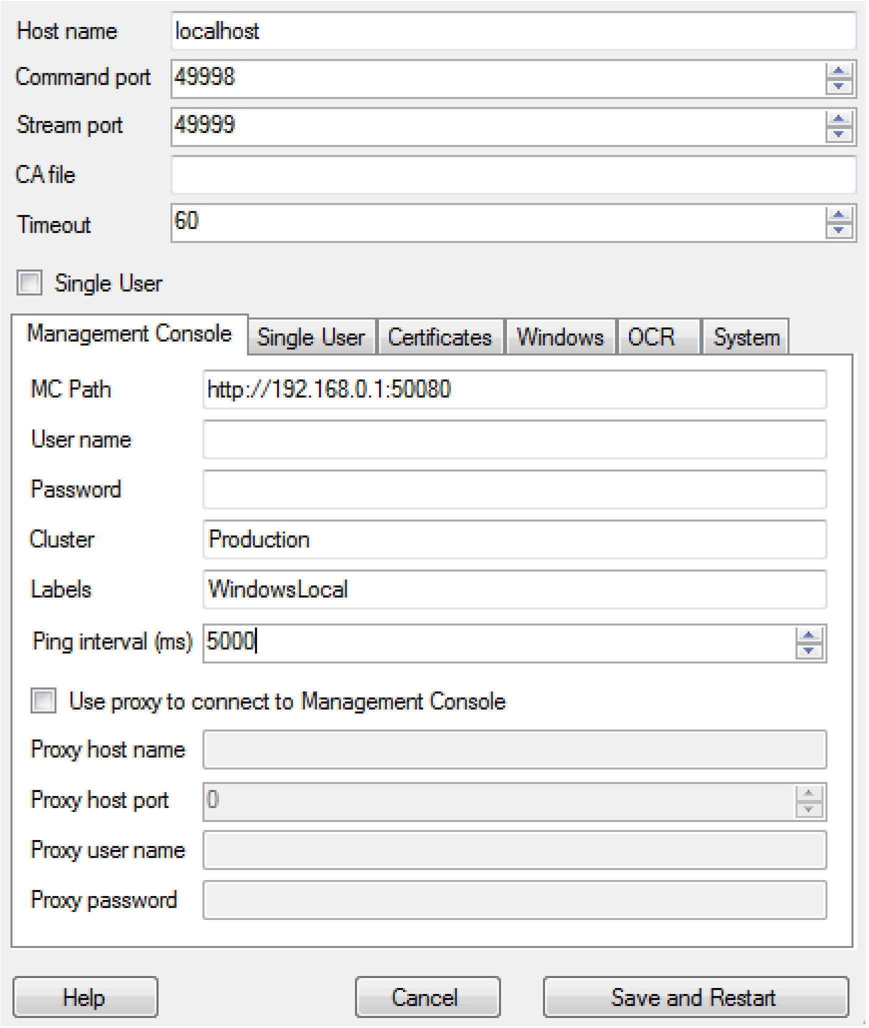
Icon	Status
	Desktop Automation Service is starting and trying to connect to the configured Management Console.
	Desktop Automation Service is running and either connected to a Management Console or running in single user mode depending on configuration.
	Desktop Automation Service is running and in use by a RoboServer or Design Studio.
	Desktop Automation Service is not running.
	Desktop Automation Service is not running due to an error.

4. To edit the Desktop Automation Service options, right-click the Desktop Automation Service icon in the notification area and select **Configure**. This opens the Desktop Automation Service window. After changing the options, click **Save and Restart**.

   To manually edit the options, open the `server.conf` file on your automation desktop. The file is located in Users > UserName > AppData > Local > Kofax RPA 11.1.0 folder where UserName is the name of the user the service is running under.

   See the table with Desktop Automation Service options below.
5. Check that the device is registered in the Management Console under **Admin** > **Devices** tab.

The following is a Desktop Automation Service configuration window.



Host name	localhost
Command port	49998
Stream port	49999
CA file	
Timeout	60

□ Single User

MC Path	http://192.168.0.1:50080
User name	
Password	
Cluster	Production
Labels	WindowsLocal
Ping interval (ms)	5000

□ Use proxy to connect to Management Console

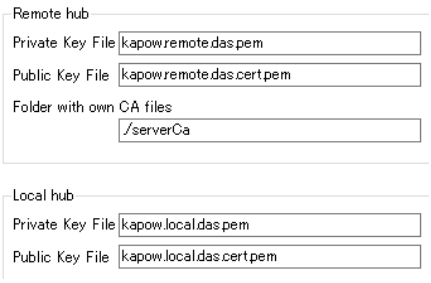
Proxy host name	
Proxy host port	0
Proxy user name	
Proxy password	

The following table lists the available Desktop Automation Service options.

Configuration Window Option	server.conf Option	Value and Description
Single User Select for direct connection to the automation desktop from Design Studio or when using the RDP connection.	"singleUser"	Clear (default) Leave empty to automatically register the Desktop Automation Service with the specified Management Console. For direct connection to the automation desktop, select the option and specify a token.*

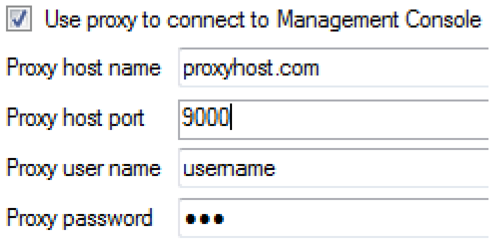
Configuration Window Option	server.conf Option	Value and Description
Host name	"hostName"	Name or IP address of the computer running the Desktop Automation Service. If a computer has multiple names or IP addresses, specify the one that RoboServers and Design Studio contact this Desktop Automation Agent with. That is, the host name or IP address must be reachable from RoboServers and Design Studio.
Command port	"commandPort"	`49998` (default) If the Desktop Automation Service is started without being manually configured, it uses the default configuration and listens on the default 49998 port. Reassign this port to the automation desktop if necessary.
Stream port	"streamPort"	`49999` (default) This port is used to send data between Design Studio and Desktop Automation Service. If streamPort is set to "0", the Desktop Automation Service selects a random port number. You might need to reassign the streamPort if there is a firewall between Design Studio and the automation desktop.
CA file	"caFile"	empty (default) You can communicate with the Management Console using SSL. If the default certificate in `node.js` is not used, you can specify a path to another certificate file using this parameter. Note that you need to have a root certificate for this to work. To save a root certificate in a file from a Google Chrome browser, do the following. **1.** Right-click the lock icon in the address bar, and click **Certificate (valid)**. **2.** On the **Certificate Path** tab, select the top most (root) certificate, and click **View Certificate**. **3.** On the **Details** tab, click **Copy to File**, then complete the wizard to export the root certificate as a base-64 encode X.509 certificate. Now you can specify the path to the file with exported certificate.

Configuration Window Option	server.conf Option	Value and Description
Timeout	"commandTimeout"	This option specifies the timeout for command execution in seconds. A command is an instruction sent to automation desktop, such as click mouse button, open application, add a location found guard, and so forth. If a command cannot be completed in a specified time, the service sends a notification and execution of the robot stops. Note that in case of a Guarded Choice step, this setting applies to invoking the guard in the workflow, but waiting for the guard to be satisfied is not bound to this timeout and can wait forever. Similar situation occurs when using the Move Mouse and Extract steps. The commands must be invoked on the device withing the timeout specified in this field, but the robot waits for up to 240 seconds for the commands to complete. The command timeout for automating terminals or browsing websites in Desktop Automation robots is set either on the **Desktop Automation** tab of the Design Studio Settings window for executing the workflow in Design Studio, or in the **Desktop Automation** section on the **Security** tab of the **RoboServer Settings** window for RoboServer execution.
Token on **Single User** tab	"token"	empty (default) If the Single User option is not selected, leave this option empty. If you use the direct connection to the automation desktop (Single User is selected), specify a token. It can be any token you define.

<table>
<thead>
<tr><th>Configuration Window Option</th><th>server.conf Option</th><th>Value and Description</th></tr>
</thead>
<tbody>
<tr>
<td><b>Certificates</b> tab

Remote hub
Private Key File kapow.remote.das.pem
Public Key File kapow.remote.das.cert.pem
Folder with own CA files ./serverCa
Local hub
Private Key File kapow.local.das.pem
Public Key File kapow.local.das.cert.pem</td>
<td>"tlsServerConfig"</td>
<td>Kofax RPA provides TLS communication between automation desktop and RoboServer or Design Studio. The communication uses certificates for encrypting the communication. The following is a <code>server.conf</code> file code extract. For more information see Use TLS Communication.
<pre>
 "tlsServerConfig": {
   "key": "kapow.remote.das.pem",
   "cert":
"kapow.remote.das.cert.pem",
   "ca": "./serverCa"
 },
</pre></td>
</tr>
<tr>
<td><b>Windows</b> tab</td>
<td>"automationnative"</td>
<td><ul>
<li><code>"useLegacy"</code>
In some situations, the Java Access Bridge does not work and it can help to switch to legacy mode. Default is <code>false</code>.</li>
<li>Installed packages
Lists Desktop Automation Service packages installed on this computer. Starting from version 10.7, new version packages are installed automatically if the <b>Lock package</b> option below is not selected. The packages in ZIP files are installed to <code>C:\ProgramData\Kofax RPA</code> on the automated computer. The appropriate package is selected automatically depending on the RoboServer version. If you want to specify only one version package to be used, select <b>Lock package</b> and select one of the installed packages.</li>
<li>Lock package
When selected you can choose a version package as the only one to work with. A RoboServer with a different version cannot connect to this service. Default: the option is clear or <code>false</code> in the server.conf file.
Select this option or change the setting to <code>true</code> in the server.conf file to run robots with triggers.</li>
<li>Map RFS share to drive letter
The Windows drive that the Robot File System file share is available in. When the file share is mapped to a Windows drive, other Windows applications can also access this file share.</li>
</ul></td>
</tr>
</tbody>
</table>

Configuration Window Option	server.conf Option	Value and Description
OCR tab	"ocrConfig"	"defaultLanguage": "eng" "ocrEngine": "tesseract", Specifies one or more languages and an engine to perform an OCR operation. You can choose from Tesseract (default) and OmniPage OCR engines. To use the OmniPage engine, select **Use Kofax OmniPage OCR (only for robots created in RPA 11.1 or later)** on the **OCR** tab. Specify OCR languages in the **Enabled OCR languages** field. For example, if you want to use the Japanese language, either replace `eng` with `jpn` or, if you want to use more than one language, add `jpn` using the plus sign, such as `eng+jpn`. For Tesseract, Kofax RPA installs only the English language. See Change Default OCR Language for Desktop Automation below for language installation instructions.
System tab		This tab helps you open the log file to examine for any errors and shows the version and location of the service file. Using this tab, you can check whether Java Access Bridge is properly installed on the computer where the service is running. See Check Java Access Bridge for details.
Management Console Options		
MC Path Connection protocol, name or IP address, port number, and path of the Management Console the device must register with. The format is as follows: `http://10.10.0.136:50080`.	"hostName"	Name or IP address of the Management Console the device must register with.
	"port"	Connection port of the specified Management Console.
	"schema"	Connection protocol of the specified Management Console.
	"path"	empty (default) The part of the path to the standalone Management Console after the port number. For example, if your Management Console is deployed on Tomcat at http://computer.domain.com:8080/ManagementConsole/, specify `"/ManagementConsole/"` in this parameter. Leave this parameter empty for the embedded Management Console installation.
User name	"user"	empty (default) User name to authenticate on the specified Management Console.

Configuration Window Option	server.conf Option	Value and Description
Password	"password"	empty (default) Password to authenticate on the specified Management Console.
Cluster	"cluster"	Production (default) Cluster name on the specified Management Console.
Labels	"labels"	"label1,label2" (default) Labels to distinguish the automation devices.
Ping interval (ms)	"pingInterval"	5000 (default) Time interval for the Desktop Automation Service to ping the Management Console.

Use proxy to connect to Management Console	"useProxy"	Select this option for the Desktop Automation Service to use proxy when connecting to Management Console. All necessary parameters are specified in the following fields.
		
		☑ Use proxy to connect to Management Console
		Proxy host name: proxyhost.com
		Proxy host port: 9000
		Proxy user name: username
		Proxy password: ●●●
		Under Linux, you can set up proxy parameters in the `managementConsole` section of the `server.conf` file.
		`    "useProxy": true,`
		`    "proxyHostName":`
		`"proxyhost.com",`
		`    "proxyPort": 9000,`
		`    "proxyUserName": "username",`
		`    "proxyPassword": "pwd"`

* The direct connection to the automation desktop is recommended for creating and debugging a robot in Design Studio as well as for using with RDP connection.

Logging for Desktop Automation Service

Kofax RPA collects usage information on specific Desktop Automation Service events, which may be useful to improve the service performance.

- If the Desktop Automation Service is connected to a Management Console, the events are stored in the RoboServer Log Database of the Management Console. To view the events, on the **Log view** page, select **DAS messages**.

  Note When the connection parameters for the Management Console are specified in the Desktop Automation Service configuration window, the events are always logged to the Management Console, even if the Single User mode is selected, that is, the connection to the automation desktop is established directly, without the Management Console.

- If the Desktop Automation Service cannot connect to a Management Console (as Management Console is not configured), it writes the events to the **Desktop Automation Service Usage.csv** log file, which resides in: `{path}\AppData\Local\Kofax RPA\<version number>\Logs\`

  The file location can be configured in the **log4net.xml** file.

The information for each event includes:

- Time that the event occurred (in UTC).
- Type of event: start, stop, connect, disconnect, suspend or lock screen.
- Identification of Desktop Automation Service, consisting of an ID in the form `host:port`, the user account running the service, and the labels defined for the service.
- Name of the robot and the execution ID (only for connect and disconnect).
- Severity indication (always "Info").
- Message (always empty).

Configure Proxy Servers in Desktop Automation

All Desktop Automation Service robots can use the Kofax RPA global proxy settings. The Desktop Automation Service uses the same proxy settings as Design Studio and Management Console. There are two ways to configure proxy server settings.

Important The local proxy settings of the built-in browser in the Desktop Automation Service have a higher priority than the Kofax RPA global proxy settings configured in Design Studio > Design Studio Settings. Make sure that the robot uses the Kofax RPA global proxy settings, unless the task requires it to use local proxy settings.

Also, the cef.cfg file should not be used to configure proxy settings, but if it is used, it has a higher priority than all of the above proxy settings.

1. For all robots running in the Desktop Automation Service, in the Design Studio Settings dialog box, on the Proxy Servers tab, complete the following Proxy Server details.
   - Host
   - Port number
   - User name
   - Password
   - Excluded host names
2. For all deployed robots, on the Management Console > Admin > RoboServers > Settings > "Proxy servers" tab, select New proxy and complete the following proxy server details.
   - Host name
   - Port number
   - User name
   - Password
   - Excluded host names

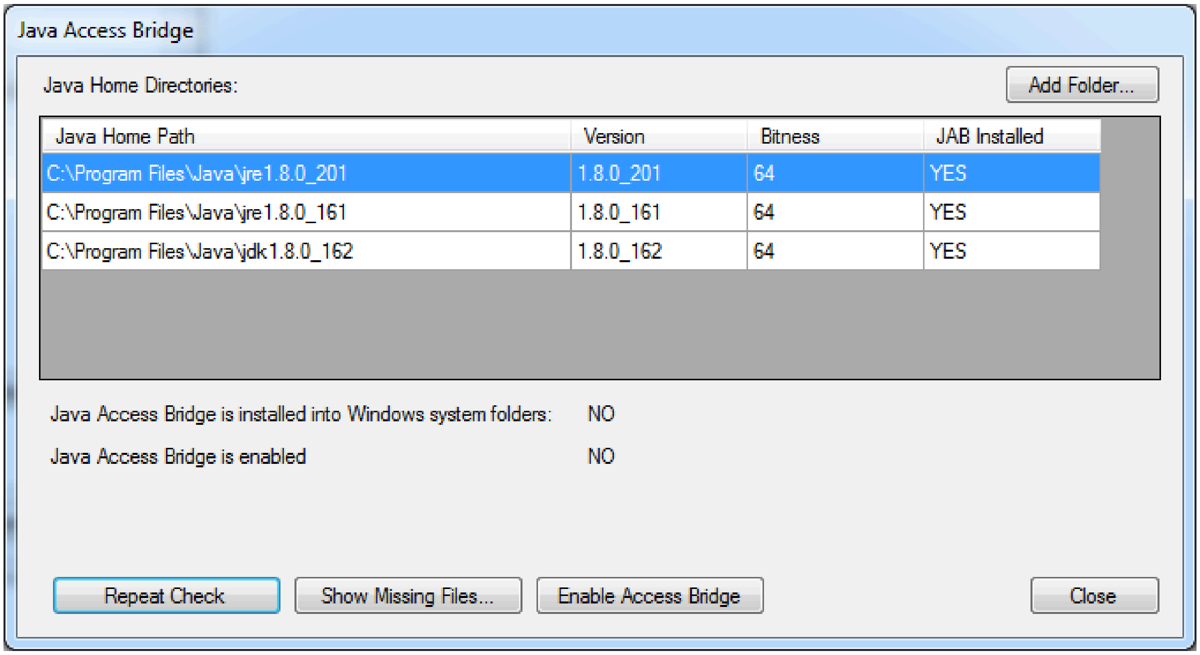
Check Java Access Bridge

Java Access Bridge is an essential component to automate your Java applications. Depending on the Java version, some necessary files may be missing in system folders and Java Access Bridge may be disabled on the computer where the Desktop Automation Service is installed. To check your Java Access Bridge installation, perform the following steps.

1. Right-click the Desktop Automation icon in the notification area and select **Configure**.
2. Click the **System** tab and click **Check Java Access Bridge files**.

   The **Java Access Bridge** dialog box opens showing installed Java versions and Java Access Bridge installation status for each version. If **JAB Installed** column, **Java Access Bridge is installed into**

Windows system folders, and **Java Access Bridge is enabled** show **YES**, Java Access Bridge is properly installed and enabled on the computer.



Java Access Bridge

Java Home Directories:

Java Home Path	Version	Bitness	JAB Installed
C:\Program Files\Java\jre1.8.0_201	1.8.0_201	64	YES
C:\Program Files\Java\jre1.8.0_161	1.8.0_161	64	YES
C:\Program Files\Java\jdk1.8.0_162	1.8.0_162	64	YES

Java Access Bridge is installed into Windows system folders: NO

Java Access Bridge is enabled NO

3. If your implementation of Java is not listed under **Java Home Directories**, click **Add Folder** and specify a home folder with installed Java files.
4. If any of the files are missing, such as **JAB Installed** column shows **NO**, click **Show Missing Files**.

   The **Java Access Bridge Missing Files** dialog box shows files that must be copied to specified folders. Click **Install Missing Files** to install the latest version of the Java Access Bridge files supplied by Kofax RPA in the Desktop Automation Service installation.
5. If **Java Access Bridge is enabled** shows **NO**, click **Enable Access Bridge**.

Change Default OCR Language for Desktop Automation

Kofax RPA uses either the Tesseract or OmniPage OCR engines to capture text from images. For Tesseract, Kofax RPA installs only the English language, while OmniPage includes all supported languages in the installation. When your robot performs text recognition in the Extract Text From Image step using Desktop Automation, Kofax RPA uses the language selected on the **OCR** tab of the Desktop Automation Service window. To change the default language for OCR, perform the following steps.

1. Right-click the Desktop Automation icon in the notification area and select **Configure**.
2. Click the **OCR** tab and type the language code of the language you want to use for OCR in the **Enabled OCR languages** field. The language code must be in ISO 639-3 or ISO 639-1 format. To use more than one language, add another language using the plus sign, such as `eng+jpn`.

   > **Note** Using more than one language simultaneously for screen recognition slows down robot execution and deteriorates recognition results.
3. Click **Save and Restart**.

If you use Tesseract for text recognition of the language other than English, you must first download and copy necessary language packs as follows.

1. Download the `.traineddata` file for the required language from the https://github.com/tesseract-ocr/tessdata. For example, the file for the French language is `fra.traineddata`.
2. Copy the downloaded trained data file to `Kofax RPA\<version>\lib\tessdata` in the **ProgramData** folder. Example:

   `C:\ProgramData\Kofax RPA\11.1.0_110\lib\tessdata`

You can train Tesseract to recognize your character set using either TTF fonts or UI screen shots. See Train Tesseract for more information.

Activate the virtual input driver

Virtual input driver is a Windows device driver capable of simulating hardware keyboard and mouse. For the operating systems supported by the driver, see the *Kofax RPA Technical Specifications* document available on the Kofax RPA documentation site. See the *Kofax RPA Installation Guide* for information on installing the driver.

> **Note** The virtual input drivers do not function when the desktop of the automated computer is being locked, such as by the Lock Screen function or an RDP step.

To enable the virtual input driver for keyboard and mouse operations on the automated computer, set the environment variable `KOFAX_RPA_VIRTUAL_INPUT` to `Y`. To cancel the virtual input driver usage, set the environment variable to `N`.

Use Local Desktop Automation

The Local Desktop Automation feature helps you design and run your robots on the same computer as the devices you wish to automate, which accelerates and facilitates the process of Desktop Automation. Local Desktop Automation is only supported on the Windows operating system.

To enable Local Desktop Automation, follow these steps:

1. Install the Desktop Automation Service and Design Studio on the same computer, which is also the computer that runs the applications you wish to automate. For more information, see the *Kofax RPA Installation Guide*.

   > **Tip** If your computer is set up with dual monitors, you can open the automated applications on one monitor and Design Studio on the other for convenience.

2. Configure the Desktop Automation Service as described in Configure Desktop Automation Service. When specifying the properties in the Desktop Automation Service configuration window, we recommend that you select the **Single User** option to set up a direct connection to the automated applications from Design Studio. Do not forget to type tokens to use for mapping.

When finished, follow these steps:

1. Create a mapping to the Desktop Automation Service installed on a local computer.
   a. Right-click the project you are working on and click **New** > **Automation Device Mapping**.
   b. Fill in the fields as described in Map Automation Device and click **Finish**.
2. Open Design Studio.
3. Create a Desktop Automation robot.
   a. Click **File** > **New Desktop Automation Robot**.
   b. Specify the name of the robot and select a project. Click **Finish**.

      The new robot appears on a new tab in the editor window. At this point, you cannot edit the Desktop Automation workflow as you need to first call the new robot from a Web Automation robot.
4. Open an existing Web Automation robot or create a new one by clicking **File** > **New Web Automation Robot**.
5. Insert an action step.
   a. Click **Select an Action** on the **Action** tab and choose **Call Desktop Automation Robot**.
   b. In the **Desktop Automation Robot** drop-down list, select the Desktop Automation robot created in Step 4.

      On the same tab, configure input values and output mappings.

      - In the **Required Devices** property, click the plus icon, select **Static Reference**, and then select the mapping you made in Step 1.
      - Click **OK**.
6. Execute the newly added action step by clicking **Start Execution** on the toolbar.

   After you execute the **Call Desktop Automation Robot** step, you can edit the workflow itself. To do so, click **Step Into DA Robot** on the toolbar.

   The tab with your Desktop Automation robot is opened and the editor is now active. You can now start designing the Desktop Automation robot. A notification is shown that the Local Desktop Automation mode is enabled.
7. In the **Recorder View**, select a tab with the application to automate. It must already be open on the computer, or you can add an Open action step to your robot that opens the application. Now you can create steps that you wish to perform on the application.
   - If you need to automate application elements that disappear when you remove the pointer, such as context and drop-down menus, use a Bundle step. A Bundle step connects a number of

steps that you perform on the automated application and turns them into a sequence that will be executed in order, starting with the first step.

- To wrap existing steps in a Bundle step, in the editor, select the steps containing the use of disappearing elements, right-click the group, and then click **Surround with Bundle step**. Also, you can insert a Bundle step directly inside the workflow and then add the necessary steps to it.
- To insert a Bundle step that contains a right-click or left-click action or points to an application component, in the **Recorder View**, right-click the required component of an application and click **Smart Focus Menu Click**. Click **Right**, **Left**, or **Hover**, respectively.

To add action steps to the Bundle step, right-click the flow point inside it and make a selection. Some steps are not available inside the Bundle step, but you can add them to your robot before or after the Bundle step.

> **Tip** To execute a Bundle step from its beginning to a particular flow point inside it, double-click the flow point or right-click it and then click **Execute to here**. If you use the Step Over or Start Execution buttons on the toolbar, the Bundle step is always executed from beginning to end.

- You can immediately execute and stream newly added action steps. Prior to adding a step or a number of steps, under the **Recorder View**, click **Auto Execute** (the circle on the button becomes red). If a step opens a new application or a dialog box, a respective tab appears in the stream view and becomes the active tab. To stop auto execution, click **Auto Execute** again.

8. When you are automating several applications in your robot, switch the focus among them. By default, when the execution starts, the focus is set on the first application being automated in the robot. To change or switch the focus to the other automated applications, add a Click action step to each of them. You can add a step that clicks inside the application, or one that clicks the application on the Windows taskbar.
9. Save the changes. To execute the created workflow, click the **Start Execution** button.

   When the workflow execution starts, the focus is instantly switched to the automated applications, and the stream status is changed to **LIVE**, which is shown in the bottom right corner of the **Recorder View**.

   When the execution is completed, to return the focus to the Desktop Automation workflow, click inside it. The stream status will be changed to **PAUSED**. When the stream is paused, the application status is not updated in the Recorder View.

   > **Note** During the execution of the Desktop Automation steps, the keyboard and mouse are automatically blocked to prevent accidental interaction with the execution, and then unblocked when it is completed. If you need to use the keyboard and mouse during the execution, press Esc.

A robot created in the Local Desktop Automation mode can then be edited and run on a remote computer, just like any other robot.

Variables

The Desktop Automation workflow can take input from a Web Automation robot and use the same variable types with some additions.

- You can use a complex variable and its attributes as input to the Desktop Automation workflow.

- You can use all complex types from your project except variables that contain Date, Currency, Country, and Language fields to create Record variables in the Desktop Automation workflow.

  Record variables are based on the record types (complex variables) that a robot uses. A record type is a data type that can be defined as a set of fields, represented by any possible value and a variable of that type.

  For example, a "Credentials" record type consists of two fields: a **Username** and a **Password**. If a user creates a variable of "Credentials" type (for example, "userCred"), the fields can be defined as follows: a **Username** is "Joe" and a **Password** is "Password". This information is represented in the **State view**. Then the value of the variable can be changed by assigning another value (for example, "Tom") to `userCred.username` in the **Assign step**. This changes the **Username** field to "Tom", while the **Password** field remains unchanged.
- You can assign record variables of the same record type to each other.
- You can assign attributes of Record variables to values with the same type or to another attributes or simple variables with the same type.

  > **Note** All the shortcut menus in the Recorder view that use variables let you choose a field from Record type variables.
  >
  > For example, the Enter Text can get its value from fields of complex variables and if the type of the selected variable is not text, a conversion function is inserted to convert the value to text. Variables and fields of types that cannot be converted to text do not appear in the list.
  >
  > For the extract step you can also extract to fields of complex variables, but the type of the field must match the type of the extracted data, such as text or binary (for images). Only variables of the correct type appear on the menu.

- You can edit existing variables without resetting the execution state of the workflow.
- You can create new variables from the context menu without interrupting your work and scrolling to the variables section.
- Date type variables from a Web Automation robot cannot be used as input in the Desktop Automation robot.
- Local variables can be created and used only in Group Steps. If you want your step to use a local variable, include the step into the group with the local variable.
- Password type variables in Desktop Automation can transfer their value from and to a password type variable created in a Web Automation robot. You cannot manually assign a value of the password type variable in the Desktop Automation workflow.

The following is a list of variable types available by default with their initial values.

- Binary: Empty
- Boolean: false
- Integer: 0
- Number: 0.0
- Password: Empty
- Text: ""

Finders

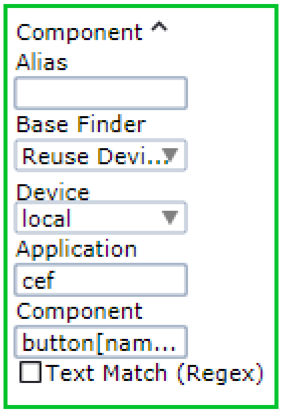
The finder is an important concept in Desktop Automation. A finder describes how to find an element you want to perform an action on. For example:

- If you want to open an application, the finder determines which device to do it on.
- If you want to click a button, the finder determines where to click.
- If you want to extract text, the finder determines where to look for the text.

Finders are created automatically when you insert an action step by right-clicking in the **Recorder View**. Design Studio attempts to construct a finder that reliably finds the element you intend to perform an action on.

If you insert a step directly, by clicking in the editor view, the finder is empty and you have to configure the finder to specify the target of the action.

You can always examine the finder used in a step by expanding the box labeled **Device**, **Application**, or **Component**.



Component ^

Alias

Base Finder
Reuse Devi..

Device
local

Application
cef

Component
button[nam...

□ Text Match (Regex)

Types of Finders

There are four main types of finders.

1. Device finder
2. Application finder
3. Component finder
4. Image finder

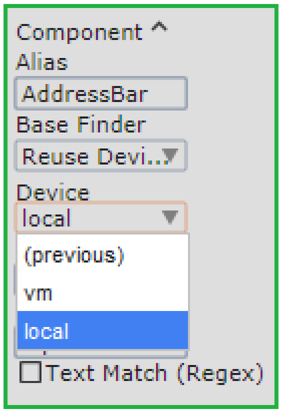
Each is increasingly more complex than the previous one and finds a more specific element.

Device Finder

The simplest finder is the device finder. A device finder contains only a device selector that chooses between accessible devices.

The device selector is a drop-down list containing the names of devices that the robot can access. The devices are listed under **Required Devices** on the **Action** tab of the Call Desktop Automation Robot step.

The drop-down list also contains aliases of other finders. Choosing one of the aliases reuses the device selector from that finder.



Component ^

Alias

AddressBar

Base Finder

Reuse Devi...

Device

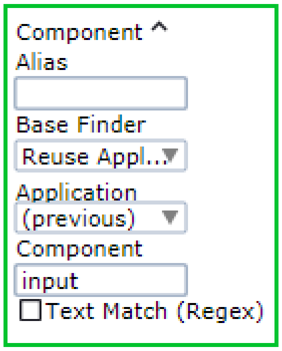
local

The Device finder can be used, for example, in the Open to select the device where you want to open an application.

Application Finder

The application finder at first selects a specific device and then a particular application on that device. The application finder requires a device selector and an application selector.

If the application finder is based on another application finder, it reuses the device selector and application selector of that finder.



Component ^

Alias

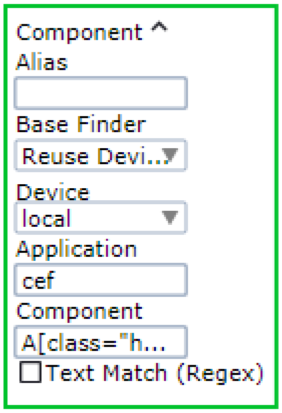
Base Finder: Reuse Appl...

Application: (previous)

Component: input

□ Text Match (Regex)

If the application finder is based on a device finder, it reuses the device selector and you must provide the application selector.



Component ^

Alias

Base Finder: Reuse Devi...

Device: local

Application: cef

Component: A[class="h...

□ Text Match (Regex)

The application selector uses CSS selector syntax as described in Selector Syntax.

When reusing an already found application, an internal handle to that application is used instead of performing the search among all applications again. This ensures that an application can be targeted correctly even when multiple new instances of the application are started with identical names.

The Application finder can be used in the Press Key to select an application that receives the key press.

Note On some configurations the application selector is ignored. For example, a key press is sent indiscriminately to the device and whatever application has focus (is in front) at the time receives the key press. In such cases you must ensure that the required application has focus when the action is performed.

Component Finder

The most common finder is the Component finder. Apart from selecting a specific device and application, the component selector finds a specific component within an application, such as an input field, a button, an icon, a table, or a menu.

The component finder is based on either of the following.

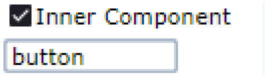
- A device
- A found application
- A found component

If the component finder is based on a device, you must provide an application selector and a component selector. If reusing an application, you need to provide only a component selector.

The component selector uses the same CSS selector syntax as described in Selector Syntax.

When reusing an already found component, an internal handle to that component is used instead of performing the search among all components again. This ensures that a component can be targeted correctly even if it changes position, and speeds up execution.

For finders reusing a component, there is an extra optional Inner Component field. This field can be used to find components within an already found component such as a cell within a table, or a button inside a modal dialog box.

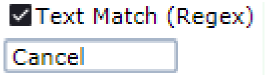


☑ Inner Component

button

The inner component selector also uses CSS selector syntax as described in Selector Syntax.

For all types of component finders, there is an optional Text Match (Regex) selector. It searches for matching text content among found components using the preceding selectors. The Text Match (Regex) selector is useful to distinguish between elements that vary only in the content they contain. For example, if a component selector finds both an OK and a Cancel button, the text selector can distinguish the text "Cancel" and target specifically the Cancel button. Text selectors are written using regular expression syntax.



☑ Text Match (Regex)

Cancel

Image Finder

Some component finders have an additional image selector. These component finders are also called *image finders*.

You can use an image finder when the element you want to interact with is not readily found in the application tree, but it has a distinct graphical appearance.

> **Note** The image finder is limited to 3000 matches per finder; a robot stops searching for a matching image after it finds 3000 instances.

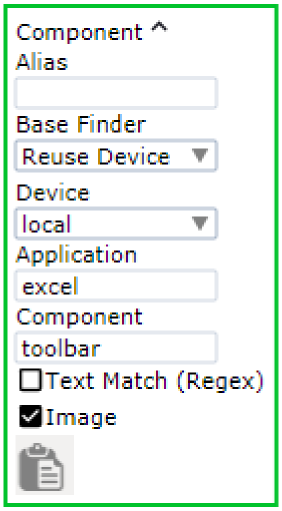
Image finders are not used by default, but they can replace the Component finder in any step that uses one. To use an image finder:

1. Drag a rectangular selection (marked in purple) in the Recorder view around the graphical element you want to find. You can modify the rectangle by dragging its sides, or just draw a new rectangle.
2. Insert a step by right-clicking inside the selection. The step should contain an image finder made from the selection.

The image selector in an image finder cannot be edited, but it can be replaced as described above in step 1.

You can remove the image selector, turning the image finder into a normal component finder, by removing the selection in the **Image** field.

The image selector has two forms: A simple image selector and a nine-grid selector (described in Nine-Grid Image Finder).



Component ^

Alias

Base Finder

Reuse Device ▼

Device

local ▼

Application

excel

Component

toolbar

☐ Text Match (Regex)

☑ Image

Important If you connect remotely to a Windows automated device via Remote Desktop Protocol (RDP), resolution and color depth depend on the network connection parameters between your computer and remote device. For example, different connection speed in different robot runs can lead to issues if robots use Image Finder and Intelligent Screen Automation (ISA). For Image Finder and ISA it is very important to always receive the same picture (pixel-by-pixel) in the same application state. Always use the same explicitly-specified resolution and color depth parameters (`g`: desktop geometry (WxH) and `a`: connection color depth) for the RDP connection in the Open step. For example:

```
rdp://user1:MyPassword@MyDesktop?g=640x480&a=16
```

Note that another connection can alter the screen graphical parameters on the Desktop Automation service computer and therefore affect the Image Finder object.

Selector Syntax

The application and component selector use the syntax of CSS selectors, which provides a powerful way to specify which elements should be selected.

The general pattern of a selector is as follows:

```
elementName[attributeName="attributeValue"]
```

For example, to find the `explorer.exe` application window with the title "Documents," you would use the following pattern:

```
explorer.exe[title="Documents"]
```

Selector patterns can also be nested with the greater than sign (>) denoting a parent-child relation and a blank space denoting ancestor-descendant relation. For example, to find a button that is a child of a toolbar element that is somewhere among the descendants of a window element, you could use the following pattern:

```
window toolbar > button
```

Advanced Selector Syntax

Kofax RPA supports most of the advanced selector syntaxes. The following table lists supported operators and how they work.

Pattern	Meaning
*	Any element
E	An element of type E
E[foo]	An E element with a "foo" attribute
E[foo="bar"]	An E element for which the "foo" attribute value is exactly equal to "bar"
E[foo~="bar"]	An E element for which the "foo" attribute value is a list of whitespace-separated values, one of which is exactly equal to "bar"
E[foo^="bar"]	An E element for which the "foo" attribute value begins exactly with the string "bar"

Pattern	Meaning
E[foo$="bar"]	An E element for which the "foo" attribute value ends exactly with the string "bar"
E[foo*="bar"]	An E element for which the "foo" attribute value contains the substring "bar"
E:root	An E element, at the root of the document
E:nth-child(n)	An E element, the nth child of its parent
E:nth-last-child(n)	An E element, the nth child of its parent, counting from the last one
E:nth-of-type(n)	An E element, the nth sibling of its type
E:nth-last-of-type(n)	An E element, the nth sibling of its type, counting from the last one
E:first-child	An E element, the first child of its parent
E:last-child	An E element, the last child of its parent
E:first-of-type	An E element, the first sibling of its type
E:last-of-type	An E element, the last sibling of its type
E F	An F element descendant of an E element
E > F	An F element child of an E element
E + F	An F element immediately preceded by an E element
E ~ F	An F element preceded by an E element

Multiple Attributes

You can use multiple attributes in a selector with the pattern to uniquely identify an application window:

```
element[attribute1=”value1”][attribute2=”value2”][attribute3=”value3”]
```

For example, to find a button that has both visible=”true” and a name that begins with ”Save,” you can use:

```
button[visible=”true”][name^=”Save”]
```

Reusable Finders

Creating a reliable finder is important for the stability of the automation process. In some cases it can be challenging and involve manual modification of the selectors in the finder. Once you are satisfied with how a finder works, you can reuse it in many places.

Another reason for reuse is to ensure consistency. If many steps perform actions on the same element, it makes sense to have all the steps use the exact same finder. This ensures that there is no disagreement as to what the element is.

A finder can be reused three ways:

- Copy and paste a finder
- Reference the previous finder
- Reference the finder by name

Copy and Paste a Finder

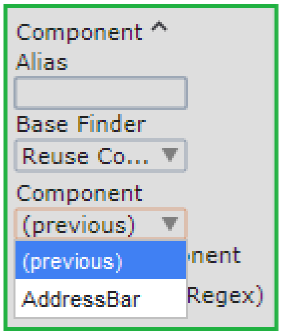
Copying and pasting a finder is the most fragile method of reuse. This method can be practical in situations when you want to create a finder but do not want to start with an empty finder.

To copy a finder, select the finder in the editor pane and either press **Ctrl+C**, or click the copy button on the toolbar.

To paste a copied finder, select the finder you want to overwrite and either press **Ctrl+V** or click the paste button on the toolbar.

Reference the Previous Finder

The reference to the previous is the most useful and common way to reuse a finder. Such a reference is marked **(previous)** in the reuse drop-down list in the second field from the top of a finder. In this case, the current finder reuses the selectors of the most recently used finder.



Component ^
Alias
Base Finder
Reuse Co...
Component
(previous)

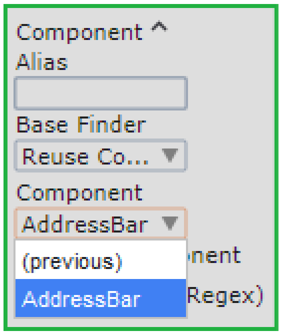
When a chain of finders use the previous finder, it means they all share the configuration of the first non-previous finder. Editing all the finders in the chain is performed by editing the first finder.

For most steps created from the Recorder view, the finders automatically contain references to the previous finder.

Reference a Named Finder

References to named finders are useful when the finder you want to reuse is not the previous finder.

To name a finder so it can be reused, type a descriptive name into the **Alias** field (the first field from the top of a finder). Once you have one or more named finders, they appear as options on the reuse drop-down list for subsequent finders.



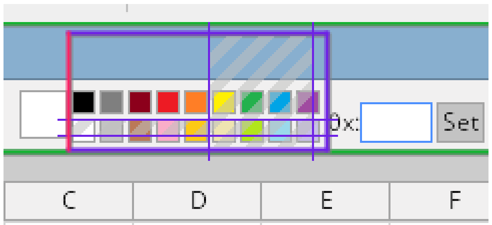
Component ^
Alias
Base Finder
Reuse Co...
Component
AddressBar

When a finder reuses another finder, it shares the configuration of that finder. Editing the named finder affects all the finders that reference it.

Nine-Grid Image Finder

Note that nine-grid image finder is being deprecated. You can use your robots with this finder, create new nine-grid image finders, but you cannot edit the finder in the editor pane. Kofax does not recommend using this finder, because it will be removed in one of the future releases.

Nine-grid image finder is a type of Image finder that can be used to find elements of different size in the Recorder View. You can use this finder primarily to find buttons or text fields. An element in this context is a sub image that is searched for when evaluating the image finder. When creating the finder, you can define up to nine elements to find. If a nine-grid finder is used for extraction, the central element is extracted. If the finder is used with the Move Mouse, the pointer is moved to the central element with regard to the offsets defined in the step.



In the image above, the striped cells are excluded from the finder evaluation.

Prerequisites

- You must define enough elements, so that the center element dimensions can be calculated, such as two diagonal corners or top and left side elements.
- The included elements are searched for in a pixel-by-pixel manner with no relaxation on the match.
- Each element must be at least 1x1 pixel in size.

Add Nine-Grid Finder

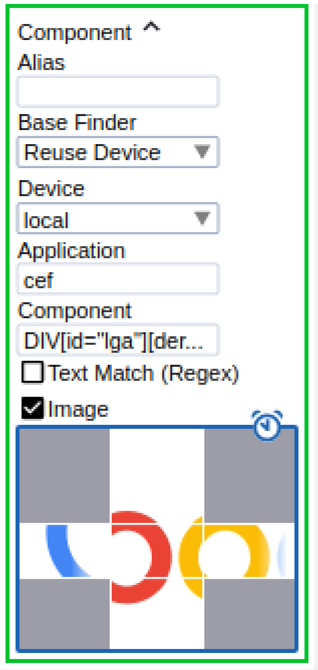
To create a nine-grid image finder, drag a box in the Recorder View similarly to the image finder and click the image finder toggle button. Moving the bars inside the finder helps you change the elements in the nine-grid finder. You can move a bar inside the finder using a mouse or by clicking it and using arrow keys. You can modify the rectangle by dragging its sides or by drawing a new rectangle.

To include an element in the finder evaluation, right-click an element and select **Include Cell When Finding**. To exclude an element, right-click a cell and select **Exclude Cell When Finding**. You can also use Ctrl+Click to include and exclude elements. Included elements are transparent while excluded elements are striped with grey diagonal lines. Note that by default all corner elements are included in the nine-grid finder.

To return to the simple image finder, click the image finder toggle button again.

Work with Nine-Grid Finder in the Workflow View

Once you add a step with a nine-grid finder, you should be able to see it in the editor view. The nine-grid finder only shows the included elements in the workflow view. The excluded elements are grey. Clicking an element in the finder toggles the inclusion state.



Component ^

Alias

Base Finder: Reuse Device

Device: local

Application: cef

Component: DIV[id="lga"][der...

☐ Text Match (Regex)

☑ Image

The image is scaled and clipped to better see the details of small elements and avoid showing huge images.

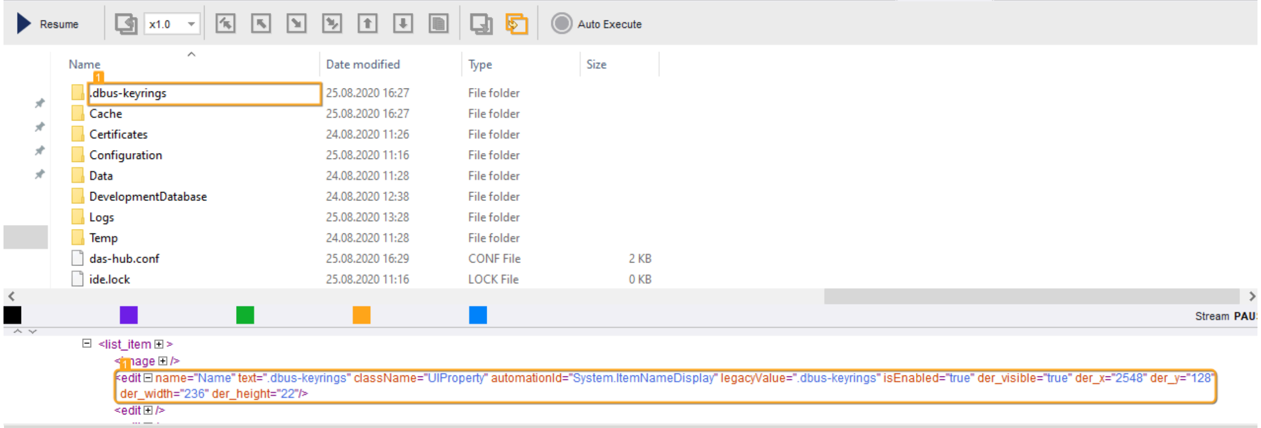
Example: Finder Samples

The following are samples of finders that locate one or more elements in the file list of Windows Explorer.

Note If a finder locates several nodes, you can cycle through found elements by clicking the "Show next location found" button.

Finding the first "edit" element that is a child of "list"

- Selector: ":nth-of-type"
- Finder: "list edit:nth-of-type(1)"

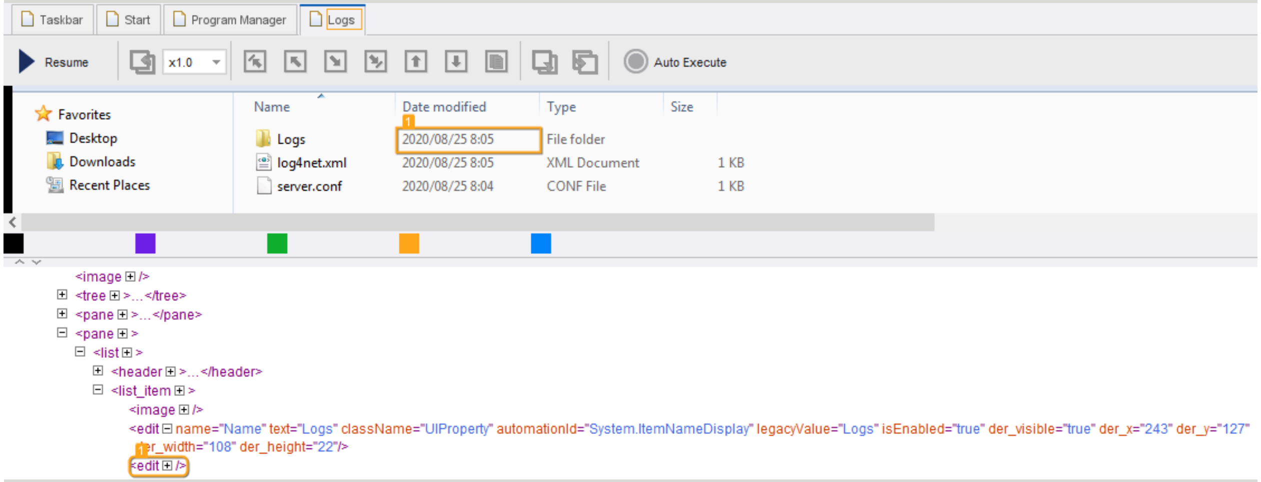


Name	Date modified	Type	Size
.dbus-keyrings	25.08.2020 16:27	File folder	
Cache	25.08.2020 16:27	File folder	
Certificates	24.08.2020 11:26	File folder	
Configuration	25.08.2020 11:16	File folder	
Data	24.08.2020 11:28	File folder	
DevelopmentDatabase	24.08.2020 12:38	File folder	
Logs	25.08.2020 13:28	File folder	
Temp	24.08.2020 11:28	File folder	
das-hub.conf	25.08.2020 16:29	CONF File	2 KB
ide.lock	25.08.2020 11:16	LOCK File	0 KB

```
⊟ <list_item ⊞ >
    <image ⊞ />
    <edit ⊟ name="Name" text=".dbus-keyrings" className="UIProperty" automationId="System.ItemNameDisplay" legacyValue=".dbus-keyrings" isEnabled="true" der_visible="true" der_x="2548" der_y="128"
     der_width="236" der_height="22"/>
    <edit ⊞ />
```

Finding the second "edit" element that is a child of "list"

- Selector: ":nth-of-type"
- Finder: "list edit:nth-of-type(2)"

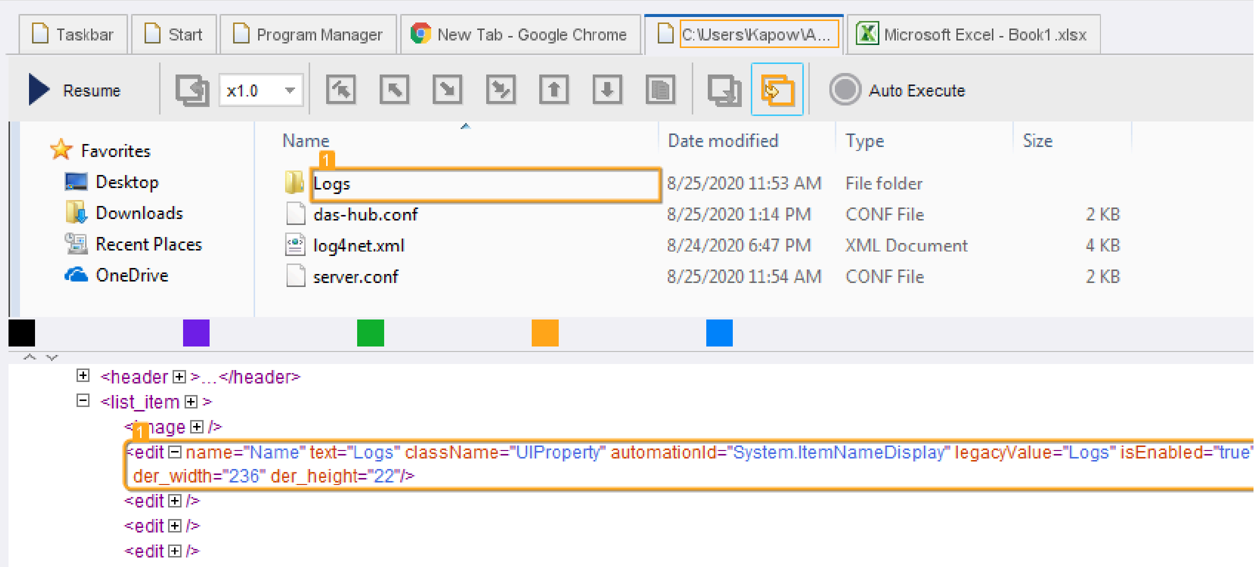


Name	Date modified	Type	Size
Logs	2020/08/25 8:05	File folder	
log4net.xml	2020/08/25 8:05	XML Document	1 KB
server.conf	2020/08/25 8:04	CONF File	1 KB

```
  <image ⊞ />
⊞ <tree ⊞ >...</tree>
⊞ <pane ⊞ >...</pane>
⊟ <pane ⊞ >
  ⊟ <list ⊞ >
    ⊞ <header ⊞ >...</header>
    ⊟ <list_item ⊞ >
        <image ⊞ />
        <edit ⊟ name="Name" text="Logs" className="UIProperty" automationId="System.ItemNameDisplay" legacyValue="Logs" isEnabled="true" der_visible="true" der_x="243" der_y="127"
         der_width="108" der_height="22"/>
        <edit ⊞ />
```

Finding an "edit" element that is the second child

- Selector: ":nth-child"
- Finder: "list edit:nth-child(2)"

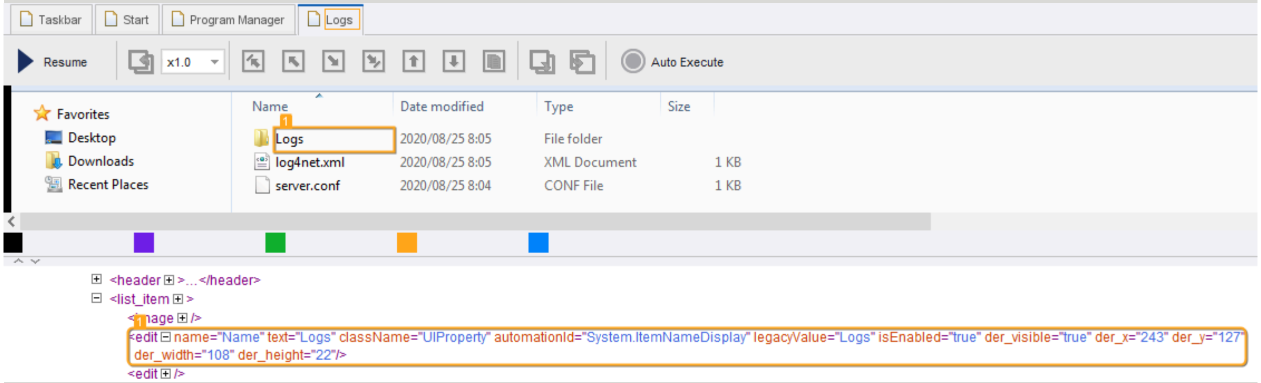


Name	Date modified	Type	Size
Logs	8/25/2020 11:53 AM	File folder	
das-hub.conf	8/25/2020 1:14 PM	CONF File	2 KB
log4net.xml	8/24/2020 6:47 PM	XML Document	4 KB
server.conf	8/25/2020 11:54 AM	CONF File	2 KB

```
⊞ <header ⊞ >...</header>
⊟ <list_item ⊞ >
    <image ⊞ />
    <edit ⊟ name="Name" text="Logs" className="UIProperty" automationId="System.ItemNameDisplay" legacyValue="Logs" isEnabled="true"
    der_width="236" der_height="22"/>
    <edit ⊞ />
    <edit ⊞ />
    <edit ⊞ />
```

Finding an "edit" element that is a descendant of "list"

- Selector: <space>
- Finder: "list edit"

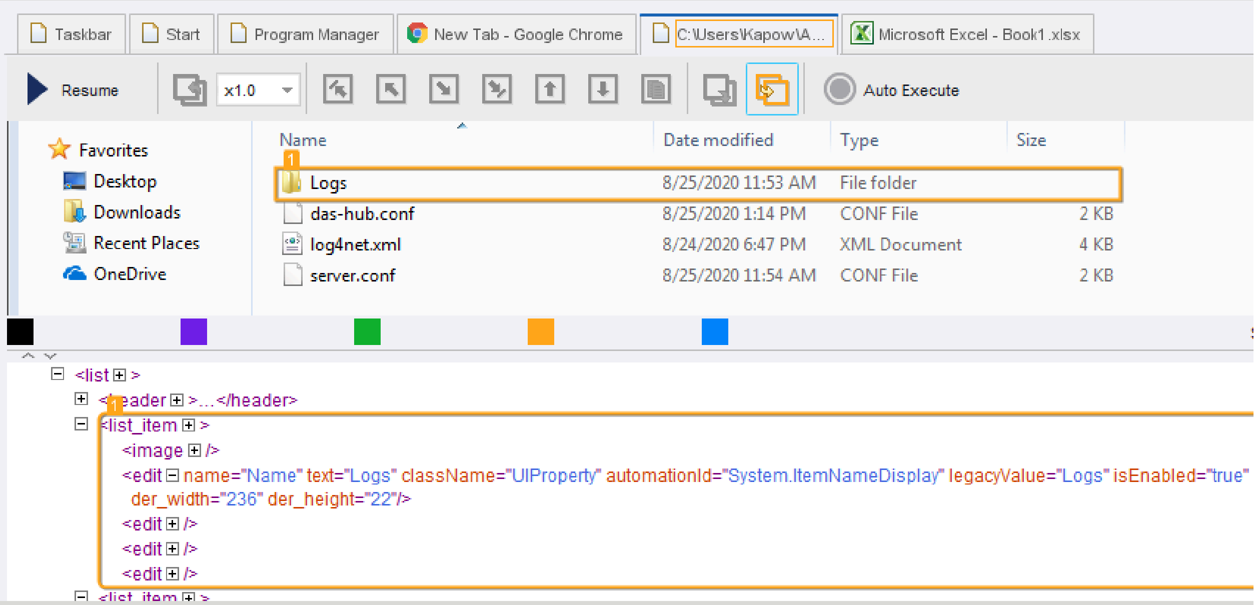


Name	Date modified	Type	Size
Logs	2020/08/25 8:05	File folder	
log4net.xml	2020/08/25 8:05	XML Document	1 KB
server.conf	2020/08/25 8:04	CONF File	1 KB

```
⊞ <header ⊞ >...</header>
⊟ <list_item ⊞ >
    <image ⊞ />
    <edit ⊟ name="Name" text="Logs" className="UIProperty" automationId="System.ItemNameDisplay" legacyValue="Logs" isEnabled="true" der_visible="true" der_x="243" der_y="127"
    der_width="108" der_height="22"/>
    <edit ⊞ />
```

Finding a "list_item" element that is a child of "list"

- Selector: >
- Finder: "list > list_item"

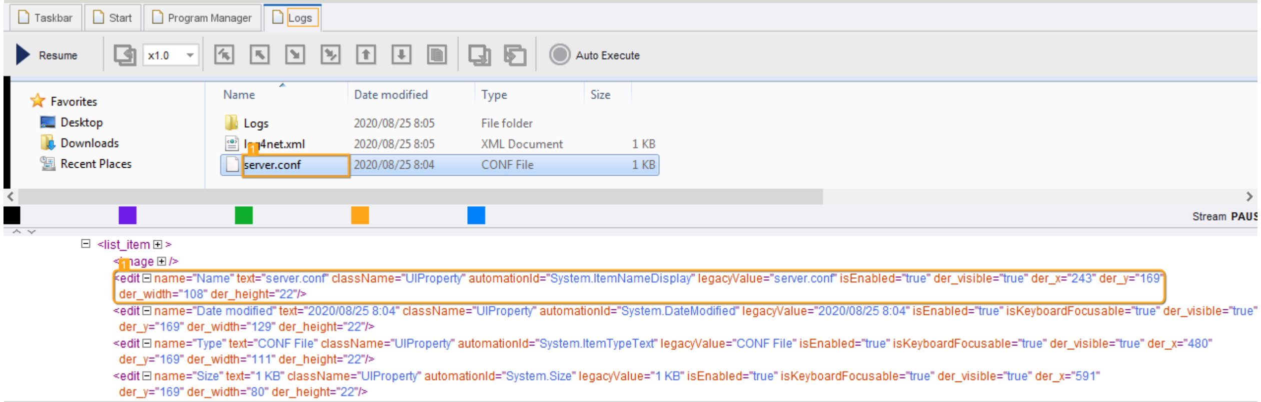


Name	Date modified	Type	Size
Logs	8/25/2020 11:53 AM	File folder	
das-hub.conf	8/25/2020 1:14 PM	CONF File	2 KB
log4net.xml	8/24/2020 6:47 PM	XML Document	4 KB
server.conf	8/25/2020 11:54 AM	CONF File	2 KB

```
⊟ <list ⊞ >
  ⊞ <header ⊞ >...</header>
  ⊟ <list_item ⊞ >
      <image ⊞ />
      <edit ⊟ name="Name" text="Logs" className="UIProperty" automationId="System.ItemNameDisplay" legacyValue="Logs" isEnabled="true"
       der_width="236" der_height="22"/>
      <edit ⊞ />
      <edit ⊞ />
      <edit ⊞ />
  ⊟ <list_item ⊞ >
```

Finding an "edit" element that has a "text" attribute with the "server.conf" value

- Selector: <attribute>
- Finder: "edit[text="server.conf"]"

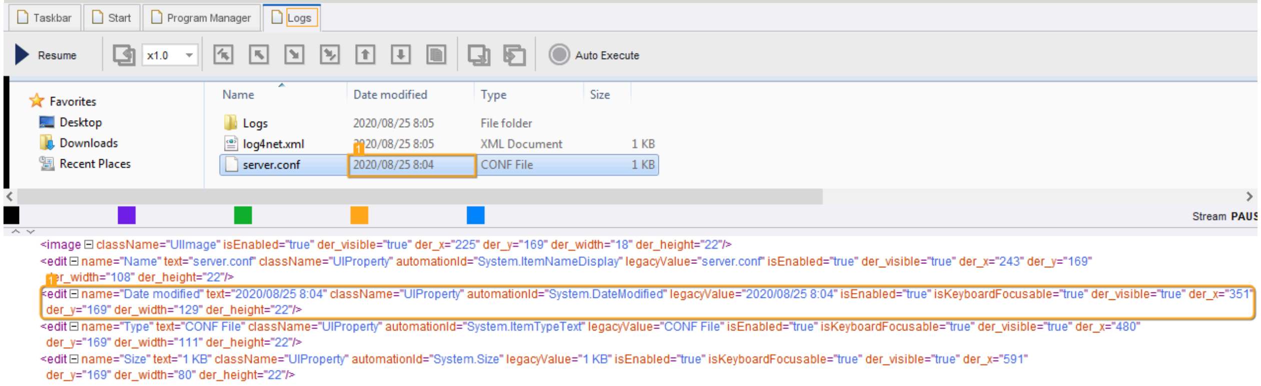


Name	Date modified	Type	Size
Logs	2020/08/25 8:05	File folder	
log4net.xml	2020/08/25 8:05	XML Document	1 KB
server.conf	2020/08/25 8:04	CONF File	1 KB

```
⊟ <list_item ⊞ >
    <image ⊞ />
    <edit ⊟ name="Name" text="server.conf" className="UIProperty" automationId="System.ItemNameDisplay" legacyValue="server.conf" isEnabled="true" der_visible="true" der_x="243" der_y="169"
     der_width="108" der_height="22"/>
    <edit ⊟ name="Date modified" text="2020/08/25 8:04" className="UIProperty" automationId="System.DateModified" legacyValue="2020/08/25 8:04" isEnabled="true" isKeyboardFocusable="true" der_visible="true"
     der_y="169" der_width="129" der_height="22"/>
    <edit ⊟ name="Type" text="CONF File" className="UIProperty" automationId="System.ItemTypeText" legacyValue="CONF File" isEnabled="true" isKeyboardFocusable="true" der_visible="true" der_x="480"
     der_y="169" der_width="111" der_height="22"/>
    <edit ⊟ name="Size" text="1 KB" className="UIProperty" automationId="System.Size" legacyValue="1 KB" isEnabled="true" isKeyboardFocusable="true" der_visible="true" der_x="591"
     der_y="169" der_width="80" der_height="22"/>
```

Finding an "edit" element located immediately after the element that has a "text" attribute with the "server.conf" value

- Selector: +
- Finder: "edit[text="server.conf"] + edit"

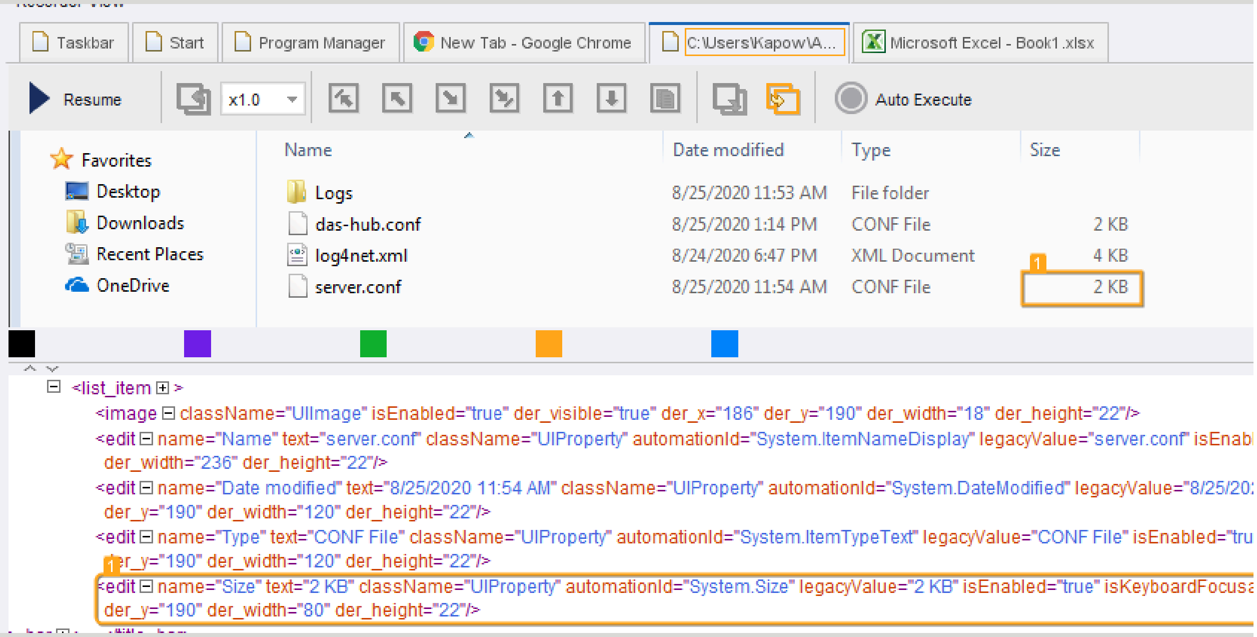


Name	Date modified	Type	Size
Logs	2020/08/25 8:05	File folder	
log4net.xml	2020/08/25 8:05	XML Document	1 KB
server.conf	2020/08/25 8:04	CONF File	1 KB

```
<image ⊟ className="UIImage" isEnabled="true" der_visible="true" der_x="225" der_y="169" der_width="18" der_height="22"/>
<edit ⊟ name="Name" text="server.conf" className="UIProperty" automationId="System.ItemNameDisplay" legacyValue="server.conf" isEnabled="true" der_visible="true" der_x="243" der_y="169"
 der_width="108" der_height="22"/>
<edit ⊟ name="Date modified" text="2020/08/25 8:04" className="UIProperty" automationId="System.DateModified" legacyValue="2020/08/25 8:04" isEnabled="true" isKeyboardFocusable="true" der_visible="true" der_x="351"
 der_y="169" der_width="129" der_height="22"/>
<edit ⊟ name="Type" text="CONF File" className="UIProperty" automationId="System.ItemTypeText" legacyValue="CONF File" isEnabled="true" isKeyboardFocusable="true" der_visible="true" der_x="480"
 der_y="169" der_width="111" der_height="22"/>
<edit ⊟ name="Size" text="1 KB" className="UIProperty" automationId="System.Size" legacyValue="1 KB" isEnabled="true" isKeyboardFocusable="true" der_visible="true" der_x="591"
 der_y="169" der_width="80" der_height="22"/>
```

Finding an "edit" element located anywhere after the element that has a "text" attribute with the "server.conf" value

- Selector: ~
- Finder: "edit[text="server.conf"] ~ edit"

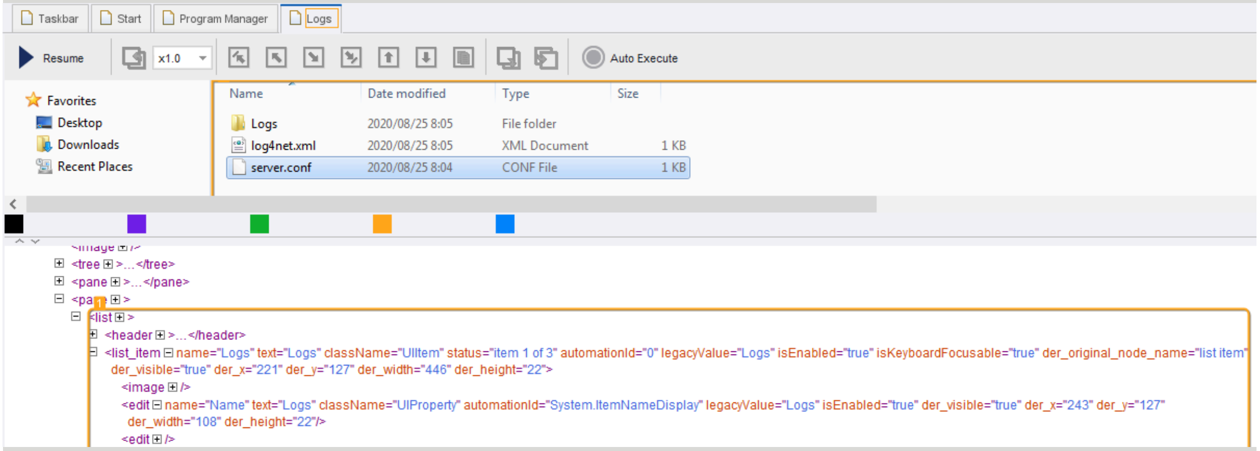


Name	Date modified	Type	Size
Logs	8/25/2020 11:53 AM	File folder	
das-hub.conf	8/25/2020 1:14 PM	CONF File	2 KB
log4net.xml	8/24/2020 6:47 PM	XML Document	4 KB
server.conf	8/25/2020 11:54 AM	CONF File	2 KB

```
⊟ <list_item ⊞ >
    <image ⊟ className="UIImage" isEnabled="true" der_visible="true" der_x="186" der_y="190" der_width="18" der_height="22"/>
    <edit ⊟ name="Name" text="server.conf" className="UIProperty" automationId="System.ItemNameDisplay" legacyValue="server.conf" isEnabl
     der_width="236" der_height="22"/>
    <edit ⊟ name="Date modified" text="8/25/2020 11:54 AM" className="UIProperty" automationId="System.DateModified" legacyValue="8/25/20
     der_y="190" der_width="120" der_height="22"/>
    <edit ⊟ name="Type" text="CONF File" className="UIProperty" automationId="System.ItemTypeText" legacyValue="CONF File" isEnabled="tru
     der_y="190" der_width="120" der_height="22"/>
    <edit ⊟ name="Size" text="2 KB" className="UIProperty" automationId="System.Size" legacyValue="2 KB" isEnabled="true" isKeyboardFocusa
     der_y="190" der_width="80" der_height="22"/>
```

Finding the first "list" element anywhere

- Selector: ":first-child"
- Finder: "list:first-child"



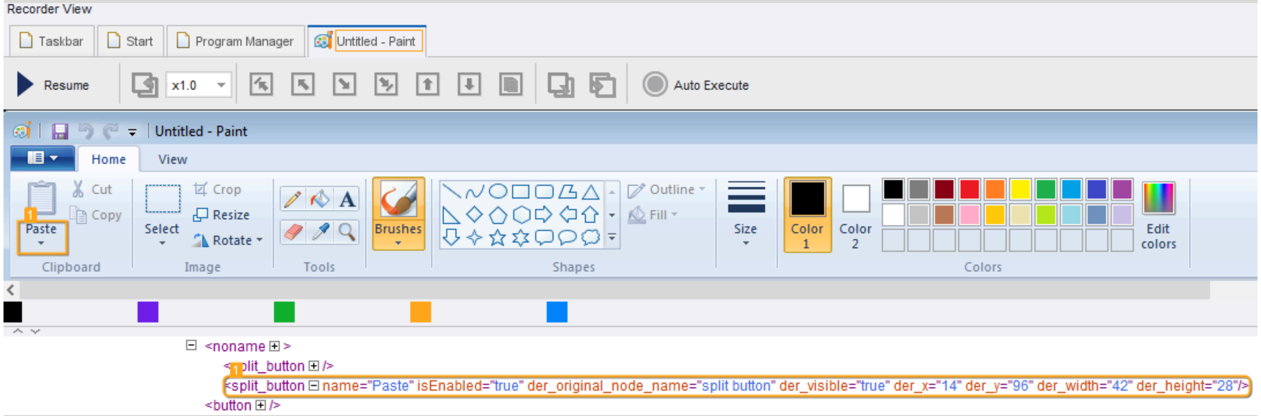
```
<image ⊞ />
⊞ <tree ⊞ >...</tree>
⊞ <pane ⊞ >...</pane>
⊟ <pa  ⊞ >
  ⊟ <list ⊞ >
    ⊞ <header ⊞ >...</header>
    ⊟ <list_item ⊟ name="Logs" text="Logs" className="UIItem" status="item 1 of 3" automationId="0" legacyValue="Logs" isEnabled="true" isKeyboardFocusable="true" der_original_node_name="list item"
       der_visible="true" der_x="221" der_y="127" der_width="446" der_height="22">
        <image ⊞ />
        <edit ⊟ name="Name" text="Logs" className="UIProperty" automationId="System.ItemNameDisplay" legacyValue="Logs" isEnabled="true" der_visible="true" der_x="243" der_y="127"
         der_width="108" der_height="22"/>
        <edit ⊞ />
```

Use multiple attributes in finder

In the following example, a finder with a conjunction of two attributes is used to click the Paste button in the Windows Paint application.

The paste icon has the same name, but it does not have the `isEnabled` flag. Design Studio generates this finder for the Paste drop-down button.

- Application: mspaint.exe
- Finder: split_button[isEnabled="true"][name="Paste"]



```
⊟ <noname ⊞ >
    <split_button ⊞ />
    <split_button ⊟ name="Paste" isEnabled="true" der_original_node_name="split button" der_visible="true" der_x="14" der_y="96" der_width="42" der_height="28"/>
  <button ⊞ />
```

Tree Modes

Tree Modes is an application-based setting that defines how the tree for applications is populated.

The Tree Modes option is available either at the top of the editor view along with Input, Output, and Variables settings, or by right-clicking an application tab title in the **Recorder View**.

Kofax RPA provides several ways to populate the application tree. By default Kofax RPA detects the type of application the robot is working with (such as a Windows application, terminal, built-in browser, and so on) and automatically forms the tree for this application. Also, Tree Modes helps you select specific options for automated applications.

- No Tree: Desktop Automation does not populate the application tree. Use this option for applications that become unstable if a robot tries to extract their tree.
- ISA: Stands for Intelligent Screen Automation. This option turns on user interface (UI) recognition that determines interface elements from a graphical representation of the program interface. You can use this mode with programs that do not provide automation API.
- Windows: Provides some options for the widget tree generated using Windows automation API.

Intelligent Screen Automation (ISA)

ISA provides extended screen recognition capabilities by automatically finding UI elements and shapes.

> **Note** Do not use Intelligent Screen Automation (ISA) in Attended Automation.

ISA helps you automate applications with limited or no automation API, such as Citrix. Also, you can use this option to automate applications that are otherwise slow, such as Remote Desktop applications, SAP, and others. In general, you can use this option for any application for which the application tree is incorrectly populated or is otherwise difficult to create a finder for.

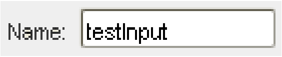
Each recognized UI element contains several attributes that you can use in your finders.

Common attributes

- `der_x`, `der_y`, `der_width`,`der_height`: Coordinates and size of the element.
- `lt_16_5`, `rb_15_4`: Calculated numbers relative to the element's left top and right bottom coordinates. Element coordinates might change in different program state. For example, selected option with bold text can have different coordinates than the cleared option, and therefore finders might work incorrectly. This property eliminates coordinate change issues and can be used to create reliable finders. If a UI element has a `name` property, it is used in a finder by default when you insert an action step. Otherwise, property `lt_16_5` is used in a finder.

The following table lists supported UI elements.

UI element name	Widget name	Element-specific attributes	Notes
Dialog box, frame, pane, and other	`container`	N/A	A general parent element that contains other elements. For example, an entire dialog box or a form inside this dialog box can be a container. One of the special containers that Kofax RPA can recognize is a table.
Element	`element`		A general UI element that does not contain other elements.
Icon	`icon`	`label`	A graphical element that does not contain any text.

UI element name	Widget name	Element-specific attributes	Notes
Table	`table`		Tables are represented in the application tree as separate elements with the following restrictions: merged cells are not recognized and headers are the same as regular cells. Each cell is defined as an `item` in the application tree. Items can contain other UI elements, such as check boxes, text boxes, and other.
Row	`row`		A row in the table tree.
Item	`item`		An item within a row in the table tree.
Text box	`textbox`	`label`	A text field.
Check box	`checkbox`	`label`	A check box.
Option button	`radiobutton`	`label`	An option button or radio button.
Text and text label	`linklabel`	`name` `inverted`	This is either a standalone text element, such as dialog box subtitle, or a UI element label, such as a check box label, text box label, and so on. • `name`: If a text is recognized as a UI element label, the UI element will contain a `label` property with the label name as its value. In the following example, `textbox` element will contain `label="Name"` property.  Name: testInput • `inverted`: If the detected by OCR text is dark on a white background, `inverted` is set to `0`. If the text is white on a dark background, `inverted` is set to `1`.

ISA tips and tricks

- Recognition results depend on many factors, such as screen fonts, background and foreground color, text size, and so on. If you want to improve recognition results, try using different color schemes in your interface, such as use regular monospaced black-colored fonts on a bright background and larger font size.

- Selected text with inverted colors in a text field might not be correctly recognized. To improve recognition results, remove the selection from the text field (such as by clicking somewhere else) before extracting its value.
- When you work with dynamic content forms and text fields that might be recognized differently in different state, use the Freeze Tree step to hold the application tree state and improve recognition results.
- If you want to extract value from an element, use the Extract Value From action as you would do when using Windows automation API. If the element's value is not recognized in ISA mode, try Extract Text From Image action.
- You can edit extended OCR settings in the `ocr.cfg` file. See Extended OCR Settings for more information.
- You can train Tesseract to recognize your character set using either TTF fonts or UI screen shots. See Train Tesseract for more information.

Important If you connect remotely to a Windows automated device via Remote Desktop Protocol (RDP), resolution and color depth depend on the network connection speed between your computer and remote device. Different connection speed in different robot runs can lead to issues if robots use Image Finder and Intelligent Screen Automation (ISA). For Image Finder and ISA it is very important to always receive the same picture (pixel-by-pixel) in the same application state. Always use the same explicitly-specified resolution and color depth parameters (`g`: desktop geometry (WxH) and `a`: connection color depth) for the RDP connection in the Open step. For example:

```
rdp://user1:MyPassword@MyDesktop?g=640x480&a=16
```

Change or add UI recognition language

By default ISA uses a language specified in the `ocr.cfg` file. If you want to add or change the UI recognition language, see Change OCR engine and language topic. If you want to change the ISA UI recognition language for a specific computer, perform the following steps.

1. In the text editor, open the `isa_v1.cfg` file located in the `lib` directory as follows.
   - On the Windows-based automated computer with installed Desktop Automation Service:

     `C:\ProgramData\Kofax RPA\<build number>\lib\tessdata`.

     > **Note** If you have several versions of the Desktop Automation Service installed on the automated computer, locate the version of the service that automates the desktop using ISA. See the "**Windows** tab" in the Configure Desktop Automation Service topic for details.

   - On the local Windows-based computer to use with built-in browser:

     `nativelib\hub\windows-x32\<build number>\lib`* in the Kofax RPA installation directory. Example:

     `C:\Program Files\Kofax RPA 10.6.0.0 x64\nativelib\hub\windows-x32\622\lib`
   - On the local Linux-based computer to use with built-in browser:

     `nativelib/hub/linux-x64/<build number>/lib` in the Kofax RPA installation directory. Example:

     `Kofax_RPA_10.6.0.0_x64/nativelib/hub/linux-x64/533/lib`

   * The build number is different in different versions of the program.
2. In the `ocr_language` parameter either replace `default` with a language of your choice or, if you want to use more than one language, add another language using the plus sign, such as `ocr_language=eng+jpn`. Save and close the file.
3. Restart the Desktop Automation Service for the changes to take effect.

To return to the default recognition language, perform the steps above specifying `default` in the `ocr_language` parameter.

Add UI recognition language for Tesseract

If you use Tesseract OCR engine, ISA can automate the English language UI by default. Use the following procedure to add more languages for UI recognition. See changing the default OCR language for information on changing the recognition language. Note that using more than one language simultaneously for screen recognition slows down robot execution and deteriorates recognition results.

> **Note** OmniPage OCR engine includes all supported languages in the Kofax RPA installation.

1. Download the `.traineddata` file for the required language from the https://github.com/tesseract-ocr/tessdata. For example, the file for the Japanese language is `jpn.traineddata`.

   > **Important** Make sure downloaded `.traindata` file is compatible with your version of the `tesseract.dll` file. For compatibility information, see the README file published at the bottom of the traineddata download page at https://github.com/tesseract-ocr/tessdata.

2. Copy downloaded trained data file to the appropriate folder:
   - On the Windows-based automated computer with installed Desktop Automation Service:

     `DesktopAutomationService\lib\tessdata` in the Desktop Automation Service installation directory. Example:

     ```
     C:\Program Files (x86)\Kofax RPA DesktopAutomation 10.6.0
     x32\DesktopAutomationService\lib\tessdata
     ```
   - On the local Windows-based computer to use with built-in browser:

     `nativelib\hub\windows-x32\<build number>\lib\tessdata`* in the Kofax RPA installation directory. Example:

     ```
     C:\Program Files\Kofax RPA 10.6.0.0 x64\nativelib\hub\windows-
     x32\622\lib\tessdata
     ```
   - On the local Linux-based computer to use with built-in browser:

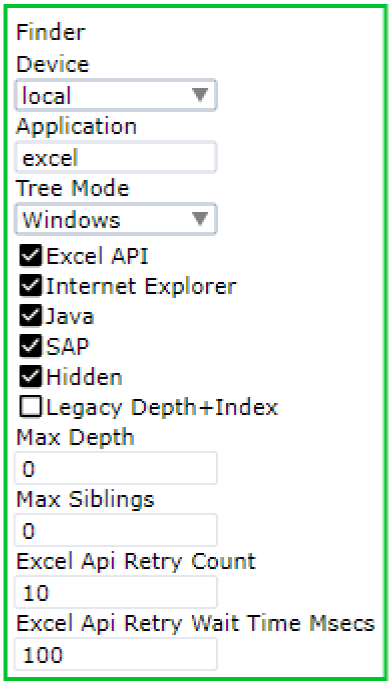
     `nativelib/hub/linux-x64/<build number>/lib/tessdata` in the Kofax RPA installation directory. Example:

     ```
     Kofax_RPA_10.6.0.0_x64/nativelib/hub/linux-x64/533/lib/tessdata
     ```

   * The build number is different in different versions of the program.

3. Change the UI recognition language as described in the "Change or add UI recognition language" section above.

Windows Options



Finder

Device

local ▼

Application

excel

Tree Mode

Windows ▼

☑ Excel API

☑ Internet Explorer

☑ Java

☑ SAP

☑ Hidden

☐ Legacy Depth+Index

Max Depth

0

Max Siblings

0

Excel Api Retry Count

10

Excel Api Retry Wait Time Msecs

100

To generate a widget tree for Windows applications, Kofax RPA uses the UI Automation framework built into Windows. Also, Kofax RPA provides some extended support for particular applications. If you experience any issues working with applications using Kofax RPA extensions, turn them off by clearing the selection in the Windows tree mode.

Excel

Turns on extended support for Microsoft Excel, such as to retrieve cells from the spreadsheet. Excel support in Kofax RPA is also extended with Built-in Excel Driver features.

Note When using Enter Text in the Excel on a remote device, do not leave the Excel spreadsheet in edit mode (the word Edit appears in the lower-left corner of the Excel program window) if you want to work with it in the following steps. To exit the edit mode, perform one of the following.

- Press ENTER using the Press Key step. Excel exits Edit mode and selects the cell directly below the current cell.
- Press TAB using the Press Key step. This stops Edit mode and selects the cell to the right of the current cell.
- Click a different cell.
- Press F2 using the Press Key step.

See Microsoft documentation for more information.

Internet Explorer

Turns on extended Internet Explorer support to retrieve DOM (Document Object Model) tree as it provides a more accurate result in the application tree. Note that Kofax RPA also provides a built-in chromium-based web browser to access websites. Note the following when considering using extended Internet Explorer support.

- If extended Internet Explorer support is enabled and the robot detects an Internet Explorer window:
  - A piece of Java-script is injected that traverses the Document Object Model (DOM) and creates a JSON representation. This runs in the context of the Internet Explorer page.
  - The JSON is placed into an input tag with id "kapowDataElement" in the page itself.
  - The robot parses the JSON content and creates the sub-tree from that content.
- If extended Internet Explorer support is disabled, the UI Automation is used to get the object tree from Internet Explorer (this is the same tree that you get using Microsoft's `Inspect.exe`).

Java
In some situations the Java Access Bridge does not work and it can help to switch to legacy mode by clearing the selection of this option. The Java option can also have an influence on Java Applets in Internet Explorer.

SAP
SAP is handled differently from other Windows applications. Working with SAP depends on scripting API that might by slow. You can turn off SAP support by clearing this option.

Hidden
Specifies whether to extract the entire tree of an application. This option is selected by default. If you clear the selection, Kofax RPA skips elements that are reported as off-screen, such as list boxes or tables with many elements. Clearing the selection reduces the time needed to extract the tree.

Legacy Depth+Index

Adds the "depth" and "index" attributes to the application tree that were used in Kofax RPA 10.2. If you have robots created in Kofax RPA version 10.2 that use component finders with "depth" and "index" attributes, select Legacy Depth+Index for the robots to work in Kofax RPA 10.3 and later.

- Max Depth: Sets the maximum number of nested elements each node shows for all application windows in the view.
- Max Siblings: Sets the maximum number of sibling elements each node shows for all application windows in the view.

  Specify `0` for no restrictions.

Excel API Retry Count
Specifies the number of retries when accessing the Excel window via Windows API. You can increase the number if Excel response is slow.

Excel API Retry Wait Time
Specifies the time in milliseconds between retries when accessing the Excel window via Windows API.

Desktop Automation Step Actions

This topic provides information on Desktop Automation step actions.

- Assign
- Browse
- Bundle
- Click

- Conditional
- Connect To Device
- Copy Component Selector
- Copy
- Convert Value
- Custom Action
- Disconnect From Device
- Document Transformation
- Enter Text
- Excel
- Extract Clipboard
- Extract Image
- Extract Text From Image
- Extract Tree as XML
- Extract Value
- Focus
- Freeze Tree
- Group
- Guarded Choice
- Initiate Session
- KTA
- Loop Steps
  - Break
  - Continue
  - For Each Loop
  - Loop
  - While Loop
- Move Mouse
- Notify
- Open
- PDF
- Press Key
- Read File
- Remote Device Action
- Return
- Scroll
- Set Clipboard
- Terminal
- Throw
- Trigger Choice
- Try-Catch

- Unrecorded Instant Click
- Windows
- Write File
- Write Log

Assign

This step assigns a value to a variable. Note that the value in the Expression field must match the variable type.

Properties

- **Name**: Name of the step.
- **Variable**: Variable name.
- **Expression**: Variable value. You can use expressions and other variables in this field. See Expressions for more information.

Browse

Use this step to open websites in one of the built-in browsers. See Access Websites for details.

Also use the Browse step to work with cookies in the Chromium built-in browser. See Cookie Management in the Desktop Automation built-in browser for details.

Properties

Browser
Select the browser engine you want to use. Kofax RPA provides a Chromium based browser.

Action
Select an action the browser must perform: **Load Page** or **Wait Downloads**.

Load Page

- **URL**

  Specify the path to a website to open. Use forward slashes in the path. For example:

  `https://www.google.com`

- **Screen Size**

  Select this option to specify the browser window width and height in pixels. The default window size is 1920x1200, which does not include the toolbar.

- **User Agent**

  Select this option to specify what the built-in browser sends in the "User-Agent" header for the HTTP requests. The "User-Agent" string contains information about your web browser name, operating system, device type, and other information.

- **Accept Language List**

  If this option is not selected, the page loads in the default language of the website. If selected, the page tries to load in the language according to the specified language code. If the website does not offer that language, the website's default language is used.

  Default code: en-US.

- **Ignore Loading State**

  Select this option for the browser to ignore background load requests, such as loading of advertising banners and the like.

  By default, CEF browser analyzes its loading state and if the browser is actively loading a webpage or redirecting to another location, it does not allow running location-based guards. The browser waits until the webpage is loaded and the browser switches to a non-loading state.

  Settings in the "Ignore loading state" property allow users to specify URLs or substrings which are used by the browser to determine if it is allowed to execute Location Found, Location Not Found, Location Disappeared, and Tree Stops Changing guards before the browser completely loads a webpage.

  Users can specify comma-separated URL parts. If while loading a webpage the browser finds a match between a webpage URL and a string specified by a user, it lets location-based guards run if any of the guards should be executed after the Open action or action which changes its loading state (such as, click a button or move mouse).

- **Authentication**

  Specify user credentials for the web page authentication. This is the authentication that requires to enter credentials in a pop-up dialog box before the page loads. If a web page does not require authentication, these settings are ignored.
- **PDF Settings**

  Specify settings for saving a web page in PDF format.
  - **Background Graphics**: Select to save background graphics of the web page.
  - **Headers and Footers**: Select to save headers and footers in the PDF document.
  - **Scale**: Specify the scale level in percentage.
  - **Paper Size**: Select the paper size for the PDF document.
  - **Layout**: Select either landscape or portrait layout of the document.
  - **Margins**: Specify page margins in the document. If you select **Custom**, enter each side margins in pixels.

    Note If you want to save your web page as PDF, use the **PDF** button on the right side of the browser window. You may need to scroll all the way to the right if the button is not visible on the screen.
- **Client Certificate**: If the website is using client certificates to authenticate the client (user), you can select this option to pass the client certificate to the website server.

  In **Certificate Storage**, specify a binary variable containing the certificate storage file. In **Storage Password**, specify a password variable containing the password to the certificate storage.

Wait Downloads

With this action, you can enable a waiting period for the files that are being downloaded. This can be useful when you need to know if a file is finished downloading or how many downloads are still active, just like you can see it in any web browser.

The maximum wait time is 300 seconds (five minutes). When the wait time is reached, and the download has not completed, the step returns the number of downloads that are still active. In this case, you can decide whether you want to continue the download (stay in the download cycle) or terminate it.

Also, you can explicitly cancel the download if it is taking too much time or is no longer needed.

- **Settings**
  - **Active Downloads**: Select this option to enable a waiting period.
  - **Cancel Downloads**: Select this option to cancel all active downloads. After the Browse/Wait Downloads step set to "Active Downloads," add a new Browse/Wait Downloads step set to cancel all active downloads.
- **Seconds**: Specify the wait time. The default is 60 seconds; the minimum and maximum wait time is 0 and 300 seconds, respectively. After the time is reached, the step returns the status of the download in the variable that you specify in **Number of Active Downloads** below. If the download process is still active, the variable shows the number of downloads in progress. If all downloads have completed, the number is 0.
- **Wait upload to RFS**: Select this option to wait for a notification from the Robot File System that the file is successfully saved to it. After the notification is received by the system, the download is stopped,

and the robot continues executing next steps. The maximum size of the file that can be uploaded to the Robot File System is 100 MB.

- **Results**:

  Number of Active Downloads: Specify an integer variable to contain information about the number of active downloads.

Bundle

This step is intended for use in the Local Desktop Automation mode. Note that Design Studio and Desktop Automation service must be set to Single User mode.

Use a Bundle step if you need to automate application elements that disappear when you remove the pointer, such as context and drop-down menus. A Bundle step connects a number of steps that you perform on the automated application and turns them into a sequence that will be executed in order, starting with the first step.

To wrap existing steps in a Bundle step, in the editor, select the steps containing the use of disappearing elements, right-click the group, and then click **Surround with Bundle step**. Also, you can insert a Bundle step directly inside the workflow and then add the necessary steps to it.

To add action steps to the Bundle step, right-click the flow point inside it and make a selection.

> **Note** Some steps are not available inside the Bundle step, but you can add them to your robot before or after the Bundle step.

Click

The Click step is one of the most commonly used actions (unless you are automating a terminal) in Desktop Automation. With a Click (and Move Mouse) step, your robot can start and close programs, work with programs interfaces, perform drag-and-drop operations, select text, and perform many other actions that a user can perform with a pointing device.

Additionally, it is possible to simulate hardware keyboard and mouse on an automated computer. For more information, see "Install the Desktop Automation Service" in the Kofax RPA Installation Guide.

Properties

Application
Application finder for the Click step.

Action
Contains three mouse actions. Each action is initiated at the current cursor position. To ensure that this is the expected location or to change the location, use the Move Mouse step.

- **Click**: Clicks in the application interface.
- **Press**: Presses the mouse button without releasing it.
- **Release**: Releases the mouse button.

Button
Select **Standard Buttons** or **Calculated Button** of the pointing device.

- **Standard Buttons**: Left, Middle, Right.
- **Calculated Button**: Select this option to determine a mouse button click when the robot is running. In the expression field, specify a virtual-key code or a space-separated list of input specifications. The result of the expression should be a value between 0 and 2 representing one of the mouse buttons. This functionality is only supported on the Windows operating system. For the list of virtual-key codes, see the Microsoft documentation.

Example

Use "0" for the left mouse button, "1" for the right mouse button, and "2" for the middle mouse button.

Modifier
Select a key modifier:

- **Fixed Key Modifier**: Contains three standard key modifiers, such as Shift, Ctrl, Alt.
- **Calculated Key Modifier**: With this option selected, specify a symbolic constant name of the virtual-key code for a modifier.

  In the text box that appears, you can enter the key codes for Shift, Ctrl, and Alt only. For example, the VK_LSHIFT key code stands for the left Shift key, VK_RCONTROL stands for the right Ctrl key, and VK_MENU stands for the Alt key. For a complete list of key codes, see the Microsoft documentation.

Count
Specify how many times to perform the action. For example, for the double-click, specify `2`.

Conditional

In this step you can specify a Boolean condition that affects the execution of the steps in your robot. This step is often used within the Loop.

Properties

Name
Name of the step.

Condition

A boolean condition.

Connect To Device

This step helps you connect to a remote device.

Properties

Name
Name of the step.

Device
Select the dynamic reference name you want to use. This reference name is specified in the **Required Devices** property of the Call Desktop Automation Robot step. The list shows only dynamic references.

Host
Enter Automation Device name or IP address.

Port
Specify the command port to use with your Automation Device (default 49998). This port is specified in the Desktop Automation Service configuration.

Token
Specify a token name. The token must match the name specified for the selected Automation Device in the **Token** field on the **Single User** tab of the Desktop Automation Service configuration.

Important The Desktop Automation Service must be in the Single User mode. See Configure the Desktop Automation Agent.

Timeout
Specify the connection attempt timeout in seconds.

Attempts
Specify the number of connection attempts. Note that there is a few seconds delay between each connection attempt.

Copy Component Selector

You can copy a component selector from the application tree view or the tag path and then paste it into the Component field of a step in the editor view. In the tree view or tag path, select and right-click an element, click **Copy Component Selector**, and then click **Copy Compact Component Selector** or **Copy Full Path Component Selector**, depending on what type of selector you require.

The compact component selector copies the smallest unique selector for the given element, which looks similar to the following:

```
TR[class="odd"]:nth-of-type(1) > TD[class=" "][der_rendered="y"]:nth-of-type(4)
```

The full path component selector copies the selector from the root of the application tree, which looks similar to the following for the same element :

```
window > _document_ > HTML > BODY > DIV:nth-of-type(3) > DIV:nth-of-type(2) > DIV:nth-
of-type(2)
    > DIV > TABLE > TBODY > TR:nth-of-type(1) > TD:nth-of-type(4)
```

The selector is copied to the clipboard. You can paste it into the Component field by using Ctrl+V.

For more information on the selector syntax, see Selector Syntax.

Copy

You can copy an element from the application tree view and use it as needed. This can be helpful if you need to write a customized component finder using particular application elements from the tree view.

In the tree view, select and right-click an element, click **Copy**, and then click **Sub Tree** or **Node**, depending on what type of a tree element you require.

To copy the selected subtree element, you can also use Ctrl+C.

The **Sub Tree** action copies the root element with all of its subelements as an XML string, which looks similar to the following:

```
<DIV class="teaser__description" der_rendered="y" der_x="1103" der_y="294"
 der_width="270" der_height="20">
 <SPAN class="teaser__nowrap" der_rendered="y" der_x="1103" der_y="295" der_width="135"
 der_height="17">
   Text
   <SPAN class="teaser__age" der_rendered="y" der_x="1222" der_y="299" der_width="16"
 der_height="13">
     0+
   </SPAN>
 </SPAN>
</DIV>
```

The **Node** action copies only the root element as an XML string, which looks similar to the following for the same element:

```
<DIV class="teaser__description" der_rendered="y" der_x="1103" der_y="294"
 der_width="270" der_height="20"/>
```

The element is copied to the clipboard.

Convert Value

You can use this step to convert a manually specified value or, together with an extract step, to convert an extracted value.

To perform the conversion for a manually specified value, the Convert Value option must be selected in the drop-down list. To perform the conversion for an extracted value, in the drop-down list, select one of the extract steps.

Properties

Convert Value

The source value to convert.

Evaluate Expression step

Contains a list of conversion functions suggested for use on a value. The function list suggested for a particular value is limited, depending on the value type. To specify a conversion function, in the step context menu, click **Evaluate Expression step**, and then do either of the following:

- Select the required function from the list of suggested functions.
- Click **Plain** to manually type the required function. For example, if you need to extract a text that represents an integer and store it in an Integer variable, you can use the expression `$initial.integer()`.

For more information on the supported functions and examples, see Expressions.

A converter can contain one or more Evaluate Expression steps.

Store Current In step

Stores a converted value in a variable of specified type. The type of the value must match that of the variable and can be one of the following: Integer, Boolean, Number, or Text.

The converter can contain one or more Store Current In steps. For example, this can useful if you need to extract a full name of a person and store it in two variables, such as for the first name and last name respectively, inside the same converter group.

A converter group can also contain the following set of action steps:

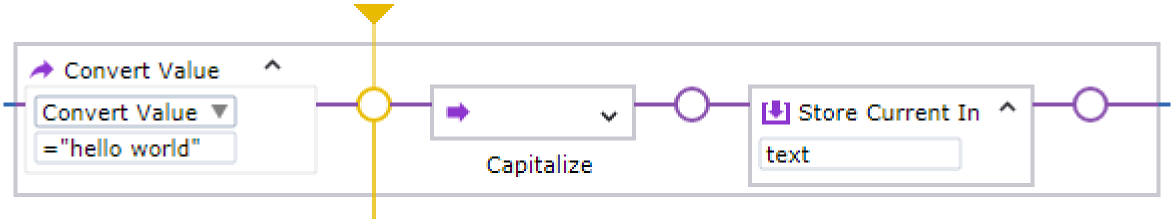
- Conditional step
- Group step
- Throw step
- Write Log step

Example: Capitalize expression "hello world"

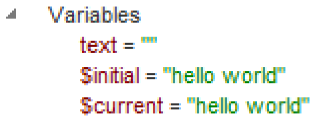
The following example demonstrates how to capitalize the expression "hello world" and store it in a variable of type text.

> **Tip** Inside a converter group, if you need to undo a change to the workflow state or see the state at a certain preceding flow point, you can simply go back inside the group of conversion steps by executing the required preceding flow point. Also, if you change the workflow before the current flow point, the workflow state is adjusted automatically, so you do not need to reexecute the conversion steps.

1. Add a Convert Value step to your automation workflow and configure it as follows.
   a. Under **Convert Value**, type **"hello world"**.
   b. After the expression, right-click the flow point and click **Evaluate Expression step** > **Capitalize: (Text) -> Text**.
2. After you execute the first part, the initial (given) value of the **"hello world"** expression is assigned to the variables **$initial** and **$current**.



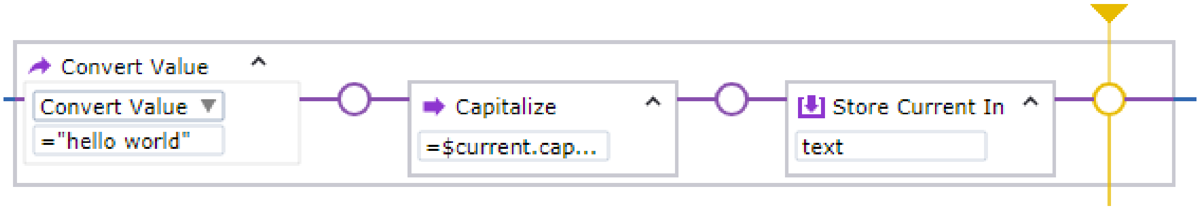
Convert Value
Convert Value ▼
="hello world"
Capitalize
Store Current In
text

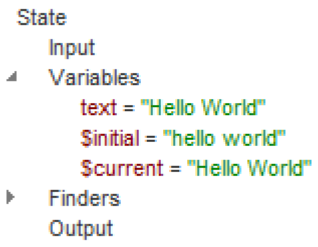


Variables
text = ""
$initial = "hello world"
$current = "hello world"

1. As you continue the execution, the expression is capitalized by the converter (note **= $current.capitalize()** in the following example), the variable **$current** is assigned a new value, **"Hello World"**, while the **$initial** variable value remains unchanged. The new value of the $current

variable, **"Hello World"**, is stored in the variable **text** of type text after the **Store Current In step** is executed.





State
Input
Variables
text = "Hello World"
$initial = "hello world"
$current = "Hello World"
Finders
Output

For more information on the $initial and $current variables, see Data Conversion.

Custom Action

The Custom Action step enables you to utilize external resources for processing data from within a robot using Connectors. Connectors are packages that define one or more actions a robot can use and contain all necessary resources to implement the actions.

Connectors can define actions that perform calls to external programs, shell commands, or execute JavaScript and Python in private instances. These actions are available as Actions in the **Custom Action** step in Design Studio.

Tip Visit the Kofax Intelligent Automation SmartHub at https://smarthub.kofax.com/ to explore additional solutions, robots, connectors, and more.

Set up Connectors

This section describes how to set up Connectors to use in the Custom Action step.

Before you can use Connectors in your robot, select **Allow the use of Connectors** on the **Security** tab in the RoboServer Settings application. RoboServer Settings can be started from the Windows Start menu. For details, see "RoboServer Configuration" chapter in the *Kofax RPA Administrator's Guide*.

Creating a Connector

Connectors are distributed as .zip archives with the `.connector` extension. Before creating a robot with a Custom Action step, archive your Connector using ZIP, rename the file extension to `.connector`, and add it to the project. When the project is uploaded to the Management Console, you can see available Connectors on the **Repository** > **Resources** tab. The Connectors are then uploaded to RoboServers and automated desktop computers if necessary.

Each .zip file must contain a `manifest.json` file in its root. This file must contain a manifest that defines all available actions for the Connector. See the manifest description below.

Each action defines a set of parameters, a response, and a command line. The parameters are presented to the robot designer and must be filled in by the robot. The command line is composed using a combination of the parameters and used to execute the request. The output of the request is parsed and used to fill the variables defined in the response.

Upload Connectors to Management Console

When you complete the design of a robot with a Custom Action step in Design Studio, upload it to the Management Console. Once the project is uploaded to the Management Console, you can see available Connectors on the **Resources** tab of **Repository**. The Connectors are then uploaded to RoboServers and automated desktop computers if necessary.

Note that Connectors using Python and NodeJS are loaded on demand and if they are not used, they do not require any resources. Each Connector receives its own python or node.js instance, and you can run multiple Connectors of these types each with a different version if necessary.

Executing a Program (type: program)

Program actions call the indicated program directly with the current working directory set to the extracted Connector file. If the executable is part of the Connector package, the manifest should make this explicit with a relative path, such as `./executable`, otherwise the robot might not locate the executable file.

If additional configuration (PATH settings or Linux dynamic loader settings) is required, it is recommended to use a shell wrapper to set up the settings and then call the executable. Parameters are passed directly to the program without further escaping.

Executing a shell command (type: shell)

Shell interfaces start a shell to execute the command line directly with the current working directory set to the extracted Connector file. Processing of this command line is handled by the shell itself and therefore depends on the platform:

- On Windows, the action calls `CMD /C` and passes all elements of the command line directly to the shell.
- On Linux, the action calls `bash -c` and passes each element of the command line as a separate argument. Parameters are passed directly to the shell without further escaping. See the documentation on the `bash -c` option for details.

Executing JavaScript (type: node)

Node.js JavaScript requests are wrapped in a `function()` context and executed. If the code contains `require` statements, they are resolved using the `node_modules` directory in the root of the Connector package.

By default, parameters are converted into strings and escaped before they are inserted into the command line. The command line is not escaped.

If the interface is synchronous, the robot receives both the value returned by the JavaScript and an exception value. Only one of these two fields has a non-empty value. An object is serialized to JSON before it is returned to the robot.

A specific Node.js executable can be embedded in the package by placing it in the root. If no executable is present, the Node.js LTS version included with Kofax RPA is used.

Executing Python (type: python)

Python requests are run using the `exec()` statement with a persistent private `global()` dictionary. The root of the connector package is added to the front of the sys.path to allow the connector to provide its own modules.

By default, parameters are converted into strings and escaped before they are inserted into the command line. The command line is not escaped.

Because the Python `exec()` statement does not allow the `return` statement at the top level, the request must assign its result to the global `rpa_return` variable instead. This variable is removed from the global scope between requests. If the interface is synchronous, the robot receives the value of the `rpa_return` variable and an exception value. Only one of these two fields has a non-empty value. An object is serialized to JSON before it is returned to the robot.

The `RPAConnector` name is reserved for internal use. Do not use it as a module name or create a directory named `RPAConnector` in the Connector package.

Kofax RPA uses the default Python interpreter. You can override this setting in the manifest by using a different name or path, such as "`python3.8`" or "`/usr/bin/python3`". It is left to the system to resolve the specified interpreter.

The package must be configured for the major Python version that is used, because of differences in the language between Python 2 and Python 3. Only Python 2.7 and Python 3 versions 3.5 and later are supported. The `six` package must be present in the Python installation.

Execute Connectors on local devices

To use your custom Connectors with the local device in your Desktop Automation robot, select the **Allow the use of Connectors** option in the **RoboServer Settings** application in the local device RoboServer.

Manifest

The `manifest.json` file must contain the following JSON elements:

Field	Status	Description
actions	required	Array of actions.
name	required	Name of the Connector shown in the Custom Action step.
python	optional	String Specifies the executable to run for Python actions. This setting applies to all platforms but can be overridden using one of the following platform-specific settings. Default: “`python`”

Field	Status	Description
python-windows	optional	String If present, specifies the executable to run for Python actions on the Windows platform.
python-linux	optional	String If present, specifies the executable to run for Python actions on the Linux platform.
python-support	optional	Integer Specifies the major version of Python that is used. Supported values are 2 for Python 2.7 and 3 for Python 3.x. Default: `3`

Action

Action format is detailed in the following table.

Action format

Field	Status	Description
name	required	Name of the Action.
type	required	Type of the Action. It must be one of the following: “`program`”, “`shell`”, “`node`,” or “`python.`”
parameters	optional	A set of parameters represented in the Custom Action step as input fields.
response	optional	A set of responses represented in the Custom Action step as output fields. If this field is omitted, a default set of responses is used depending on the definition of the Action.
commandline	required	A set of strings representing the parameters of the request command. For “`program`” actions, the first parameter is the executable. Parameters can be inserted using `%n` escapes (inserts the nth value from the parameters array). Use `%%` to insert a percent sign. Elements that are empty after parameter substitution are removed. For the “`node`” and “`python`” requests, it is permitted to send complete scripts.

Field	Status	Description
prune	optional	Boolean Indicates that elements expanded to an empty string should be removed from the command line array. Default: `true`
wait	optional	Boolean Indicates that the robot should wait for the action to finish. This setting is ignored for “`node`” and “`python`” actions or if the action has a response array. Default: `true`
stdout	optional	Boolean Enables the capturing of the standard output stream from a “`program`” or “`shell`” action in a variable. This field is ignored if “`wait`” is `false`. Default: `false`
stderr	optional	Boolean Enables capturing of the standard error stream from a “`program`” or “`shell`” action in a variable. This field is ignored if “`wait`” is false. Default: `false`

Parameter
Parameter format is detailed in the following table.

Parameter format

Field	Status	Description
name	required	Name of the parameter. The name is shown in the Custom Action step.
type	required	Type of the parameter. The type must be one of the following: “`string`” or “`number`.” “`number`” parameters are restricted to integral numbers.
default	optional	Default value of the parameter. Default: “” or 0 depending on the type of the parameter.
min	optional	Number The minimum permitted value for the parameter. This attribute is only supported for “`number`” parameters; it is ignored for other types. Default: -2147483648

Field	Status	Description
max	optional	Number The maximum permitted value for the parameter. This attribute is only supported for “number” parameters; it is ignored for other types. Default: +2147483647
optional	optional	Boolean Optional parameters can be left unassigned or empty in the robot. Default: false
escape	optional	Boolean Wrap the value in quotation marks and escape special characters. This attribute is currently supported for “node” and “python” string parameters; it is ignored in all other situations. Default: true

For numerical parameters, the default value must be between the min and max values (inclusive). All three settings must be in the range of -2147483648 to +2147483647 (inclusive).

Response
The following table describes response format.

Response format

Field	Status	Description
name	required	Name of the response. The name is shown in the Custom Action step.
type	required	Type of the response. The type must be one of the following: “string” or “number.” “number” response are converted to integral numbers.
optional	optional	Boolean Indicates that the response is not required to be present in the output. If the response is omitted, the variable is filled with its default value. Default: false

Field	Status	Description
default	optional	Default value for the parameter. This value is used if an optional parameter is omitted from the response. Default: “” or 0 depending on the type of the parameter.

Floating-point numerical values in the application’s response are converted to integers.

manifest.json examples

The following Linux-based examples use different connectors (shell, node, python, program) that call corresponding functions to add two numbers. The function parameters are defined in the order they are passed to this function, such as: `First number` and `Second number`. The result of the calculation is passed back to the robot in the variable assigned to the `Sum` value. The output value is assigned differently in different connectors.

python.connector

For this connector to work, install the latest Python version on the computer where a robot with the connector is running.

```
{"actions": [
    {
      "name": "Adder",
      "type": "python",
      "parameters": [
        {
          "name": "First number",
          "type": "number"
        },
        {
          "name": "Second number",
          "type": "number",
          "min": 1,
          "default": 1000
        }
      ],
      "response": [
        {
          "name": "Sum",
          "type": "number"
        }
      ],
      "commandline": [
        "rpa_return = { 'Sum': %1 + %2 }"
      ]
    }
  ],
  "name": "Adder Function (python)"
}
```

shell.connector

```
{
  "actions": [
    {
      "name": "Adder",
      "type": "shell",
      "parameters": [
        {
          "name": "First number",
          "type": "number"
        },
        {
          "name": "Second number",
          "type": "number",
          "min": 1,
          "default": 1000
        }
      ],
      "response": [
        {
          "name": "Sum",
          "type": "number"
        }
      ],
      "commandline": [
 "echo \"{ \\\"Sum\\\": $((%1 + %2)) }\""
      ]
    }
  ],
  "name": "Adder Function (shell)"
}
```

node.connector

```
{
  "actions": [
    {
      "name": "Adder",
      "type": "node",
      "parameters": [
        {
          "name": "First number",
          "type": "number"
        },
        {
          "name": "Second number",
          "type": "number",
          "min": 1,
          "default": 1000
        }
      ],
      "response": [
        {
          "name": "Sum",
          "type": "number"
        }
      ],
      "commandline": [
        "return { Sum: %1 + %2 };"
      ]
    }
  ],
  "name": "Adder Function (node)"
}
```

program.connector

This connector uses an executable program "add" called by the connector.

```
{
  "actions": [
    {
      "name": "Adder",
      "type": "program",
      "parameters": [
        {
          "name": "First number",
          "type": "number"
        },
        {
          "name": "Second number",
          "type": "number",
          "min": 1,
          "default": 1000
        }
      ],
      "response": [
        {
          "name": "Sum",
          "type": "number"
        }
      ],
      "commandline": [
        "./add", "%1", "%2"
      ]
    }
  ],
  "name": "Adder Function (program)"
}
```

Response definitions

Program

- If the action has a response array, the wait field is ignored and the robot waits until the program terminates. The standard output of the program is captured and parsed as a JSON string and the variables are filled with the respective fields from this string. The robot fails if the program does not produce a valid JSON string.
- If the action has no response array and the wait field is false, the step has no response variables.
- If the action has no response array and the wait field is true, the step has the following response variables:
  - `rc`: Return code of the program.
  - `stdout`: Standard output stream of the program (provided only if `stdout` is `true`).
  - `stderr`: Standard error stream of the program (provided only if `stderr` is `true`).

Regardless of the configuration of the connector, the standard output stream and standard error stream are logged at the INFO level to the Kofax RPA logs.

Shell

Shell actions are executed using CMD.EXE on Windows and bash on Linux.

- If the Action has a response array, the wait field is ignored and the robot waits until the shell terminates. The standard output of the shell is captured and parsed as a JSON string and the variables are filled with the respective fields from this string. The robot fails if the program does not produce a valid JSON string.

- If the Action has no response array and the wait field is false, the step has no response variables.
- If the Action has no response array and the wait field is true, the step has the following response variables:
  - `rc`: Return code of the shell.
  - `stdout`: Standard output stream of the shell (provided only if `stdout` is `true`).
  - `stderr`: Standard error stream of the shell (provided only if `stderr` is `true`).

Regardless of the configuration of the connector, the Standard output stream and Standard error stream are logged at the INFO level to the Kofax RPA logs.

NodeJS

The command line is executed in the context of a function. The command must use the return statement to return its results to the robot.

- If the action has a response block, it is expected to return an object or JSON string and the variables are filled with the respective fields from this object. The robot fails if an exception is thrown and not handled.
- If the action has no response block, the step has the following two response variables:

  `result`: If the statement completed successfully, this variable receives the value returned by the statement.

  `error`: If an exception is thrown and not handled in the JavaScript code, this variable contains the text of the exception.

Python

The command line is executed using the `exec()` function. The command must assign a value to the global variable `rpa_return` to return its results to the robot.

1. If the action has a response block, `rpa_return` must contain an object or JSON string and the variables are filled with the respective fields from this object. The robot fails if an exception is thrown and not handled.

1. If the action has no response block, the step has the following two response variables:
   - `result`: If the statement completed successfully, this variable receives the value of `rpa_return`.
   - `error`: If an exception is thrown and not handled in the Python code, this variable contains the text of the exception.

Implementation details

The following elements in the .zip file are reserved:

Type	Path	Description
file	/manifest.json	Manifest
file	/node	Node.js executable for the Linux platform. If this file is not present, the Node.js instance included with Kofax RPA is used.

Type	Path	Description
file	/node.exe	Node.js executable for the Windows platform. If this file is not present, the Node.js instance included with Kofax RPA is used.
directory	/node_modules	Location of the Node.js modules.
directory	/RPAConnector	Reserved for internal use.
directory	/node_modules/RPAConnector	Reserved for internal use.

Disconnect From Device

Use this step to disconnect from a remote device that you connected to using the Connect To Device step.

Properties

Device
Select the name of the device you want to disconnect from. Only the dynamic reference devices are in the list.

Document Transformation

The Document Transformation step helps you extract and use information from images and text documents. The Kofax RPA Document Transformation Service can process `.png`, `.jpeg`, `.jpg`, `.tif`, `.tiff`, `.pdf`, and `.txt` files. You can submit multiple documents either as a .zip archive or as a path to a folder with files. If you use the Document Separation feature in Kofax Transformation, Kofax RPA receives several documents that you can navigate to in the DT browser.

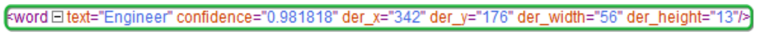
The Kofax RPA Document Transformation Service can also process Natural Language Processing (NLP) requests using the Sentiment project to help you detect the mood of the text, such as positive or negative, and to extract entities, such as company names, person names, and so on. You can use the Sentiment project to process customer reviews to understand whether customers are satisfied with the service or not. Moreover, you can use it to find all mentions of your company in an article. The Sentiment project can be used with KTT version 6.3.1 or later. See the Sentiment project in Predefined projects for details.

Document Transformation workflow

The Document Transformation action processes your graphical or PDF documents using a selected project. A project is a module that processes and transforms your documents using OCR and other specified operations.

The processing result is returned to the Desktop Automation robot and opened in the Document Transformation Browser in the Recorder view. The service forms an element tree with all extracted information. Note that in a multi-page document, you can browse through the pages using the Previous and Next buttons on the DT browser toolbar. See DT browser for details.

Elements in the tree contain confidence levels for the OCR results and other extraction results defined by the project. The `confidence` attribute can contain values from zero to one where the most confident is one.



```
<word ⊟ text="Engineer" confidence="0.981818" der_x="342" der_y="176" der_width="56" der_height="13"/>
```

Derived attributes such as `der_x` help you find the element and can be used in finders.

Once the transformed document is in the editor, you can determine whether you want to perform the validation of the transformation results. If you are satisfied with the transformation results without any validation, you can extract and use the data in the document.

Validation is performed by the Document Transformation Thin Client. Click ⇅ in the Document Transformation Browser to send the document to the specified Thin Client. A unique URL is generated and returned to the robot. The robot extracts the URL and uses it to send the document to a validation user, such as via email. The validation user clicks the URL, enters credentials, and after that the document with the extracted data opens. The validation user inspects the transformed document and, if needed, modifies extracted information in the document.

When validating documents, the user can enable the Online Learning feature to increase the rate of field recognition on similar documents. This feature is based on remembering the layout of a sample document, such as an invoice. By using automatic field completion, manually typing or selecting the correct value in the document, the user contributes to the knowledge base, which improves extraction results when the user works on a similar document next time.

When validation is finished, the validation user marks the document as valid. When the document is marked valid, it is used as an argument for a robot specified in the Callback URL in the **Document Transformation** action.

Step properties

Action

Select an action to perform using the Kofax RPA Document Transformation Service.

Service URL

Specify a URL and a port if necessary for the computer running the Document Transformation Service. If the service is installed locally, enter `localhost` in this field. The URL must include the `http://` or `https://` prefix. If you use `https`, the web hosting service should have a certificate accepted by well-known certificate authorities.

Project Type

- **Default Project**: This option provides a set of predefined projects. See Predefined projects.
- **Custom Project**: When you select this option, specify the path to the project to process your documents in **Custom Project Path**.

Document Source

Select how the robot locates a document to process.

- **Local File**: Enter the path to one or more documents to process in **File name**. Use either a full path to an image file, .zip archive, folder with files, or another file of the supported format accessible form the computer running a robot.
- **Robot File System**: Enter the path to the configured file system and the file name, such as **myshare/doctotransform.pdf**. The file system name must correspond to that specified in the Robot File System section in Management Console.
- **Binary Variable**: Specify a binary variable that contains a document.

When a path to multiple documents is specified, you can navigate between the documents using the DT browser toolbar buttons.

Meta data

Select this option to pass additional data to the Document Transformation Service.

This data is added to the input document as XValues, so it can be used by the Document Transformation Service project. The projects usually use the data to tune or control the analysis. Common use cases for this option are language settings or customer identification. The XValues are available in the device tree after the processed documents are received back from the Document Transformation Service.

Consult the developer of your Document Transformation Service projects to identify which values are supported by a specific project.

> **Note** You can add more than one Key/Value pair to the **Meta data** property.
>
> If any **Key** appears more than once in the list, the **Value** of the last occurrence is used.

Validation URL

Select this option to specify a URL for the thin client service. This property is required to send processed documents for validation. The URL is specified in the `ValidationService` property of the Document Transformation Service. The URL may look similar to the following:

```
http://localhost:8082
```

Callback URL

Select this option to specify a REST Robot URL for the thin client service to call after a document is validated. When validation is complete, this URL is used to start a robot by the Management Console. The URL must include the Management Console address along with a path to the robot to run. The URL may look similar to the following:

```
http://localhost:8080/ManagementConsole/rest/run/Default project/
binaryInputAndWait.robot
```

You can find a valid callback URL in the Management Console by clicking the **REST** button for the given robot. Such a robot should have an input variable called `document` with an attribute `doc` of type binary. When the robot is called, the `doc` attribute of the document contains the transformed and validated document. If the Management Console needs credentials for login, they can be provided in the URL as follows:

```
http://user:password@localhost:8080/ManagementConsole/rest/run/Default
project/binaryInputAndWait.robot
```

See REST for more information.

Predefined projects

You can edit transformation projects supplied with Kofax RPA and your custom projects in the KTT Project Builder installed by Kofax RPA. Once you open the Project Builder you can access its documentation.

Barcode project
The purpose of this project is to extract all barcodes from the document.

To change the Barcode project settings, perform the following steps.

1. Locate the Kapow_Barcodes.fpr project file.
2. Open it in Project Builder.
3. Select the class **Default** in the Project tree on the left side.
4. Click the eye symbol to open the details.
5. Under **Locators**, double-click the BL barcode locator. By default, the locator is configured to auto-detect the barcode type.
6. Clear the **Auto detect** option under **Type** and select a specific type.
7. By default, the locator is configured to automatically detect the orientation. Clear the **Auto detect** option under **Orientation** and select a specific orientation.
8. By default, the locator is configured to look for barcodes on all pages of a document. To limit the barcode detection to a specific set of pages, select the **Regions** tab and modify the **Enable locator for** settings accordingly.
9. When you finish editing the project, close all dialog boxes and click **Save Project** on the Project tab.

Invoice projects (Invoice Sales TAX and Invoice VAT)

These projects are designed to extract invoices from the USA and it also supports sales tax. The projects require setup of ERP master data to extract the vendor properly. The master data for vendors and internal companies needs to be provided as a csv file. The projects contain a `vendors.csv` file and an `internal_venders.csv` file that can be adapted to the company-specific vendors.

To provide and configure master data, perform the following steps.

Prerequisites

- The vendors file is a semicolon-separated document called `Vendors.csv`. The file must have the following columns:
  - VendorID (required)
  - CompanyCode (optional)
  - Name (required)
  - Street (required)
  - City (required)
  - ZIP (required)
  - PostBox (optional)
  - Country (required, 2-character country code)
  - FIDNumber (optional)
  - Phone (optional)
  - Fax (optional)
  - URL (optional)
  - Email (optional)
- The internal vendors file must be called `Vendors_Internal.csv`. It is a semicolon-separated file with the same columns as `Vendors.csv`. Internal vendors are those that are internal within the customer's enterprise. This file is used to exclude those from the vendor results, as they are easy to confuse with the bill-to address on any normal external invoice.

1. Locate the `Kapow_Invoices_SalesTax.fpr` project file or the `Kapow_Invoices_VAT.fpr` project file and open it in the Project Builder.
2. Open **Project Settings**.
3. Select the **Databases** tab.

   Note that the 2 Fuzzy Database items are flagged as red, because the path is incorrect.
4. Double-click **Vendors**.
5. Select the path to the `Vendors.csv` on the network share.
6. Click **OK** to close the dialog box. The file is imported.
7. Double-Click **Vendors_Internal**.
8. Select the path to the `Vendors_Internal.csv` on the network share.
9. Click **OK** to close the dialog box. The file is imported.
10. When you finish editing the files, close all dialog boxes and click **Save Project** on the **Project** tab.

Online Learning

By default, the Online Learning feature, which helps increase the rate of field recognition on similar documents, can only be enabled for the Invoices projects. When specifying the path to the folder where your training documents are stored, ensure that the folder already exists. If it does not exist, you receive a notification prompting you to create it. To proceed, click **Yes**.

Language project

The purpose of this project is to identify the language a document is written in.

This project is not configurable.

OCR project

The purpose of this project is to return full text OCR results for the document. Note that this project does not include the validation process by default.

To change the OCR recognition language from the default (English), perform the following steps.

1. Locate the Kapow_OCR.fpr project file.
2. Open it in Project Builder.
3. Click **Project Settings**.
4. In the **Project Settings** dialog box, select the **Recognition** tab.
5. Select the **FineReader** page profile.
6. Check the desired language.
7. Close all dialog boxes and click **Save Project** on the **Project** tab.

US Address Extraction Project

The purpose of this project is to extract all US addresses from a document.

This project is not configurable.

Sentiment project

The purpose of this project is to deduce the mood of the text, such as positive or negative, and to identify entities, such as company names, person names, and so forth. In the transformed document, the mood is displayed in the `Sentiment` field in a number from -1 to 1 where -1 is completely negative and 1 is completely positive. For example, `0.257545` represents a slightly positive text.

By default, the project processes English language texts. Language bundles for the Sentiment project are distributed separately in three `.msi` installers:

- Kofax NLP Western Default Language Bundle: English, French, German, Portuguese, Spanish
- Kofax NLP Western Extended Language Bundle: Dutch, Italian, Romanian
- Kofax NLP Additional Language Bundle: Japanese, Korean, Mandarin

Language bundles are not installed by default. To use any available language, install the appropriate language bundle. For example, to use the English language, install the Kofax NLP Western Default Language Bundle.

The bundles are installed as Windows programs and have no options. To remove a language bundle, open the **Programs and Features** or **Apps and features** from the Control Panel, select the bundle and click **Uninstall**.

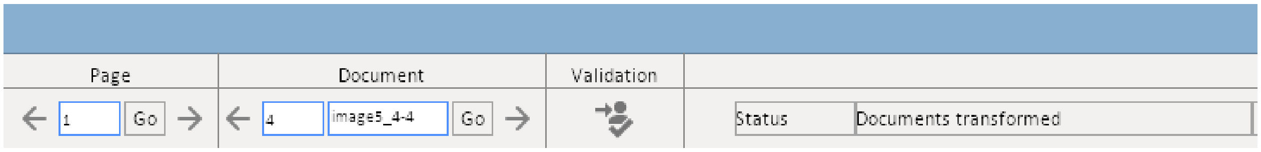
To change the recognition language from default (English), perform the following steps.

1. Open the Sentiment project in the Project Builder.
2. Click **Project Settings**.
3. In the **Project Settings** dialog box, click the **Properties** button.
4. Clear the default English language option.
5. Select the language.
6. Close all dialog boxes.
7. In the Project Tree, select the **Default Project Class** definition.
8. Scroll down and select the desired language from the language list.
9. Click **Save Project** on the **Project** tab.

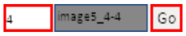
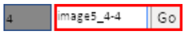
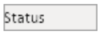
Customized project
When you select this option, specify the path to the project to process your documents in the **Project Name** property, such as `c:\rpa\ocr`. The project link must be a locally accessible folder on the Document Transformation host and not on a computer running Design Studio.

DT Browser and Action Steps

The DT (Document Transformation) browser shows transformation results and helps you work with extracted data in the document. The following table explains the DT browser toolbar elements.



Button	Description
The **Page** section helps you navigate in a multi-page document	
←	Navigates one page back in a multi-page document.

Button	Description
→	Navigates one page forward in a multi-page document.
	Navigates to the specified page in a multi-page document.
The **Document** section helps you navigate in multiple and separated documents	
←	Navigates to the previous document.
→	Navigates to the next document.
	Navigates to the document by number.
	Navigates to the document by name. • For normal documents, specify the filename without extension, such as `mydocument` for the `mydocument.pdf` file. • For separated documents, specify the filename without extension followed by an underscore and a page range suffix. For example, if a four-page `mydocument.pdf` file is split into half two pages each, the two resulting documents are called `mydocument_1-2` and `mydocument_3-4`.
The **Validation** section helps you validate a document	
	Sends a document for manual validation to the specified Document Transformation Thin Client server.
	Status of the transformed document with error description if any.

Application action steps

All of the actions that appear on the toolbar are also available from the context menu when you right-click the DT Browser tab. The following table lists and describes those actions that are available only from the context menu.

Action step	Description
Count Pages	Gets the number of pages in the currently active document.
Count Documents	Gets the number of document in the batch.
Close	Closes the application window.

Action step	Description
Get Document	Extracts a particular document from the batch. You can use this action to extract a subdocument when your Document Transformation Service project is configured to use Document Separation. **1.** In the step properties, select a name of the subdocument to extract or its index (number in the batch). The **Index** option takes precedence over the **Name** option, so if you specify both, the index is used to select the document. **2.** Select a file format to extract a document: **Get Image** (.tif) and/or **Get XDoc** (.xdc). **3.** Specify a binary variable to store the result. Then, you can use the binary data in several different ways, such as send it to Kofax TotalAgility with the Create Document action or use the Write File step to write the data into a file on a local or remote computer. For example, when using the Write File step, consider the following for a file name for the file to contain the extracted document, : • If you selected "Get Image" only, use the extension **.tif**. • If you selected "Get XDoc" only, use the extension **.xdc**. • If you selected both formats, use the extension **.zip**. In this case, an archive is created containing the extracted document in two formats. The .xdc format can be read with the XDoc Browser application included in your Document Transformation Service installation.

Component actions

Action	Description
Get Field Value	Extracts document field value to the specified variable.

Enter Text

In this step your robot can type text into a text field. You can provide the necessary text directly in the **Text** field of the step or use the text from a variable. This is an application-level step and it is available by right-clicking the application tab. Note that if you do not click the text field or create an appropriate finder before entering the text, this step inserts the text to the first available text field in the application tree.

You can input text in the selected field by right-clicking the field and selecting **Replace Text** on the shortcut menu. This command creates an Input step that contains a finder and all necessary actions necessary to replace a text in the selected field.

Note When using Enter Text in the Excel on a remote device, do not leave the Excel spreadsheet in edit mode (the word Edit appears in the lower-left corner of the Excel program window) if you want to work with it in the following steps. To exit the edit mode, perform one of the following.

- Press ENTER using the Press Key step. Excel exits Edit mode and selects the cell directly below the current cell.
- Press TAB using the Press Key step. This stops Edit mode and selects the cell to the right of the current cell.
- Click a different cell.
- Press F2 using the Press Key step.

See Microsoft documentation for more information.

Enter text with the virtual input driver

When you enable the virtual input driver on the automated device (see Activate the virtual input driver in Configure Desktop Automation Service), Enter Text steps in Device Automation on Windows automatically use this driver for entering text. The text is entered as though via a hardware keyboard, and is subject to keyboard layout interpretation by the operating system. It is only possible to enter text, which can be typed with a physical keyboard using the application's active keyboard layout. It is not possible to enter all Unicode characters.

Properties

Name
Name of the step.

Finder
Device: Select the name of the automation device.

Application: Specify the name of the application the action is performed in.

Text
Either type in the text directly or specify a variable with text. The variable name must be preceded by an equal sign, such as `=EnterTextVariable`.

Excel

With this step you can perform some operations on Excel spreadsheets using the built-in Excel driver. For the list of available actions and other information, see Built-in Excel Driver.

Note The built-in Excel driver is available only for local automation device.

Properties

Action

Create File

Select this option to create a new Excel document.

Open File

Select this option to open an existing Excel document. After you select **Open File**, specify the full path to the spreadsheet in **Worksheet Path**. For example, `c:/documents/myspreadsheet.xlsx`

Visible Area

Specifies the number of rows and columns to show when an Excel document opens (default 45 rows and 20 columns).

> **Note** The specified number of columns is the minimum number. The Excel driver loads either the specified number of columns or more to fill the entire screen width.

Extract Clipboard

This step extracts information from the clipboard to a variable.

Properties

Device
Select the name of the automation device.

Alias
Alias of the component.

Evaluate Expression step
Optional. Contains a list of conversion functions suggested for use on a value. The function list suggested for a particular value is limited, depending on the value type. To specify a conversion function, in the step context menu, click **Evaluate Expression step**, and then do either of the following:

- Select the required function from the list of suggested functions.
- Click **Plain** to manually type the required function.

For more information on the supported functions and examples, see Expressions.

You can add one or more Evaluate Expression steps.

Store Current In step
Name of the variable of specified type to store the extracted/converted value.

You can add one or more Store Current In steps.

The Extract Clipboard step can also contain the following set of action steps:

- Conditional step
- Group step
- Throw step

- Write Log step

Extract Image

This step extracts an image from the selected area on the screen and saves it in a binary type variable. You can select an image in the **Recorder View** by pressing the mouse button and drawing a selection rectangle. You can modify the rectangle by dragging its sides or just draw a new rectangle.

Properties

Component
Alias: Alias of the finder.

Base Finder: Specify the component to use.

Component: Set the application component name, such as button, window, pane, etc.

Contents: A regular expression to find an element by its value. This parameter is usually used when working with a browser in the Desktop Automation robot.

Image: A screen shot of the selected area stored as an image.

Output Variable
Specify a binary variable to store the image.

> **Note** It is not possible to extract an image from cell elements in tables.

See Nine-Grid Image Finder for more information about image finders.

Extract Text From Image

This step helps you extract text from an image using the selected OCR engine. You can select either Tesseract (default) or OmniPage engine to capture text from images. For Tesseract, only English language is included in the installation. OmniPage includes all supported languages in the installation. Robots created in Kofax RPA prior to version 11.1 use Tesseract engine. See Change Default OCR Language for Desktop Automation for more information.

When you use this step without the Desktop Automation Service, such as in the built-in browser, change the OCR settings in the `ocr.cfg` file. See Extended OCR Settings for more information.

> **Note** It is not possible to extract text from cell elements in tables.

Properties

Name
Name of the step.

Variable
Specify a variable to store the extracted text.

Text Font Size

- Small: Font smaller than 12px.
- Medium (Default): Font size between 12px and 24px.
- Large: Font more than 24px.

Note that your font size choice affects the speed of text analysis and recognition. For example, when a large image is analyzed, selecting Large speeds up the analysis two or three times compared to Medium. On the contrary, selecting Small reduces recognition speed two or three times compared to Medium. Try different settings and choose the best in terms of speed and recognition results.

Image Binarization

- Auto: Tesseract algorithm is used to prepare an image for text recognition.
- Custom: Kofax RPA algorithm is used to prepare an image for text recognition. See Tweaking Text Recognition for more information.

  Threshold Delta

  None

  Positive

  - Small
  - Medium
  - Large

  Negative

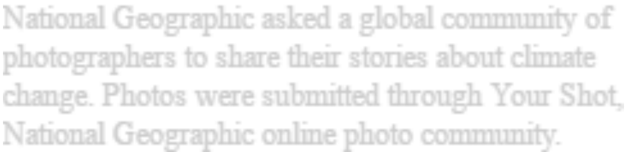
  - Small
  - Medium
  - Large

Tweaking Text Recognition

The following information is applicable to Tesseract engine only.

By default, Kofax RPA uses the Tesseract algorithm for OCR that produces acceptable results most of the time. Before the text is recognized, the algorithm converts an image with text to a black-and-white image and performs some other adjustments to make the text stand out. In case the recognizable text blends with the background and recognition result is not good, you can change to **Custom** in the **Image Binarization** option and adjust the **Threshold Delta** options to produce acceptable results.

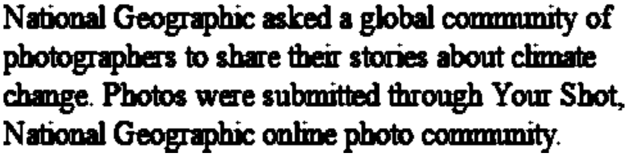
The following is an image copied from the screen for recognition.



National Geographic asked a global community of
photographers to share their stories about climate
change. Photos were submitted through Your Shot,
National Geographic online photo community.

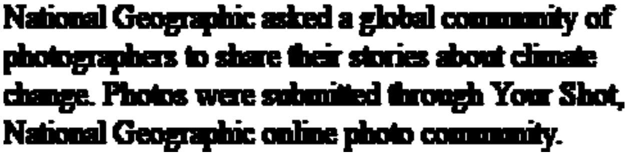
The following are internal image adjustment results from the Kofax RPA algorithm for text recognition. Each image is labeled with a set of Threshold Delta options. In complicated cases, try different options and choose the best in terms of recognition results.

Threshold Delta: None



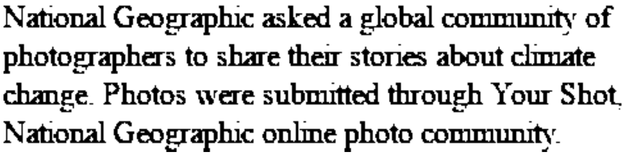
National Geographic asked a global community of photographers to share their stories about climate change. Photos were submitted through Your Shot, National Geographic online photo community.

Threshold Delta: Positive Medium



National Geographic asked a global community of photographers to share their stories about climate change. Photos were submitted through Your Shot, National Geographic online photo community.

Threshold Delta: Negative Medium



National Geographic asked a global community of photographers to share their stories about climate change. Photos were submitted through Your Shot, National Geographic online photo community.

You can edit extended OCR settings in the `ocr.cfg` file. See Extended OCR Settings for more information.

Extended OCR Settings

Kofax RPA provides optical character recognition (OCR) functionality to extract text from images and to automate applications with limited or no automation API.

OCR is a complicated process and recognition results depend on many factors, such as screen fonts, background and foreground color, text size, and so on. Kofax RPA installs the `ocr.cfg` file that contains some configuration settings you can use to alter recognition results. The file includes detailed description of configuration settings. The `ocr.cfg` file is located in the Kofax RPA installation directory as follows.

- On the Windows-based automated computer with installed Desktop Automation Service:

  `DesktopAutomationService\lib` in the Desktop Automation service installation directory. Example:

  ```
  C:\Program Files (x86)\Kofax RPA DesktopAutomation 11.1.0
  x32\DesktopAutomationService\lib
  ```

- On the local Windows-based computer to use with the built-in browser:

  `nativelib\hub\windows-x32\<build number>\lib`* in the Kofax RPA installation directory. Example:

  ```
  C:\Program Files\Kofax RPA 11.1.0 x64\nativelib\hub\windows-x32\166\lib
  ```

- On the local Linux-based computer to use with the built-in browser:

  `nativelib/hub/linux-x64/<build number>/lib` in the Kofax RPA installation directory. Example:

  ```
  Kofax RPA_11.1.0_x64/nativelib/hub/linux-x64/166/lib
  ```

* The build number is different in different versions of the program.

Change OCR engine and language

Change OCR engine

Kofax RPA uses either Tesseract (default) or OmniPage engine to capture text from images. To change the OCR engine from default, perform the following steps.

1. Locate the `ocr.cfg` file on your computer.
2. Open `ocr.cfg` in a text editor and locate the `engine_type` option.
3. Specify an OCR engine, such as `omnipage` as a value as follows `engine_type = omnipage`.

   If you want to use the default OCR engine (Tesseract), either specify `tesseract` as a value in the `engine_type` option or just delete any value from this option.

Change OCR language

1. Locate the `ocr.cfg` file on your computer.
2. In the text editor, open `ocr.cfg` and locate the `default_language` option.
3. Either replace `eng` with another language code, for example `jpn` or, if you want to use more than one language, add `jpn` using the plus sign, such as `default_language=eng+jpn`. Save and close the file.

OmniPage includes all supported languages in the installation. For Tesseract, only English language is included in the installation. To add more languages for UI recognition by Tesseract, see step 1 and 2 in the "Change or add UI recognition language for Tesseract" section in Tree Modes.

Important OCR engine and language settings for the Desktop Automation Service are specified in the Desktop Automation Service configuration window separately for each computer running the service. See Configure Desktop Automation Service for details.

Image preprocessing

The following information is applicable to Tesseract engine only.

Before the actual OCR process is initiated for an image, the image is preprocessed using a particular algorithm. In the ocr.cfg file, the `preparation` setting defines the algorithm to use. By default, it is set to `normal`.

Note In case the default preprocessing algorithm gives a result that you find unsatisfactory, you may try switching to a different algorithm. To do so, change the value of `preparation` to `10.2` and save the changes.

Train Tesseract

Kofax RPA uses either the Tesseract or OmniPage OCR engine to capture text from images and to perform Intelligent Screen Automation (ISA). OmniPage includes all supported languages in the installation. For Tesseract, only English language is included in the installation. You can change the language in Tesseract by supplying a `.traineddata` file for the corresponding language.

If you experience issues recognizing specific languages or letters, you can train Tesseract to read the fonts properly.

The supplied by Kofax RPA scripts for preparing training data are intended for Linux operating systems. Currently, Tesseract version 3.4.0 is used.

Prerequisites

Make sure your system complies with the following prerequisites before creating training data.

System requirements for Ubuntu-based systems

Install the following libraries using the `sudo apt-get install` command as follows.

```
sudo apt-get install libicu-dev libpango1.0-dev libcairo2-dev git
```

Training prerequisites

Go to `nativelib/hub/linux-x64/<hub_id>/tools/tesseract_train/bin` in the Kofax RPA installation directory and run the `prepare.sh` script. For example:

```
$ cd /home/user88/Kofax_RPA/nativelib/hub/linux-x64/574/tools/
tesseract_train/bin
$ ./prepare.sh
```

Automatic training

Choose this mode if you have the TTF font file used in the UI that you want to recognize. This mode is simpler than the manual training mode. To create a training data file for the desired font, run the `tesseract_auto.sh` script located in `tesseract_train/bin` folder specifying the language code, the font name, and the font file directory as follows.

Note Make sure you execute the script from `tesseract_train/bin` working directory.

```
$ ./tesstrain_auto.sh --lang eng --fontlist 'Envy Code R' --fonts_dir ..
```

Once you execute the script, you should see the following message.

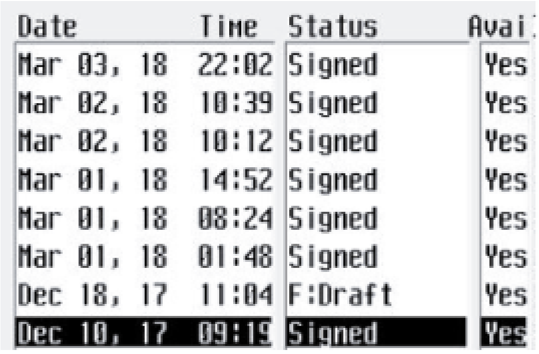
```
Moving /tmp/tmp.OtEqYbS3qV/eng/eng.traineddata to ../output
Completed training for language 'eng'
```

Now you can use the trained data file in Kofax RPA. See Change Default OCR Language in Configure Automation Device and "Change or add UI recognition language" under the Intelligent Screen Automation topic in Tree Modes.

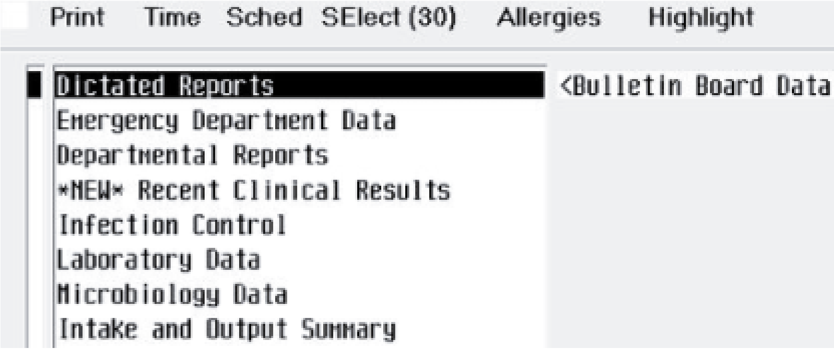
Manual training

Choose this mode if you do not have the TTF font file used in the UI (so the Auto mode cannot be applied), but you have many UI screen shots that include all alphabet characters you want the robot to recognize. Unlike the automatic mode, where a training image file is created automatically by the script, you need to manually create a training image. It requires some time and diligence to craft such a file.

Perform the following steps to create a training data file for Tesseract. The file should contain all characters (uppercase and lowercase letters, numbers, punctuation marks, and more) that need to be present in the final training data file. The partial example below shows how to create training data for use with the following UI.



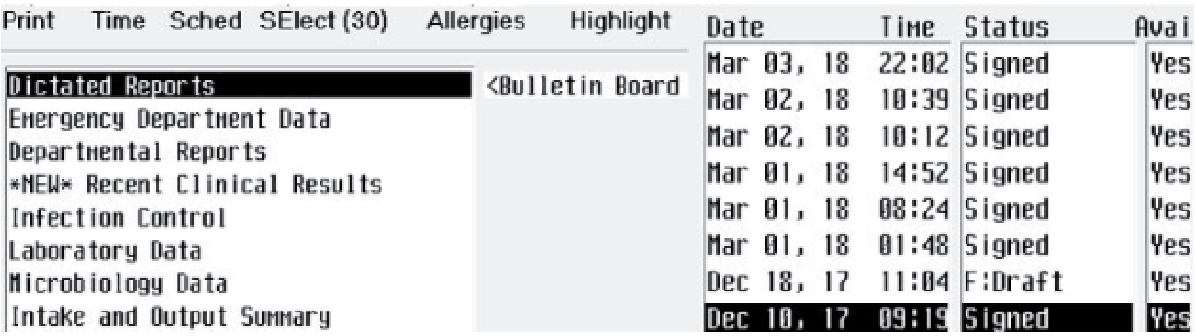
Date	Time	Status	Avai
Mar 03, 18	22:02	Signed	Yes
Mar 02, 18	10:39	Signed	Yes
Mar 02, 18	10:12	Signed	Yes
Mar 01, 18	14:52	Signed	Yes
Mar 01, 18	08:24	Signed	Yes
Mar 01, 18	01:48	Signed	Yes
Dec 18, 17	11:04	F:Draft	Yes
Dec 10, 17	09:19	Signed	Yes



Dictated Reports
Emergency Department Data
Departmental Reports
NEW Recent Clinical Results
Infection Control
Laboratory Data
Microbiology Data
Intake and Output Summary

<Bulletin Board Data

1. Determine the full character set to be used. Bear in mind when creating a training file that a minimum number of samples for each character is five. For the most frequently used characters, include additional samples.
2. Put all parts of the UI screen shots that will be used for training into a single TIFF file. You can use any image editor for this operation. In this example, we limit the target alphabet to 10-15 English letters. In production, make sure that you have examples of all letters.

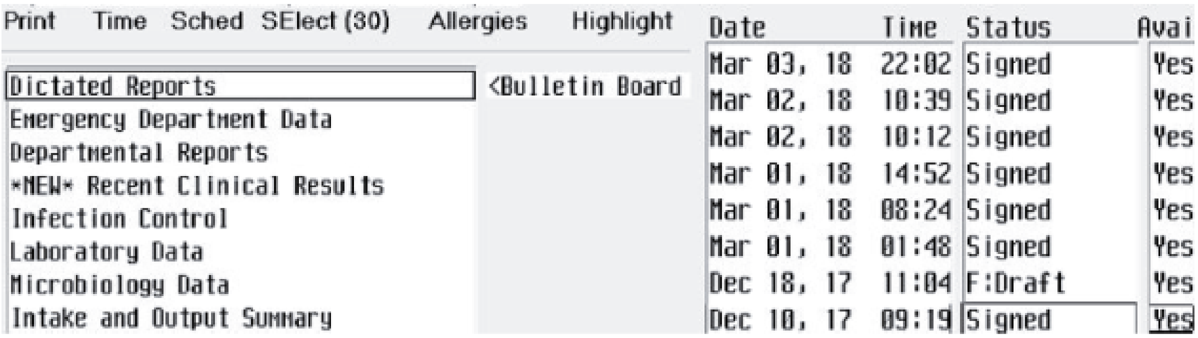


Dictated Reports
Emergency Department Data
Departmental Reports
NEW Recent Clinical Results
Infection Control
Laboratory Data
Microbiology Data
Intake and Output Summary

<Bulletin Board

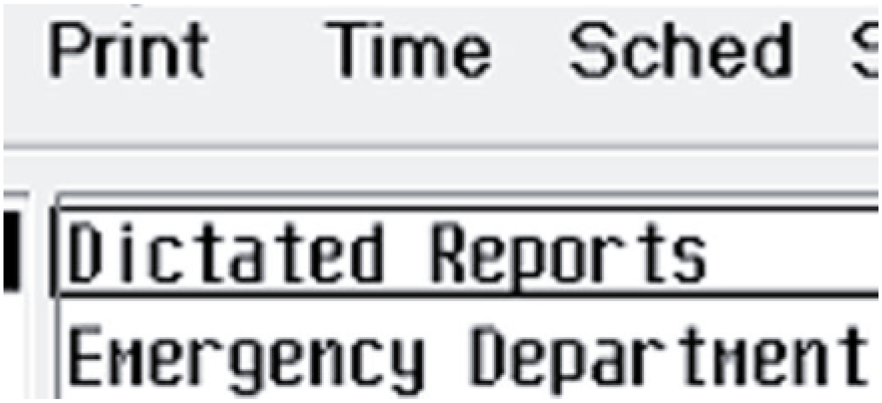
Date	Time	Status	Avai
Mar 03, 18	22:02	Signed	Yes
Mar 02, 18	10:39	Signed	Yes
Mar 02, 18	10:12	Signed	Yes
Mar 01, 18	14:52	Signed	Yes
Mar 01, 18	08:24	Signed	Yes
Mar 01, 18	01:48	Signed	Yes
Dec 18, 17	11:04	F:Draft	Yes
Dec 10, 17	09:19	Signed	Yes

3. Select areas with inverted colors and restore them to normal.



```
Print   Time  Sched  SElect (30)   Allergies   Highlight   Date          Time   Status       Avai
                                                           Mar 03, 18    22:02  Signed       Yes
Dictated Reports                         <Bulletin Board   Mar 02, 18    10:39  Signed       Yes
Emergency Department Data                                  Mar 02, 18    10:12  Signed       Yes
Departmental Reports                                       Mar 01, 18    14:52  Signed       Yes
*NEW* Recent Clinical Results                              Mar 01, 18    08:24  Signed       Yes
Infection Control                                          Mar 01, 18    01:48  Signed       Yes
Laboratory Data                                            Dec 18, 17    11:04  F:Draft      Yes
Microbiology Data                                          Dec 10, 17    09:19  Signed       Yes
Intake and Output Summary
```

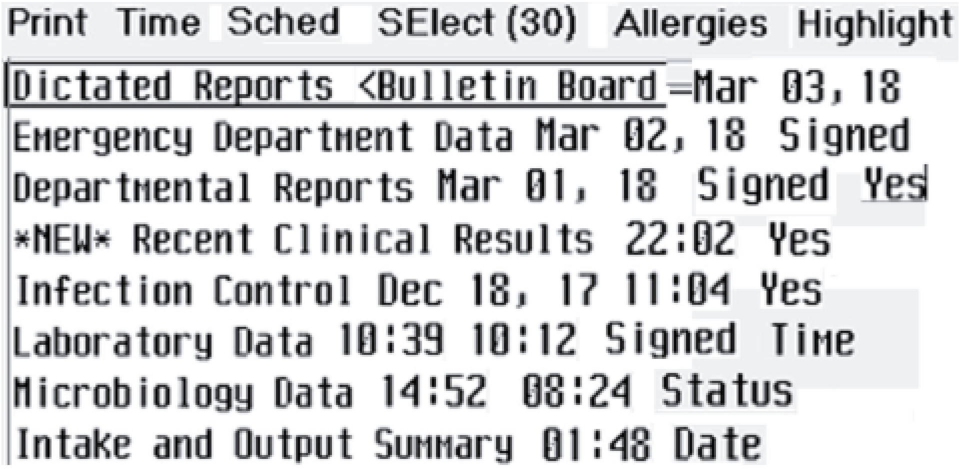
4. Scale the image using cubic interpolation so that the uppercase letters have height equal to 36px. For this particular example, we upscaled the image 2.97 times (showing only a part of the image).



```
Print    Time   Sched  S

Dictated Reports
Emergency Department
```

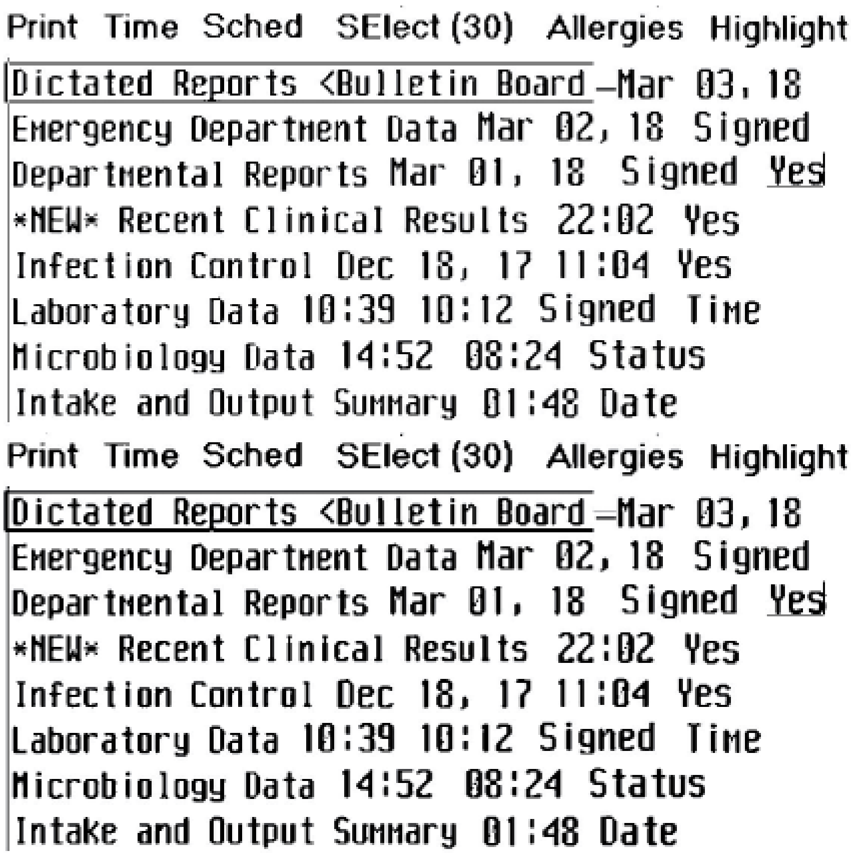
5. Rearrange words to have easily detectable text lines without large spaces between text regions. Remove text that is redundant in your judgment, as in the following example (downscaled to fit the page).



```
Print  Time  Sched  SElect (30)  Allergies  Highlight
Dictated Reports <Bulletin Board =Mar 03, 18
Emergency Department Data Mar 02, 18  Signed
Departmental Reports Mar 01, 18  Signed  Yes
*NEW* Recent Clinical Results  22:02  Yes
Infection Control Dec 18, 17 11:04  Yes
Laboratory Data 10:39  10:12  Signed  Time
Microbiology Data 14:52  08:24  Status
Intake and Output Summary  01:48 Date
```

6. Convert the image to grayscale and apply a threshold color effect that produces text of the best quality. It might be difficult to select the proper threshold. Consider applying two or more different

thresholds and copy the result images into a single TIFF file. The training image would contain many different representations of the same letter. In this example we applied 125 and 150 thresholds in GIMP editor and copied the images into one file. You may notice that text in the upper half of the image is thinner than in the bottom half (downscaled to fit the page).



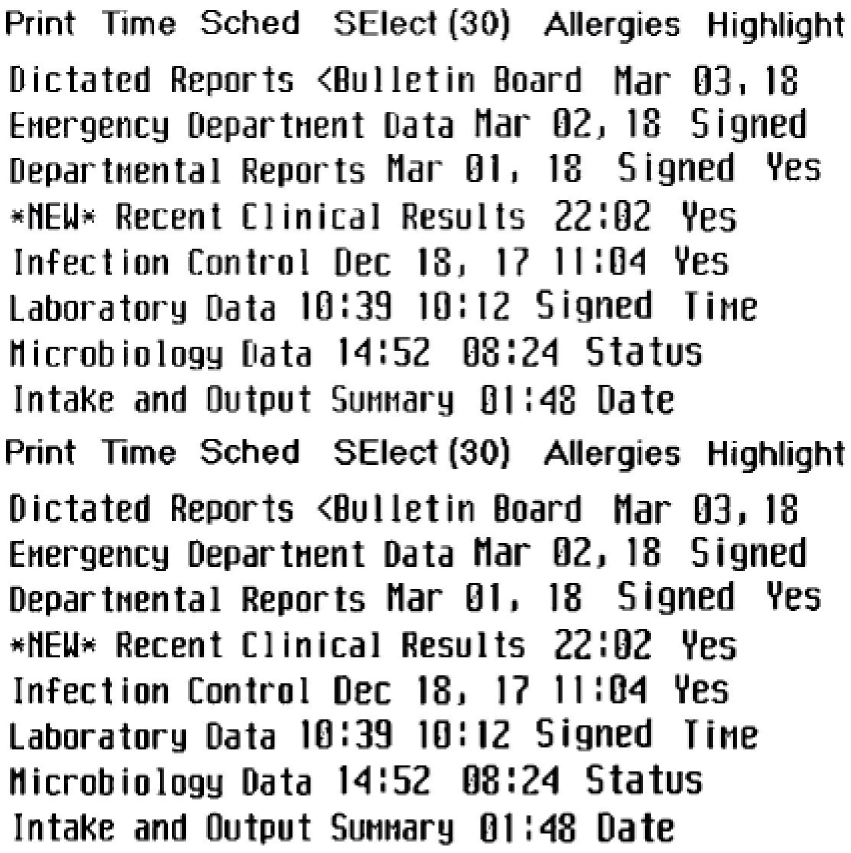
Print Time Sched SElect (30) Allergies Highlight

Dictated Reports <Bulletin Board –Mar 03, 18
Emergency Department Data Mar 02, 18 Signed
Departmental Reports Mar 01, 18 Signed Yes
NEW Recent Clinical Results 22:02 Yes
Infection Control Dec 18, 17 11:04 Yes
Laboratory Data 10:39 10:12 Signed Time
Microbiology Data 14:52 08:24 Status
Intake and Output Summary 01:48 Date

Print Time Sched SElect (30) Allergies Highlight

Dictated Reports <Bulletin Board –Mar 03, 18
Emergency Department Data Mar 02, 18 Signed
Departmental Reports Mar 01, 18 Signed Yes
NEW Recent Clinical Results 22:02 Yes
Infection Control Dec 18, 17 11:04 Yes
Laboratory Data 10:39 10:12 Signed Time
Microbiology Data 14:52 08:24 Status
Intake and Output Summary 01:48 Date

7. Manually remove noise as in the following example (downscaled to fit the page).



Print Time Sched SElect (30) Allergies Highlight

Dictated Reports <Bulletin Board Mar 03, 18
Emergency Department Data Mar 02, 18 Signed
Departmental Reports Mar 01, 18 Signed Yes
NEW Recent Clinical Results 22:02 Yes
Infection Control Dec 18, 17 11:04 Yes
Laboratory Data 10:39 10:12 Signed Time
Microbiology Data 14:52 08:24 Status
Intake and Output Summary 01:48 Date

Print Time Sched SElect (30) Allergies Highlight

Dictated Reports <Bulletin Board Mar 03, 18
Emergency Department Data Mar 02, 18 Signed
Departmental Reports Mar 01, 18 Signed Yes
NEW Recent Clinical Results 22:02 Yes
Infection Control Dec 18, 17 11:04 Yes
Laboratory Data 10:39 10:12 Signed Time
Microbiology Data 14:52 08:24 Status
Intake and Output Summary 01:48 Date

8. Save the image in TIF or TIFF format without compression, such as `MyFont.tif`.

9. Make a box file. The box file is a text file that lists characters in the training image, one per line, with the coordinates of the bounding box around the image. See the "Training Tesseract - Make Box Files" page in Tesseract project on GitHub: https://github.com.

   Copy the box text and put into a new file, such as `MyFont.box`.

   In our example, the box file should start with the following lines:

```
P 15 1076 39 1108 0
r 41 1076 53 1100 0
i 57 1076 62 1108 0
n 68 1076 89 1100 0
t 92 1076
...
```

10. Go to `tesseract_train/bin` folder and run `tesstrain_manual.sh` script, specifying language code and paths to the TIF image and box file, for example:

```
$ ./tesstrain_manual.sh --lang eng --box_file ../MyFont.box --
training_image ../MyFont.tif
```

   Once you execute the script, you should see the following message.

```
Moving /tmp/tmp.OtEqYbS3qV/eng/eng.traineddata to ../output
```

Now you can use the trained data file in Kofax RPA. See Change Default OCR Language in Configure Automation Device and "Change or add UI recognition language" under the Intelligent Screen Automation topic in Tree Modes.

More information is available on the Tesseract wiki pages on the GitHub website.

Extract Tree as XML

This step helps you extract a part of the application tree and save it in a variable as an XML string.

Properties

Include Derived Attributes
You can choose to include or exclude derived attributes. For example, If your goal is to extract the pure HTML code from an HTML node, select **excl. Derived Attr. into** on the shortcut menu when inserting the step in the Recorder view, or clear the **Include Derived Attributes** option in the step in the workflow view. For more information about derived attributes, see "Application Tree" in Introduction to Desktop Automation.

Evaluate Expression step
Optional. Contains a list of conversion functions suggested for use on a value. The function list suggested for a particular value is limited, depending on the value type. To specify a conversion function, in the step context menu, click **Evaluate Expression step**, and then do either of the following:

- Select the required function from the list of suggested functions.
- Click **Plain** to manually type the required function.

For more information on the supported functions and examples, see Expressions.

You can add one or more Evaluate Expression steps.

Store Current In step
Name of the variable of specified type to store the extracted/converted value.

You can add one or more Store Current In steps.

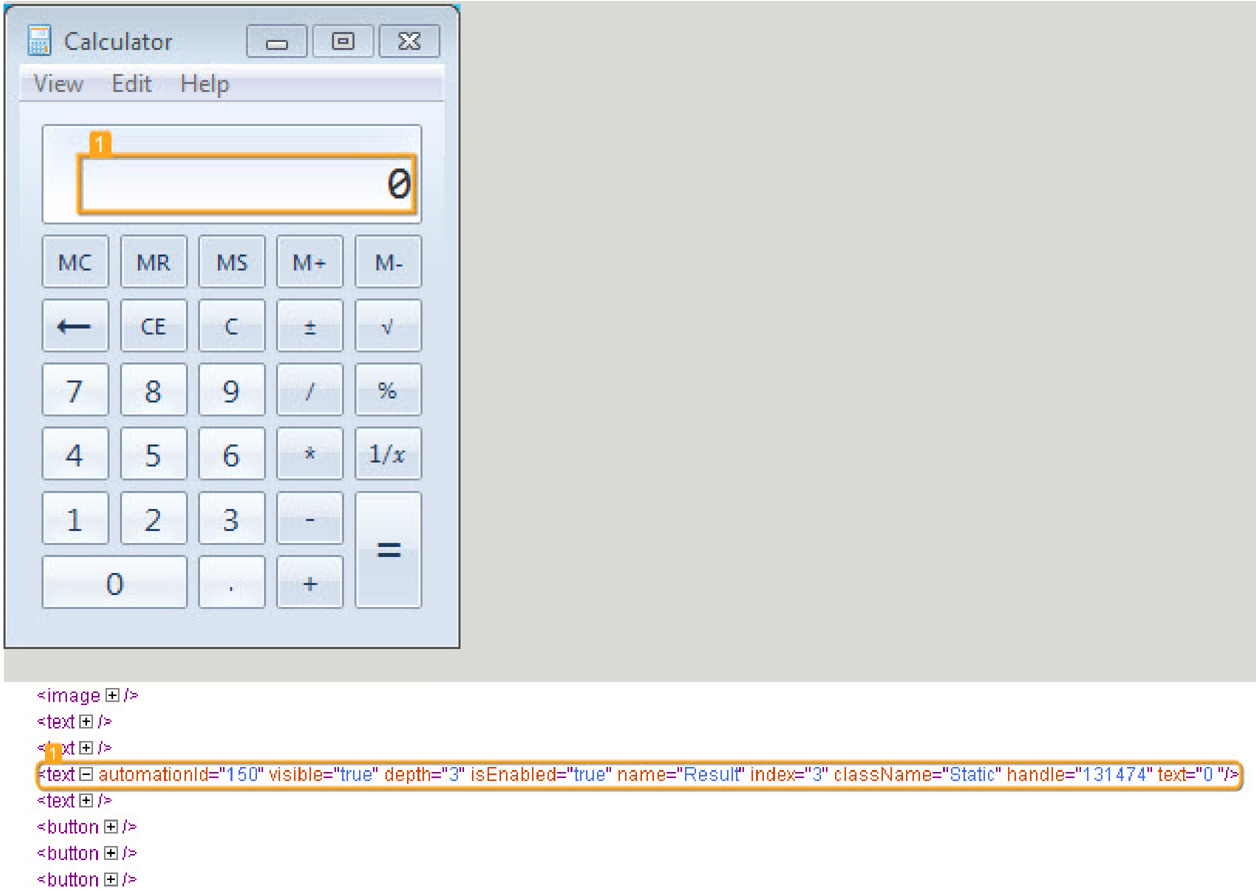
The Extract Tree as XML step can also contain the following set of action steps:

- Conditional step
- Group step
- Throw step
- Write Log step

Example: Selected tree and XML string

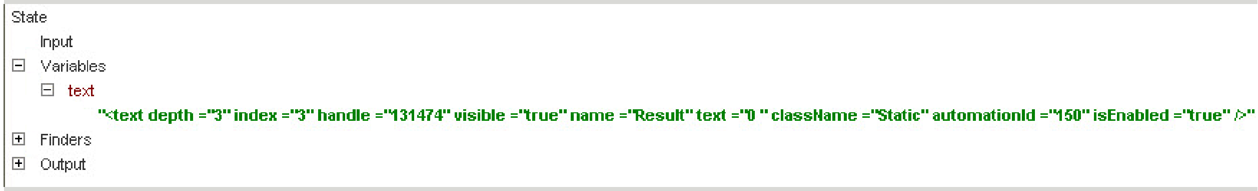
The following figures show a part of the Calculator widget tree exported to a variable as an XML string.

Note The order of exported attributes in a variable can be different from the application tree in the Recorder view.



```
<image ⊞ />
<text ⊞ />
<text ⊞ />
<text ⊟ automationId="150" visible="true" depth="3" isEnabled="true" name="Result" index="3" className="Static" handle="131474" text="0 "/>
<text ⊞ />
<button ⊞ />
<button ⊞ />
<button ⊞ />
```

The following is an XML string exported to a variable.



```
State
    Input
⊟   Variables
    ⊟  text
         "<text depth ="3" index ="3" handle ="131474" visible ="true" name ="Result" text ="0 " className ="Static" automationId ="150" isEnabled ="true" />"
⊞   Finders
⊞   Output
```

Extract Value

This step extracts values of different elements.

Properties

Component
Component finder of the step.

Extraction Type
Select the type of information you want to extract.

- Attribute: Extracts the value of the specified attribute.
- Derived Attribute: Extracts the value of the selected derived attribute. Specify the name of the attribute without prefix `der_`.
- Text: Extracts text from the immediate child element of the selected component. To extract text from all descendant elements, select **Include All Descendants**.

Evaluate Expression step
Optional. Contains a list of conversion functions suggested for use on a value. The function list suggested for a particular value is limited, depending on the value type. To specify a conversion function, in the step context menu, click **Evaluate Expression step**, and then do either of the following:

- Select the required function from the list of suggested functions.
- Click **Plain** to manually type the required function. For example, if you need to extract a text that represents an integer and store it in an Integer variable, you can use the expression `$initial.integer()`.

For more information on the supported functions and examples, see Expressions.

You can add one or more Evaluate Expression steps.

Store Current In step
Name of the variable of specified type to store the extracted/converted value. The type of the value must match that of the variable and can be one of the following: Integer, Boolean, Number, or Text.

You can add one or more Store Current In steps. For example, this can useful if you need to extract a full name of a person and store it in two variables, such as for the first name and last name respectively, inside the same step.

The Extract Value step can also contain the following set of action steps:

- Conditional step
- Group step
- Throw step
- Write Log step

Focus

This is an application-level step that brings the selected application to focus. You can use this step to, for example, bring to foreground an application that was started minimized. This step is available for applications on remote devices.

To insert a Focus step, right-click the application tab in the **Recorder View** and select **Application Action** > **Focus**. Kofax RPA automatically inserts necessary finders and guards to execute the step.

Freeze Tree

Freeze Tree step is a group step that freezes the application tree refresh in the Recorder View when executing steps in the workflow. While the steps within the Freeze Tree step execute, the application tree is not reloaded. Once the execution flow is outside the Freeze Tree group, the application tree is reloaded.

This step can help you greatly increase performance while executing cyclic operations on static windows, such as tables, spreadsheets, forms, and the like.

The Freeze Tree step is not supported for use in the built-in browser.

Group

This step combines several steps into a group. In the Group step, you can create local variables that are available only within a group. If you want your step to use a local variable, include the step into the group with the local variable. You can create steps in a group or use the cut and paste operations to include existing steps in a group.

Guarded Choice

The guarded choice step is used to set up a number of conditions, each associated with some actions. Whichever condition or guard is satisfied first, its associated actions or steps are executed. This is often used to ensure that an interface element, such as a button, is present before trying to move to and click it. To avoid waiting indefinitely, a timeout guard is added. In Kofax RPA you can use location and timeout guards to make sure the robot finds the required elements and works as designed.

In many cases Kofax RPA adds guards automatically when you insert a step, such as Click or Press steps. The following guards are available.

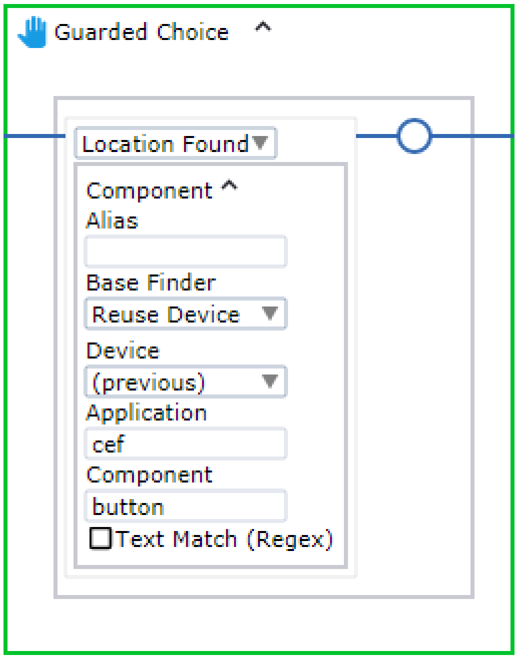
When seconds has passed
This is a timeout guard that waits a specified time before executing the next step in your robot.

> **Note** A default 60-second guard is inserted in the following steps when they are added via the **Recorder View**: Click, Scroll Mouse, Enter Text, Extract Value, Extract Contents, Extract Image, Extract Text From Image, and Press Key.

Application Found
Application guard that makes sure the application is found via a specified finder. If it does not find the application, it waits until it appears. If it finds more than one application, it waits until there is only one application.

Application Not Found
Application guard that makes sure the application is not found via a specified finder.

Location Found
Location guard that makes sure the element is found via a specified finder.



Location Found

Component

Alias

Base Finder: Reuse Device

Device: (previous)

Application: cef

Component: button

☐ Text Match (Regex)

Location Not Found
Location guard that makes sure the element is not found via a specified finder.

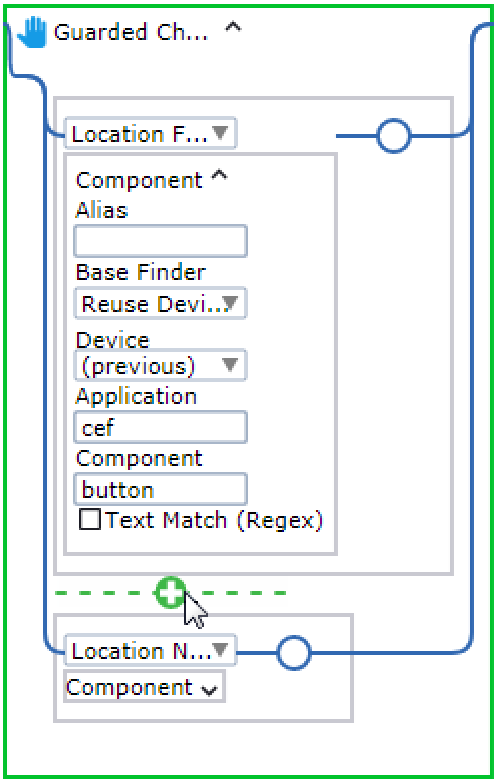
Location Removed
Finds an element via a specified finder and waits until the element is removed before executing the next step.

Tree Stop Changing
A timeout guard that waits a specified time after the last application or web page tree change before executing a step. The timeout is specified in milliseconds.

A Finder must be specified for each of the following location guards.

Add Guards Manually

To manually add a guard to a Guarded Choice step, hover your mouse over the bottom line of the guard frame until you see a green plus sign and click the plus to add a guard. See the following example.



Tips and Tricks

Open documents directly
You do not need to launch the application first and then open the document. Instead, just open the document and it will launch the associated program.

Watch out for accessibility detection

- When opening applications: Opening Adobe Reader using Desktop Automation shows the Accessibility Setup Assistance window. Usually this is a one time setup, but if you have desktops being spawned on demand, it may require your robot to handle it.
- When opening files: When a PDF file is opened, the Adobe Reader asks if the screen reader should process the document.

After pasting text into a field, use a guard for the next step

A guard verifies that the contents of the text field match what you pasted for the next action. Otherwise, the full value may not yet be in the field if you press Enter right after the paste operation.

Note This tip does not apply to password fields.

Initiate Session

Use this step to connect to devices over an RDP connection. See Use RDP Connection for details.

Note The Initiate Session step waits until the connection is established before continuing robot execution. If remote connection fails, an error message is provided.

Properties

Action
Select an action to perform by the step.

Host
Specify the host name to connect to.

Account
Specify an account name for RDP connection.

Domain
Specify a Domain name for RDP connection.

Password
Specify a password to authenticate on the remote desktop.

Note If the password or host name contains characters not accepted in an URL, such as a backslash, they must be encoded.

Desktop size
Set desktop geometry (WxH).

Color depth
Set connection color depth, such as 16, 32, or other.

Note Always use the same explicitly-specified resolution and color depth parameters for the RDP connection. Windows 10 does not support color depth lower than 32 bit. Therefore, the request from the RDP to change connection color depth will be ignored for this version of Windows.

Logon dialog dismissal timeout
Specify a number of seconds to wait before dismissing an extra screen during login. If the system you are connecting to is configured to show an extra screen when the user logs in, set a number of seconds to wait before dismissing the extra screen during login in this option. If this screen is not dismissed, the action may fail.

Keep-awake key press interval
Specify the number of seconds between the dummy keystrokes to keep the RDP session alive. Default is 30 seconds. To turn off keystrokes, specify zero (0).

KTA

This step connects to a Kofax TotalAgility (KTA) server and can start a new job, get a job status, create a new document, get a job variable, or raise a job event.

Important Set "Allow multiple logons" in Kofax TotalAgility to work with RPA.

Properties

Action
Select from the following actions depending on your task: **Create Job**, **Get Job Status**, **Create Document**, **Get Job Variable**, **Raise Job Event**.

Kofax TotalAgility Server
Specify the name of the Kofax TotalAgility installation to use. To add and configure a new Kofax TotalAgility installation, navigate to the **Admin** > **Settings** > **KTA Configurations** section in the Management Console. For more information, see KTA Configurations.

Actions

Create Job

This action starts a job in Kofax TotalAgility. Note that only released versions of Processes can be started from RPA. The step does not wait for the process to complete the execution, but assigns the resulting job ID from Kofax TotalAgility to a variable, so that the result of the execution can be monitored.

First, select the Kofax TotalAgility category that your process is in. As a result, a drop-down list of Kofax TotalAgility processes in that category is displayed. Then, select the process you want to start. The list of parameters belonging to that process is displayed. All parameters have a default value in Kofax TotalAgility. To override the default value in your robot, here you can select the required parameter and fill in your new value.

Get Job Status

With this action, the KTA step provides status information for a specific job. The job is identified by its job ID such as a status returned from the Create Job step. The status information consists of two fields: a number and a description (**Value** and **Formatted as Text** in the following table).

Value	Formatted as Text
0	Active
1	Completed
2	Terminated
3	Suspended
4	Pending Completion
5	Locked
6	Ready For Evaluation
7	On Hold
8	Awaiting Completion
9	Awaiting Case Completion
10	Awaiting Completion Terminated
11	Awaiting Case Completion Terminated

Create Document

This action creates a document in Kofax TotalAgility. The document ID is returned to the robot. This document ID can be passed to the Create KTA Job step as a variable of type Document. See the Kofax TotalAgility documentation for the list of document types that can be created.

The following parameters cannot be set from RPA:

- Checklist
- Data Backbone
- Dynamic Complex
- XML expression

Get Job Variable

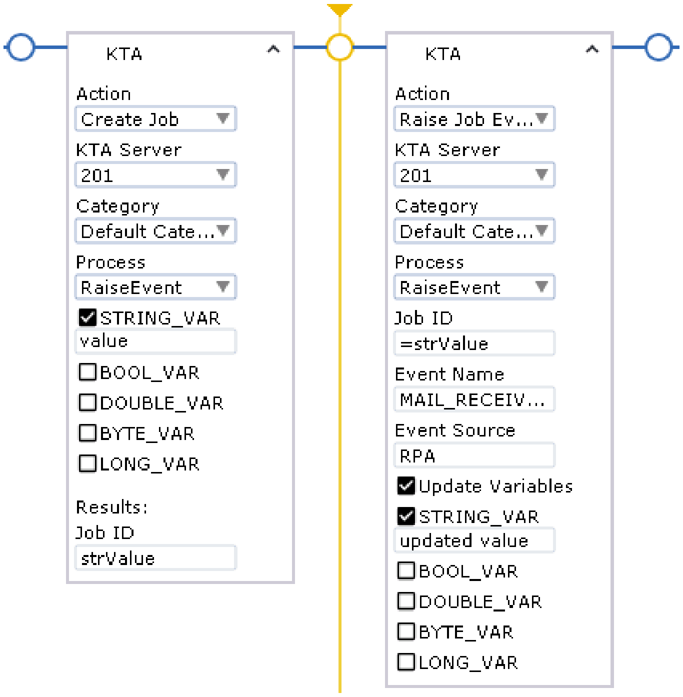
With this action, KTA step provides a value of a specific Kofax TotalAgility variable. To specify a Kofax TotalAgility variable to get a value for, fill in the following fields:

- **Job ID**: Type in the Kofax TotalAgility job ID
- **Variable ID**: Type in the Kofax TotalAgility variable ID

The returned information shown in **Results** consists of the **Value** field and the **Variable is Found** field, indicating if the variable is found in the job.

Raise Job Event

Use this action to send events to Kofax TotalAgility and modify Kofax TotalAgility variables.



KTA

Action: Create Job

KTA Server: 201

Category: Default Cate...

Process: RaiseEvent

☑ STRING_VAR: value

☐ BOOL_VAR

☐ DOUBLE_VAR

☐ BYTE_VAR

☐ LONG_VAR

Results:

Job ID: strValue

KTA

Action: Raise Job Ev...

KTA Server: 201

Category: Default Cate...

Process: RaiseEvent

Job ID: =strValue

Event Name: MAIL_RECEIV...

Event Source: RPA

☑ Update Variables

☑ STRING_VAR: updated value

☐ BOOL_VAR

☐ DOUBLE_VAR

☐ BYTE_VAR

☐ LONG_VAR

After the Kofax TotalAgility job is created, use the **Raise Job Event** action and fill in the following fields:

- **Job ID**: Type in the Kofax TotalAgility job ID

  You can type in the same value as used in the **Create Job** action.
- **Event Name**: Type in the Kofax TotalAgility event name

  > **Note** Event name is case insensitive.
- **Event Source**: Type in the Kofax TotalAgility event source

To modify a variable, first check the **Update Variables** option.

Then, check a variable from the list and modify its value.

Note that **Raise Job Event** action makes it possible to modify all the Kofax TotalAgility variables, not only initialization ones.

Document validation example

Follow these steps to validate a PDF document.

1. Add a **KTA** step to the Desktop Automation robot.
2. Add a **Read File** step and Load a PDF document into the variable `DocData`.

3. In the **KTA** step, add a **Create Document** action to save the document in the Kofax TotalAgility database. Fill in the following options. Note that other options depend on your Kofax TotalAgility server.
   - **Document Data**: Specify the `DocData` variable with a document to process.
   - **MIME Type**: Specify the format of the document: `application/pdf`.
   - **Document ID**: Specify the ID that Kofax TotalAgility assigned to the document.
4. In the **KTA** step, use the **Create Job** action to start a Kofax TotalAgility process that can validate your document.
   - **Process**: Specify the process name created in Kofax TotalAgility that can process your document.
   - **MYKTADOCUMENT**: The parameter of this Kofax TotalAgility process that passes the document ID.
   - **Job ID**: Specify the ID that Kofax TotalAgility assigned to your process.
5. Add a **Get Job Status** step to ensure that the job is successful. Fill in the following options.
   - **Job ID**: Specify the ID of the **Create Job** action in the KTA step.
   - **Value**: Specify a variable to contain the job status value.
   - **Formatted as Text**: Specify a variable to contain the status description.

When the **Get Job Status** step returns the "Ready For Evaluation" status, you can set the activity in Kofax TotalAgility to be completed manually or automatically. Once the activity is completed, the "Completed" status is returned after you run the Get Job Status step.

Loop steps

This section describes looping steps in Desktop Automation robots.

There are three loop steps: the Loop step, the While Loop step and the For Each Loop step.

The following is common for all loop steps:

1. All loop steps have an optional iteration variable.
2. You can break out of loop steps using the Break step.
3. You can skip to the next iteration using the Continue step.

Iteration variable

All loop steps has an optional iteration variable. This is an Integer variable that you can define in the step with the following characteristics.

- The variable's initial value is 0, that is, during the first iteration the variable is 0.
- It is increased by one at the end of each iteration of the loop.
- The variable is local to the loop step and it is not accessible outside the loop.
- The variable is read only, which means you cannot change it using the Assign step.

To refer to an iteration variable inside the loop, select **Iteration Variable** and enter the name of the variable to store the iteration.

Break

This step helps you break out of the Loop step. It must be used only inside the Loop step. You can use several Break steps inside one loop.

Continue

This step helps you skip to next iteration in the Loop step. It must be used only inside the Loop step.

For Each Loop

The For Each Loop step iterates over nodes in the application tree. It has a component finder called the Scope Finder, which defines a part of the tree to find the nodes to iterate over. Iteration never loops over nodes that are not below the scope node (the node found by the scope finder). The Scope Finder is similar to any finder in a step, because it finds a part of the tree that the step works on. The scope finder must always have a name and it must be unique inside the For Each Loop step.

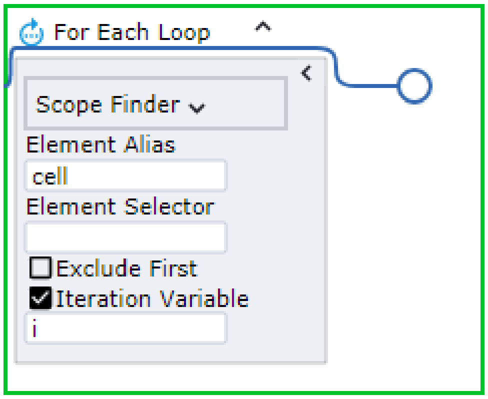
The element finder in the for Each Loop step consists of a name of the finder and a relative selector, such as "> DIV". Just like the Scope finder, the element finder must also always have a name and it must be unique inside the For Each Loop step.

The relative selector is used to find the elements to loop over. The selector is called relative because it is meant to be combined with another selector to form a new real selector. If the Scope finder has a selector `"DIV[class="someClass"]"` and the Element finders selector is `"> DIV"`, then the combined selector is `"DIV[class="someClass"] > DIV"`. The actual finder used to find the elements to iterate over is essentially the same as the Scope finder except that the selector is replaced with this new combined selector.

The For Each Loop works as follows. In the first iteration, the first element found by the combined finder is the one that will be bound to the element finders name. On the subsequent iteration, the found element is the next element found after the previous one. If the tree changed during the execution of the loop step and new element appears in the tree, then the new element may or may not be included in the future iteration depending on whether it appeared after or before the element of the current iteration.

The element found by the element finder can then be used inside the body of the loop as references in other finders.

This is an example of a For Each Loop step:



For Each Loop

Scope Finder

Element Alias

cell

Element Selector

☐ Exclude First

☑ Iteration Variable

i

Recorder View Loop Menu

The For Each Loop step may be inserted directly as any other step in the editor pane, but it is more convenient to insert it by right clicking inside the Recorder View and using the shortcut menu. Select a menu item called Loop, which has four submenus:

- All Sibling
- Each <tag name> Sibling
- Each <tag name> (<level>) Ancestor
- Each Table Row

These submenus help you create a For Each Loop step that will loop over various elements depending on the currently selected element. Design Studio also inserts a Guarded Choice step around the For Each Loop step, which ensures that the Scope finder is found in the tree before the loop step is executed.

All Sibling

This menu item inserts a For Each Loop step that loops over all siblings of the selected tag. The Scope finder finds the parent (called the scope node) of the selected element and the Element Selector is "> *". That is, it finds any child node under the scope node.

Each <tag name> Sibling

This menu item inserts a For Each Loop step that loops over all siblings of the selected tag with the same tag name. The Scope finder finds the parent (called the scope node) of the selected element and if the selected element has a tag name P, then the Element Selector is "> P".

Each <tag name> (<level>) Ancestor

This menu item is specific to using the built-in browser in Desktop Automation robots. The command searches for a good ancestor node to loop over. In doing so it either looks for an ancestor node that has more than one or more sibling nodes that are similar to itself or for an element with one of the following tag names: "TR", "LI", "TD", "TH", "DD", "OPTION", "PARAM". Two nodes are similar if:

- They have the same tag name and no class attribute.
- They have the same tag name and the same class attribute.

If you select one of the inner P tags in the following HTLM, the loop will not iterate over one P tag in its enclosing DIV tag, but it will loop over the DIV tags containing the P tag.

For example:

```
<DIV>
```

```
  <DIV>
    <P>1</P>
  </DIV>
  <DIV>
   <P>2</P>
  </DIV>
</DIV>
```

The level in the parenthesis of the menu item indicates how many levels above the selected element the actual loop element is located. In the above example this is 1. The tag name shown in the menu item is the tag name of the actual loop element.

Each Table Row
This menu item is specific to using the built-in browser in Desktop Automation robots. The command inserts a For Each Loop step that iterates over all the rows in a table.

Work with For Each Loop Step

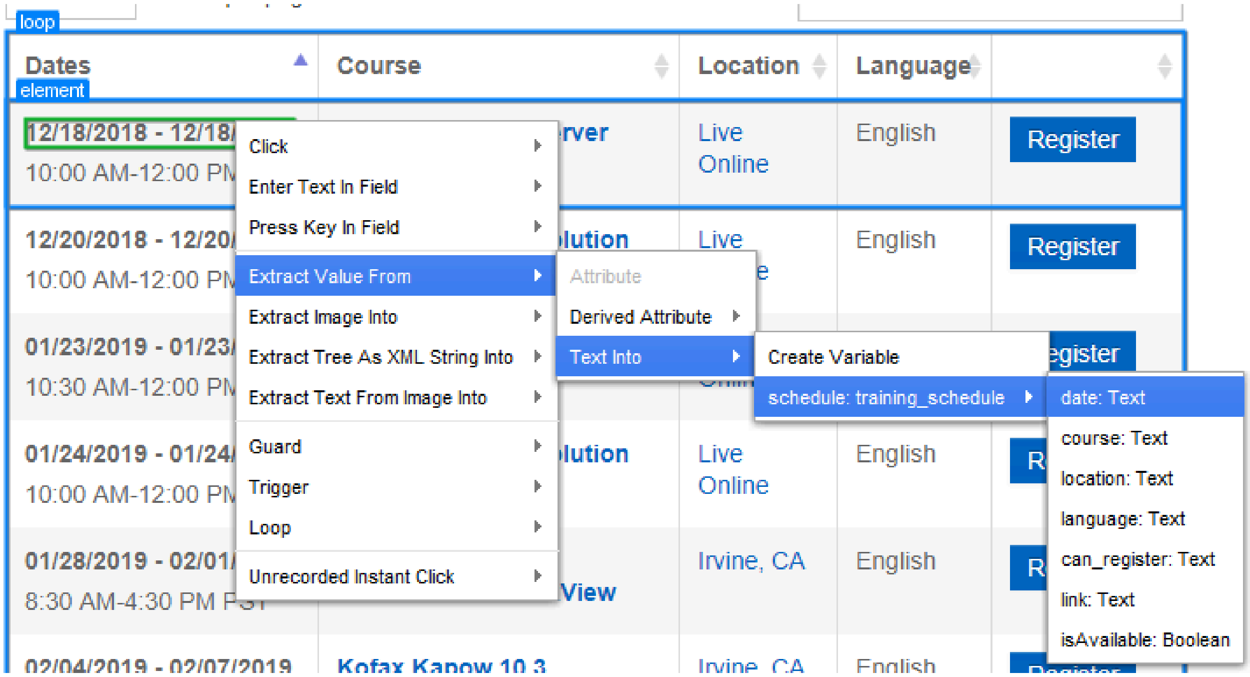
There are a number of things to consider when working with For Each Loop step.

Skip iterations
If you want to skip some element during looping, such as a few initial nodes or every second node, insert a Conditional step as the first step inside the loop that continues when the test is true. For example, with the following condition the loop skips all even iterations.

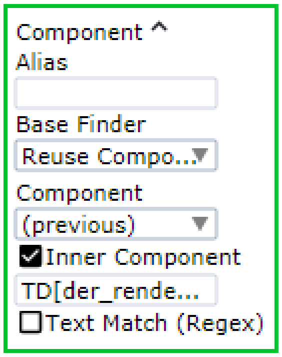
```
=i % 2 == 0
```

Finders in the loop step

Kofax RPA automatically finds elements relative to the found element. If you insert an action by right-clicking in the Recorder View and the element is inside a named found element, the generated finder is relative to the found element as shown below.



The result should look similar to the image below. The actual result depends on the element you select and the name you gave to the element finder.



Component ^

Alias

Base Finder

Reuse Compo... ▼

Component

(previous) ▼

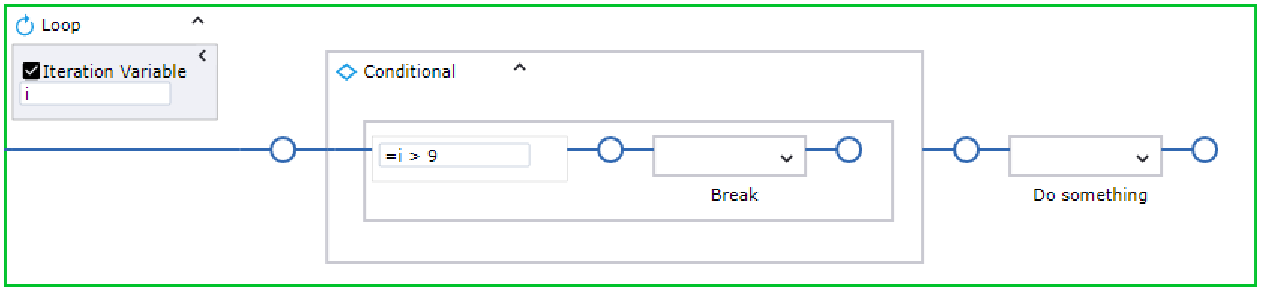
☑ Inner Component

TD[der_rende...

☐ Text Match (Regex)

Loop

A Loop step is a group of steps accompanied by a Break step to break out of the loop, or a Continue step to skip to next iteration. To loop with a condition, use a Conditional inside the loop. To loop by waiting for something to happen on a device, use a Guarded Choice instead.

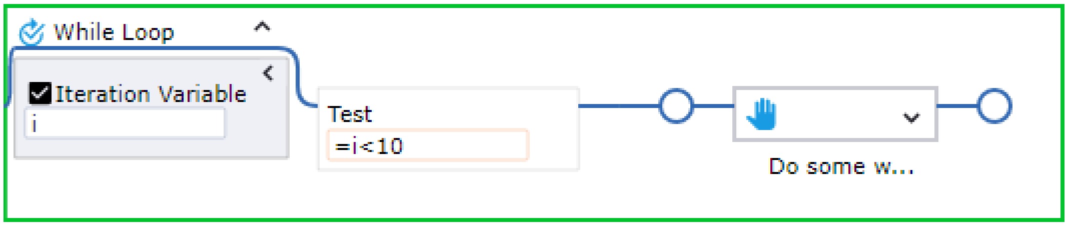


Press the **Go to next Iteration** button to execute until the same flow point is reached again. The loop can be executed more than once if the flow point is skipped in some iterations. If there are no more iterations, the execution stops at the flow point outside the Loop step.

While Loop

The While Loop step is similar to the Loop step with an additional property: Test. This is a test that is evaluated before each iteration of the step. If the test is true, the body of the step is executed. If the test is false, the execution breaks out of the loop.

The following is a While Loop step that executes the "Do some work" step 10 times and then breaks out of the loop when the test fails.



Move Mouse

This step moves the mouse to a specified location on the screen. When you add the Click from the Recorder View, the Move Mouse step is added automatically before the Click step. Mouse coordinates are relative to the top left corner of the window. X is the horizontal axis that goes from left to right and Y is the vertical axis that goes from top to bottom.

Additionally, it is possible to simulate hardware keyboard and mouse on an automated computer. For more information, see "Install the Desktop Automation Service" in the Kofax RPA Installation Guide.

Properties

Offset

- None: Does not use any coordinates offset and moves to the center of the selected element. It is equivalent to the following:

  Relative to set to **Center** with x=0, y=0
- Use: Specify the offset in pixels using the following options.

  Relative To

  This option specifies the starting point to calculate the offset.
  - Top Left: Top left corner of the window or the selected element with x=0 and y=0.
  - Top: Middle of the top border of the window or the selected element with y=0.
  - Top Right: Top right corner of the window or the selected element with y=0.
  - Left: Middle of the left border of the window or the selected element with x=0.
  - Center: Middle of the window or the selected element.
  - Right: Middle of the right border of the window or the selected element.
  - Bottom Left: Bottom left corner of the window or the selected element with x=0.
  - Bottom: Middle of the bottom border of the window or the selected element.
  - Bottom Right: Bottom right corner of the window or the selected element.

X

Specifies a horizontal offset relative to the selected starting point. Positive numbers move the mouse to the right of the starting point. Negative numbers move the mouse to left of the starting point.

Y

Specifies a vertical offset relative to the selected starting point. Positive numbers move the mouse down from the starting point. Negative numbers move the mouse upward from the starting point.

Notify

The step displays a message in the Automated Device taskbar notification area. This step is designed for using with the Trigger Choice in the Attended Automation robots. When a trigger event is intercepted by the robot and the robot prevents the user from using the keyboard and mouse while running, the step notifies the user what is going on while the robot is executing. The step has the following parameters.

Properties

- Device: Device name.
- Title: Title of notification message.
- Message: Notification message.
- Icon: Select from None, Information, Warning, and Error.

Open

Opens an application on the Automation Device or locally. For example, a headless terminal is opened on the local device if its driver is enabled on the local device.

Properties

Device
Select the device where you want to open an application.

URI

- Specify the path to the application or a website to open. Use forward slashes in the path. For example:
  - `C:/Program Files/SAP/FrontEnd/SAPgui/saplogon.exe`
  - `="C:/Program Files/SAP/FrontEnd/SAPgui/saplogon.exe"`
  - `https://www.google.com`
  - `about://version`

    See Access Websites for more information.
- For the built-in Windows applications, you can specify the process name, such as `calc.exe`.
- To open built-in Excel driver, specify the following:

  `excel://new` to create a new spreadsheet

  `excel://<full path to spreadsheet>/<spreadsheet name>.xlsx` to open an existing spreadsheet

  For more information, see Built-in Excel Driver.
- For the RDP connection, specify the following:

  ```
  rdp://<domain>\<username>:<password>@<hostname>?
  <param1>=<value1>&<param2>=<value2>
  ```

  Where available parameters are:
  - d: domain (or type domain as part of username in URL)
  - g: desktop geometry (WxH)
  - a: connection color depth
  - Z: specify a number of seconds to wait before dismissing an extra screen during login (see below)
  - kapow-keep-awake: Specify the number of seconds between the dummy keystrokes to keep the RDP session alive. Default is 30 seconds. To turn off keystrokes, specify zero (0), such as `kapow-keep-awake=0`.

  For example: `rdp://admin:AdminPassword@Server1`

  > **Note** If the password or host name contains characters not accepted in an URL, such as a backslash, they must be encoded.

  If the system you are connecting to is configured to show an extra screen when the user logs in, set a number of seconds to wait before dismissing the extra screen during login in the Z parameter, such as `rdp://admin:AdminPassword@Server1?Z=3`. If this screen is not dismissed, the action may fail.

  > **Note**
  > - The Open step with an RDP connection waits until the connection is established before continuing robot execution. If remote connection fails, an error message is provided.
  > - Always use the same explicitly-specified resolution and color depth parameters for the RDP connection. Windows 8 and 10 do not support color depth lower than 32 bit. Therefore, the request from the RDP to change connection color depth will be ignored for these versions of Windows.

Also use the Open step to work with cookies in the Desktop Automation built-in browser. See Cookie Management in the Desktop Automation built-in browser for more details.

PDF

The PDF step helps you extract content from a PDF document.

Note The PDF extract feature is not supported on CentOS/Red Hat Enterprise Linux 7.x operating systems.

The **Recorder View** shows a single page of the PDF document tree and the extracted text. The robot can navigate through the document using the **Next Page**, **Previous Page** and **Goto Page** actions available on the **Application Action** menu. The menu is available when you right-click the application tab in the **Recorder View**.

Text extraction results depend on the internal data and structure of the PDF document. The text is split based on the formatting in the PDF document and the underlying accessibility of data and might include text outside the page boundaries or hidden by overlapping elements. If the required accessibility data is missing from (usually older) PDF documents, it might be necessary to use the Extract Text From Image step to extract the text using OCR.

The **Extract text** application action and the **Extract text** component action can be used to extract structured text from a specific area of the page.

Properties

Action
Select an action to perform using the PDF.

Document Source

- **Local File**: specify the path to the file in the local file system in the **File path** field.
- **Robot File System**: Specify the path to the file in the robot file system in the **File path** field.
- **Binary**: Specify a variable or expression containing a PDF document in binary form.

Page number
Optionally specify the physical page to show after opening the document. If this property is not specified, the first page is shown.

Component actions

Action	Description
Extract text	Extracts text from the selected element of the PDF document.

Press Key

This action presses a specified key. This is an application-level step that is available when you right-click the following:

- The application tab.
- A text field in an application or a web site.
- A program point in the Desktop Automation robot.

Using the virtual input driver

When you enable the virtual input driver on the automated device (see Activate the virtual input driver in Configure Desktop Automation Service), Press Key steps in Device Automation on Windows automatically use this driver for entering text. The keys are entered as though via a hardware keyboard, enabling system-only combinations, such as Ctrl+Alt+Del to work. When using calculated keys, any flags other than "u" (for a key-up event) are not supported. The driver does not support holding down more than six keys at the same time.

Properties

Name
Name of the step.

Finder
Device: Select the name of the automation device.

Application: Specify the name of the application the action is performed in.

Key
Select **Standard Keys** or **Calculated Key**.

- **Standard Keys**: Select from the standard keyboard keys, such as letters, numbers, punctuation marks, arrow keys, function keys, and more.
- **Calculated Key**: Select this option if the provided options for keyboard keys are insufficient. In the **Key Code** field, specify a virtual-key code or a space-separated list of input specifications. This functionality is only supported on the Windows operating system.

  A virtual-key code is a symbolic constant name, such as VK_LBUTTON for "left mouse button." For the list of virtual-key codes, see the Microsoft documentation.

  An input specification is a sequence of one or more keydown or keyup events. When adding an input specification, specify a virtual-key code or scan code, using a respective prefix:

  - **v** for a virtual-key code, such as v0xXX.
  - **s** for a scan code, such as s0xXX.

  By default, an input specification is a keydown, virtual-key event. To override this default, add an **f** flag to an input specification and separate them by a comma. The following flags are supported: **u** for keyup; **s** for scan code; **e** for extended key; **U** for Unicode.

  Examples

  - The `v0x30 v0x30,fu` calculated key presses the zero key and then releases it. The `v0x30` input specification is a keydown event, while `v0x30,fu` is a keyup event.
  - The `v0x5b v0x52 v0x52,fu v0x5b,fu` calculated key is for the Run command (Win+R): It presses the left Win key, then the R key, and then releases both keys. The `v0x5b` and `v0x52` are keydown events, while `v0x52,fu` and `v0x5b,fu` are keyup events.
  - The `s0x04c1,fU s0x04c1,fUu` calculated key is for the Cyrillic character Ӂ (Zhe with breve). While the `0x04c1` code is the Unicode for Ӂ, `s0x04c1,fU` is a scan code, keydown event and `s0x04c1,fUu` is a scan code, keyup event.

Modifier
If you selected Calculated Key in the Key property, the Modifier property is ignored, and you do not need to configure it.

Select a key modifier:

- **Fixed Key Modifier**: Contains three standard key modifiers, such as Shift, Ctrl, Alt.
- **Calculated Key Modifier**: When this option is selected, specify a symbolic constant name of the virtual-key code for a modifier.

  In the text box that appears, you can enter the key codes for Shift, Ctrl, and Alt only. For example, the VK_LSHIFT key code stands for the left Shift key, VK_RCONTROL stands for the right Ctrl key, and VK_MENU stands for the Alt key. For a complete list of key codes, see the Microsoft documentation.

Count
Specify how many times to perform the action. The format is an equal sign and a number, such as `=1`.

Read File

The Read File Step extracts data from the file on a local or remote desktop into a binary variable. This step can be used to read from a file on the Robot File System.

Note To enable file system access, including access to the local file system in the Local Desktop Automation mode, select the **Allow File System and Command Line Access** option on the **Security** tab in the RoboServer settings window.

If your RoboServer is configured to not allow local file system access, you cannot use this step with **File Access** set to **Direct Access** on the local device. However, you can use it on the local device with **File Access** set to **Via RFS**. On a remove device, both options, "Direct Access" and "Via RFS", can be used at all times.

Properties

- **Device**: Select the reference name to use. This reference name is specified in the Required Devices property of the Call Desktop Automation Robot step.
- **File Access**: Specify how the file must be accessed.
  - To read from a file on the specified local or remote device, select **Direct Access**.
  - To read from a file on a robot file system, select **Via RFS**.
- **File Name**: Specify the path to the file from which to extract the data.
- **Variable**: Specify the binary variable in which to store the extracted data.

Remote Device Action

The Device Action step can perform some actions with the Desktop Automation Service running on a remote computer.

Properties

Name
Name of the step.

Device
Select the remote device to manage the service on.

Action

- **Suspend**: Suspends the device. To restore the service operation, a user or an administrator needs to manually start the Desktop Automation Service on the device.

  The suspended state makes the DA service unavailable for robots to use, but the state information is send to the Management Console via the ping mechanism and the device is displayed in the **Admin** > **Devices** section. This command is useful if for some reason the service or the computer running it needs some configuration changes.
- **Shutdown**: Stops the service, which makes the remote device unavailable. The computer running the Desktop Automation Service is removed from the list in the Management Console.
- **Restart**: Stops and starts the service. A robot or Design Studio loses the connection to the device and must be reloaded to restore it.
- **Lock Screen**: Locks the screen on the remote device. This action requires a password as a parameter. See Use Lock Screen for more information.
- **Restart Machine**: Restarts a computer running the Desktop Automation Service.
- **Shutdown Machine**: Shuts down a computer running the Desktop Automation Service.

Use Lock Screen

In some cases it is necessary to lock computer screens when working with automation devices. You can lock a screen by using the **Lock Screen** command on the Desktop Automation Service menu. Before locking your device screen, make sure the service is running and it is in the connected state. To lock a screen, right-click the Desktop Automation Service icon and select **Lock Screen**.

If Windows is configured to show an extra screen when the user logs in, Lock Screen tries to detect this extra screen and dismiss it by pressing OK. If this screen is not dismissed, the action may fail. When an extra screen is detected, the Lock Screen feature dismisses it three seconds after the connection with the system is established. If automatic detection fails or three seconds do not suffice, add a KAPOW_LEGALNOTICE_SECONDS system variable in the environment variables on your automated device, and set the number of seconds in the *Variable value* field to wait before dismissing the window after the connection. Restart the Desktop Automation Service after adding the variable.

Lock Screen Usage Prerequisites
To use the Lock Screen feature with Desktop Automation, your device must meet the following requirements.

- Remote Desktop connection must be enabled.
- The user under which the Desktop Automation Service runs must be allowed to connect via Remote Desktop (as a member of the Admin group or the Remote Desktop group) and use a password.
- The effective group policy of **Computer Configuration** > **Administrative Templates** > **Windows Components** > **Remote Desktop Services** > **Remote Desktop Session Host** > **Security** *"Always prompt for password upon connection"* must be off.
- Port 3389 must be open.
- The Automation Device cannot be a domain controller.

Return

This is a final step in robot execution that outputs variable values. Specify variables in the Return step in the same order as the types in the **Output** section of the automation workflow. This step is mandatory in the robot. Leave the step empty if you do not want to output any variable value.

You can use more than one Return step, but once the first Return step is executed, robot execution stops. This might be helpful in conditional steps when you check the condition and output variable values if they comply with the condition. If the condition is not met, the robot continues executing.

Properties

Name
Contains the name of the step.

Values
Specify variables with values you want to output. Note that the order of variables must match the list of types in the **Output** section. If all variables are the same type, the order is not important.

Scroll

This step helps you scroll in the program windows. This is an application-level step and it is available by right-clicking the application tab. Note that you must select the appropriate element on the window before inserting this step.

Additionally, it is possible to simulate hardware keyboard and mouse on an automated computer. For more information, see "Install the Desktop Automation Service" in the Kofax RPA Installation Guide.

Properties

Name
Specify the name of the step.

Amount
Specify the amount to scroll. The value in this field equals the number of notches of the mouse scroll wheel. Initially one notch equals 3 lines of text in text editors. You can change this parameter in the Mouse Properties window on the Automation Device. You can use both positive and negative numbers. For example, if you select **Down** and use a negative number, the element scrolls up.

When you use this step with the built-in browser, Amount option means a number of pixels to scroll.

> **Note** Make sure you correctly select an element to scroll. If an element cannot be selected, try clicking the element first and then use Scroll.

Set Clipboard

This step assigns a value to the clipboard of the Automation Device.

Properties

Name
Name of the step.

Device
Select the name of the automation device.

Contents
Specify a value to copy to clipboard. You can specify a variable name in this field.

Terminal

Use this category to connect to and automate a terminal. See Automate Terminals and Basic Terminal Tutorial for details.

Properties

Emulator
Select the emulator for your terminal. Kofax RPA supports connection to and interaction with 3270, 5250, 6530, and stream-based (vt100 and ANSI) terminals.

Action
Select an action to perform by the step. It can be either **Connect** or **Connect (SSH)**.

Host
Specify the terminal name or IP address and the port number if necessary. For example, `TerminalServer:25`

Credentials
If you select **Connect (SSH)** in the **Action** list, you can specify a user name and password to connect to the terminal in this property.

Options
Select to specify any connection options for the selected terminal, such as set color support, create a trace file, set a number of buffered lines, and so forth. See Automate Terminals for details.

Throw

This step throws an exception to indicate an error and handle it at another place in the Desktop Automation workflow.

When other workflow steps encounter errors, they also throw exceptions. Errors discovered by logic in the workflow and errors discovered by steps are handled in the same way. See Try-Catch for the list of exceptions thrown by other workflow steps.

Whichever way an exception is thrown, it is caught and handled by the closest Try-Catch containing the specified exception in a Catch branch. If there is no such Try-Catch Step, the exception is set to be "not handled" within the workflow. In that case, execution of the workflow as well the Call Desktop Automation

Robot step stops, and the error is handled as specified on the **Error Handling** tab of the Call Desktop Automation Robot step.

A typical use of the Throw step is in conjunction with timeout guards. A timeout occurs when an intended interaction with the Device (for example, set by a Location Found guard) is not possible. In some cases when a timeout occurs, it is possible to do something else and thus recover. When recovering is not possible, use the Throw step to communicate the failure in a structured way. This makes it possible to add a Try-Catch to properly handle an error (for example, by backing out of the interaction with the Device).

Using the same exception name for similar errors in different places in the workflow (that is, in different Throw steps) makes it possible to handle all errors in the same Try-Catch step. Therefore, the exception name should provide a classification of the error situation, not all the details.

This step throws an exception and robot execution stops. This step is helpful when designing and debugging your robot. For example, if you want to know when a 60 second timeout guard waits for 60 seconds without any action, insert the Throw step into a timeout guard with a text similar to "60 seconds timeout has passed." If you see your message during the execution, it means the guard waited for 60 seconds and nothing happened.

The Throw step cannot be inserted in the Finally block of the Try-Catch step.

Properties

Name
Contains the name of the step.

Exception
Name of the exception. This name must adhere to the variable name rules. See Naming Policy.

Trigger Choice

This step is a part of the Attended Automation functionality. This step helps you select a trigger and insert action steps launched by the trigger. Insert one or more action steps that are executed when the trigger event is intercepted by the Trigger Choice step.

When a triggered event is detected, the robot takes over the automated device and might prevent the user from using the mouse and keyboard. To inform the user of the action performed by the robot, use the Notify.

Trigger tips and tricks

- You can insert the Trigger Choice step by right-clicking the Recorder view tab and selecting **Trigger** on the menu.
- You can insert the **Component Click** event of the Trigger Choice step by right-clicking an element in the Recorder view and selecting **Trigger** on the menu.
- You can use only one trigger in a robot.
- A trigger cannot be used in Snippets.
- You cannot create triggers inside triggers.

The following trigger events are available.

Trigger

Application Opened
The robot executes selected actions when a specified application opens. Specify the trigger name and the application that triggers the action.

Application Closed
The robot executes selected actions when a specified application closes. Specify the trigger name and the application that triggers the action.

Component Clicked
The robot executes selected actions when a specified component is clicked. Specify the trigger name, the application that triggers the action, and the component within the application. Also, select a mouse button that clicks the component.

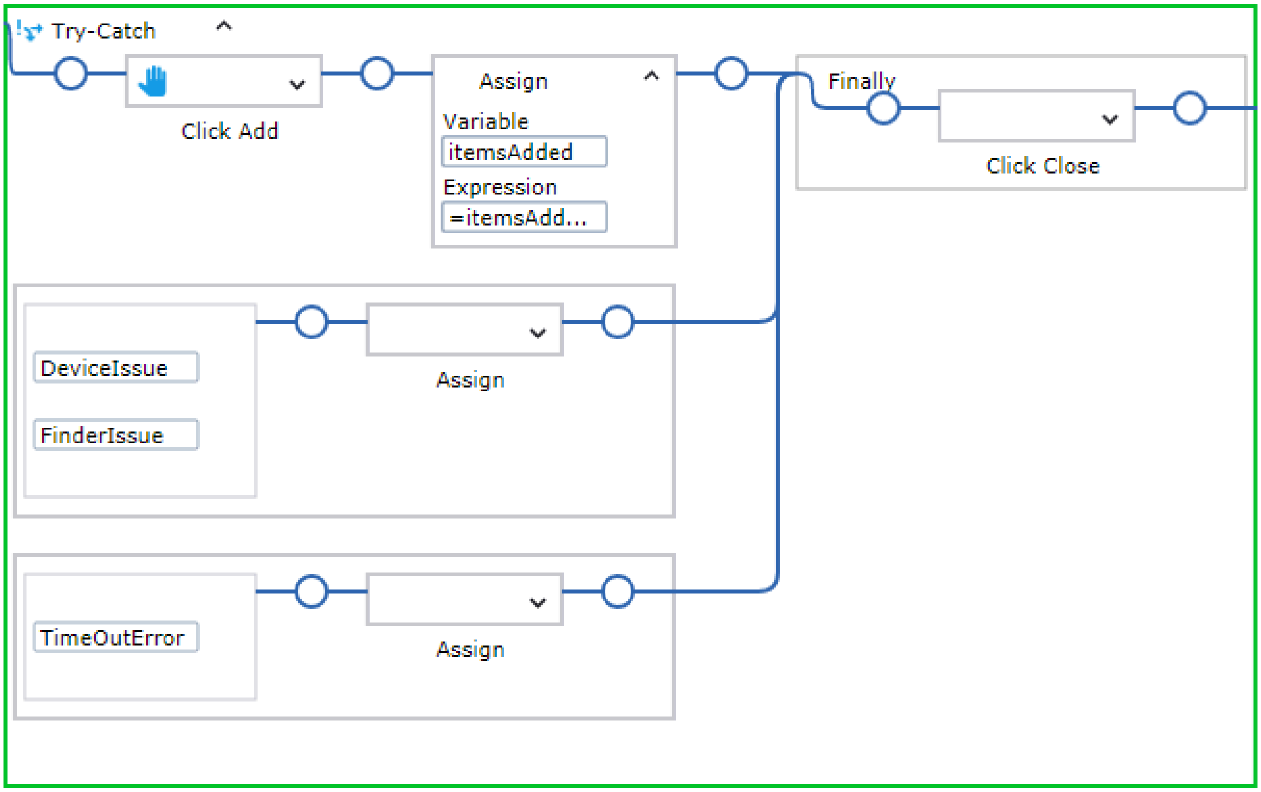
Hot Key Pressed
The robot executes selected actions when a specified key or key combination is pressed. Specify the trigger Name, and select the Key from the list. Also, select from the three standard key modifiers, such as Shift, Ctrl, Alt.

Try-Catch

This step helps you perform an action and catch one or more exceptions that might result from the action. The step consists of a number of branches divided into three parts.

- Try branch: The topmost part that specifies an action to perform.
- Catch branches: Specifies one or more exceptions that may be thrown when the action in the Try branch is executed; and what action to perform if that happens. You can have more than one Catch branch and each can list any number of exceptions that are handled in the same way.
- Finally branch: Specifies an action to perform. This branch is always executed last regardless of the Try and Catch execution results.

Exceptions may be thrown either explicitly by means of the Throw, or because other steps encounter errors during execution. The exceptions thrown are called Predefined Exceptions.



Properties

Name
Contains the name of the step.

Try
Specify an action to perform. If an exception is expected as a result of an action, specify the exception in the Catch block.

Exceptions
Specify one or more exceptions you expect to catch.

Each Catch branch consists of a list of exceptions and, to the right of that, the action to perform if execution of the Try branch throws one of these exceptions.

Each exception is given a name, corresponding to the exception name used in a Throw step or a predefined exception name.

When an exception is added or edited in a Catch branch, the editor proposes those exceptions that may be thrown inside the Try branch and which are not yet listed in a Catch branch.

Finally
Specifies the action to perform just before leaving the Try-Catch step.

Execution

Execution of the Try-Catch step can be a bit more complex than other steps. The most common execution cases are the simplest and explained first. The most complex cases appear when the Finally branch contains steps (is not empty).

In all cases, execution of the Try-Catch step begins by executing the Try branch. This can end normally, or by an exception thrown by one of its steps.

Most common cases: Finally branch is empty

Try branch ends normally

Execution proceeds with the step after the entire Try-Catch step. That is, the Catch branches are not executed in this case.

Try branch ends with an exception thrown

From the step that throws an exception, execution proceeds directly to the beginning of the Catch branch that lists the exception.

More complex cases: Finally branch is empty

Try branch ends with an exception thrown, but no Catch branch lists that exception

This case is treated as if the Try-Catch step itself throws the exception, which is handled the same way as when any other step throws an exception. All cases listed here apply.

> **Note** This strategy ("treated as if the Try-Catch step itself throws an exception") is used in many other cases.

If all Try-Catch steps have empty Finally branches, the workflow logic searches for a matching Catch branch in the surrounding Try-Catch steps that contain Try branches with this Try-Catch step. If such a Catch branch cannot be found in any surrounding Try-Catch step, the exception is set to "not handled" within the workflow. In such a case, execution of the workflow as well as the containing Call Desktop Automation Robot step stops, and the error is handled as specified on the **Error Handling** tab of the Call Desktop Automation Robot step.

If any Try-Catch step also has a Finally branch, the execution is similar, but with executing one "throw" at a time.

Try branch ends by an exception thrown and the appropriate Catch branch does the same

The exception thrown in the Catch branch is not handled by the Catch branches in the same Try-Catch step. Instead, this is treated as if the Try-Catch step itself throws that exception. The details are as described in the previous case.

A note on nested Try-Catch steps

An exception that is handled by a Try-Catch step is not handled by a surrounding Try-Catch step. Once the Catch branch that can handle the exception is found, the exception is considered fully handled and is "forgotten." Execution of the Catch branch starts and proceeds in the normal way. Thus, each exception is handled only once.

Most complex cases: Finally branch is not empty
In these cases, the steps in the Finally branch are executed just before execution leaves the Try-Catch step, no matter how the execution goes. The following cases show how this works out in detail for each of the cases discussed above.

Try branch ends normally
Execution proceeds with the steps in the Finally branch. What happens afterwards depends on how execution of the Finally branch ends.

- If execution of the Finally branch ends normally, execution proceeds with the step after the entire Try-Catch step.
- If an exception is thrown during execution of the Finally branch, it is treated as if the Try-Catch step itself throws that exception.

Try branch ends with an exception thrown and the Catch branch ends normally
After executing of the Catch branch, the logic is exactly as in the previous case.

Try branch ends with an exception thrown, but no Catch branch lists that exception
In this case the exception is "remembered" and execution proceeds with the steps in the Finally branch. What happens afterwards depends on how execution of the Finally branch ends.

- If execution of the Finally branch ends normally, execution proceeds as if the Try-Catch step itself throws the "remembered" exception again.
- If an exception is thrown during the execution of the Finally branch, it is not handled by the Catch branches in the same Try-Catch step. Instead, this is treated as if the Try-Catch step itself throws that exception (that is, the exception that was thrown by the Finally branch). The "remembered" exception is effectively "forgotten" at this point.

Try branch ends with an exception thrown and the appropriate Catch branch does the same
This is handled as in the previous case, except that the "remembered" exception is the one thrown by the Catch branch rather than the one thrown by the Try branch. As shown above, the exception thrown by the Try branch is fully handled and "forgotten" at the moment when execution of the Catch branch begins.

Predefined Exceptions

When a step encounters an error during execution, it throws one of the following exceptions. These exceptions can also be thrown explicitly by Throw steps if needed.

When thrown because of step errors, the predefined exceptions include a message explaining the issue. This message is made available if the exception is not handled by a Try-Catch step in the workflow, but instead terminates the execution of the Call Desktop Automation Robot step.

All "internal" exceptions are predefined, and their names cannot be changed. The name of a "User-defined" exception can be changed by the user, depending on what step action the timeout refers to. For example, it can be changed to InputNameTimeOut or LoginTimeOut.

- `TimeOutError`: Thrown if the execution times out
- `FinderIssue`: Thrown if a finder fails to find an element
- `DeviceIssue`: Thrown if a problem on a device or a driver that prevents the execution of a step
- `IncorrectValueIssue`: Thrown if the value of an expression is not suitable where it is used, such as `-1` in `"one".substring(-1)`
- `ExtractIssue`: Thrown if the Extract step fails to extract anything

- `DivisionByZeroIssue`: Thrown if a division by zero (or modulo by zero) occurs during the evaluation of an expression
- `OverFlowIssue`: Thrown if an overflow occurs in the evaluation of an expression
- `ConversionIssue`: Thrown if during the evaluation of an expression a conversion from one type to another fails, such as `"one".integer()`

Whenever an expression is a part of a step, execution of the step can throw the following exceptions:

- `IncorrectValueIssue`
- `DivisionByZeroIssue`
- `OverflowIssue`
- `ConversionIssue`

The following table lists exceptions that can be thrown by steps, finders and other workflow elements. **Expression issues** are any of the issues thrown by the steps with expressions.

Workflow Elements	Exception
Steps	
Click	DeviceIssue, FinderIssue, Expression issues
Enter Text	DeviceIssue, FinderIssue, Expression issues
Press Key	DeviceIssue, FinderIssue, Expression issues
Scroll	DeviceIssue, FinderIssue, Expression issues
Move Mouse	DeviceIssue, FinderIssue, Expression issues
Set Clipboard	DeviceIssue, Expression issues
Assign	Expression issues
Extract Value	DeviceIssue, FinderIssue, Expression issues, ExtractIssue
Extract Clipboard	DeviceIssue
Extract Image	DeviceIssue, FinderIssue, Expression issues, ExtractIssue
Extract Tree As XML	DeviceIssue, FinderIssue, Expression issues, ExtractIssue
Extract Text From Image	DeviceIssue, FinderIssue, Expression issues
Loop	None
Conditional	Expression issues
Group	None
With	DeviceIssue, FinderIssue, Expression issues
Guarded Choice	Depends on the guards as listed in the table below
Try-Catch	None
Break	None
Throw	None

Workflow Elements	Exception
Return	Expression issues
Open	DeviceIssue, Expression issues
Connect To Device	DeviceIssue, Expression issues
Remote Device Action / Lock Screen command	DeviceIssue, Expression issues
Remote Device Action / other	DeviceIssue
Expressions	
Any expression	IncorrectValueIssue, DivisionByZeroIssue, OverFlowIssue, ConversionIssue
Guards	
When seconds have passed	Expression issues, IncorrectValueIssue
Application Found	Expression issues, DeviceIssue, FinderIssue
Application Not Found	Expression issues, DeviceIssue, FinderIssue
Location Found	Expression issues, DeviceIssue, FinderIssue
Location Not Found	Expression issues, DeviceIssue, FinderIssue
Location Removed	Expression issues, DeviceIssue, FinderIssue
Stop Tree Changing	Expression issues, IncorrectValueIssue, DeviceIssue, FinderIssue
Finders	
Device Finder	DeviceIssue
Application Finder	DeviceIssue, FinderIssue, Expression issues
Component Finder	DeviceIssue, FinderIssue, Expression issues

The Guarded Choice step may throw the following exceptions, depending on the guards used in the step:

Guard	Exception
When seconds have passed	Expression issues
any other	DeviceIssue, FinderIssue, Expression issues

Unrecorded Instant Click

You can perform an instant mouse click on an element without recording it in the workflow. To perform this action, right-click an element in the **Recorder View**, click **Unrecorded Instant Click**, and select the mouse click type to use.

Windows

This step helps you work with a Windows desktop and run Windows applications.

Properties

Specify the following properties to run an application.

Device
Select the device where you want to open an application.

Action
Select the action to perform.

Executable

- Specify the path to the application to execute. You can use local drive path or network path.
- For the built-in Windows applications, you can specify the process name, such as `calc`.

  The following example provides several ways to start the Calculator application.

  - `calc`
  - `calc.exe`
  - `C:\\Windows\\System32\\calc.exe`
  - `C:\Windows\System32\calc`
  - `C:/Windows/System32/calc`
  - `C://Windows//System32//calc.exe`

  Specify the network path as follows:

  `\\MyServer\shared\reportform.exe`

Working Directory (optional)
Specify the path to the application's working directory. For example, you can use this path if you have applications with the same name located in different folders.

Arguments (optional)
Specify any arguments for the application if necessary. For example, if you specify "kofax" for a web browser application, the browser opens the https://www.kofax.com web site.

Step example

The following example shows Windows step properties that help you open the `text.txt` file located in `C:\Temp` using the Notepad application.

- **Action**: `Execute`
- **Executable**: `notepad.exe`
- **Working Directory**: `C:\Temp`
- **Arguments**: `text.txt`

Component actions

When working with Windows applications you can use the following actions on the **Component Action** menu.

Action	Description
Set Keyboard Focus	Sets keyboard focus on the selected component to use keyboard input.

Write File

The Write File Step writes data from a binary variable to files on a local or remote desktop. You can use this step to write to a file on the Robot File System (for an example, see the *Kofax RPA Administrator's Guide*).

> **Note** To enable file system access, including access to the local file system in the Local Desktop Automation mode, select the **Allow File System and Command Line Access** option on the **Security** tab in the **RoboServer Settings** application.
>
> If your RoboServer is configured to not allow local file system access, you cannot use this step with **File Access** set to **Direct Access** on the local device. However, it can be used on the local device with **File Access** set to **Via RFS**. On a remove device, both options, "Direct Access" and "Via RFS", can be used at all times.

Properties

- **Device**: Select the reference name to use. This reference name is specified in the Required Devices property of the Call Desktop Automation Robot step.
- **Contents**: Specify a value to write to a file. You can specify a variable name in this field.

  For example, when you write to a file on a robot file system, the field can contain the value `="data".binary("utf-8")`, where `"data"` is the string written in the file. The file content must be binary.
- **File Access**: Specify how the file must be accessed.
  - To write to a file on the specified local or remote device, select **Direct Access**.
  - To write to a file on a robot file system, select **Via RFS**.
- **File Name**: Specify the path to the file to which the data is written.

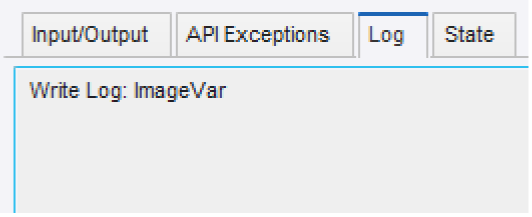
  For example, to write to a file on a robot file system, the path must start with the configured file system name and can be similar to the following: `myshare/myfile.bin`

  The file system name to use must correspond to that specified in the Robot File System section in the Management Console.

Write Log

The Write Log Step writes predefined Message into the Log right after the step it follows.

To enable the Log tab in Design Studio, run a robot in Debug Mode.



Write Log: ImageVar

To see the Message in the Management Console, go to **Log View** > **Robot Runs** tab and double-click a robot, containing Write Log Step.

Properties

Message: Specify either a text, a variable or an expression to write to the Log.

Automate Terminals

Kofax RPA supports connection to and interaction with 3270, 5250, 6530, and stream-based (vt100 and ANSI) terminals.

Command timeout for automating terminals is set either on the **Desktop Automation** tab of the Design Studio Settings window for executing the workflow in Design Studio, or in the **Desktop Automation** section on the **Security** tab of the **RoboServer Settings** window for RoboServer execution. See Runtime > Security in the Kofax RPA Administrator's Guide.

> **Note** You do not need to install the Desktop Automation service on your remote computer or create a device mapping in the Management Console when automating a terminal device.
>
> The terminals do not support mouse operations even though the Move Mouse and Click steps can be inserted. The terminals must be operated using keyboard keys.

Fonts

Kofax RPA bundles several fonts for the terminals to render their screens. They are located at:

```
C:\Program Files\Kofax RPA 11.1.0\nativelib\hub\windows-x32\XXX\fonts
```

Where XXX is a three-digit number for internal purposes.

If a terminal connection is established with a host that uses characters that are not found in the bundled fonts, the characters render as squares on the screen. This can be fixed by adding the path to a Truetype font in the `fontlist.txt` file found in the same Kofax RPA font directory. For example:

```
C:\TerminalFonts\MyTerminalFont.ttf
```

tn6530 terminals

To connect to 6530 terminals, insert the Terminal step, select **Nonstop (tn6530)** and specify all necessary parameters, such as connection type, credentials, and connection options.

Kofax RPA emulates a TN6530-8 terminal with an 80x24 display. The terminal starts in conversational mode.

Keyboard support

The following 6530 keys are supported in protected block mode:

- Return, Shift + Return, Ctrl + Return
- Home, Ctrl + Home
- End
- Backspace
- Tab, Shift + Tab
- Insert, Alt + Insert, Ctrl + Insert
- Delete, Alt + Delete, Ctrl + Delete
- Alt + 2, Shift + Alt + 2
- Cursor keys
- Function keys F1-F16. (For compatibility Alt+F1 – F6 are mapped to F11-F16 as well)
- 6530 function keys: Alt + Up, Alt + Down, PgUp, Alt + PgUp, PgDn, Alt + PgDn

Kofax RPA supports the following 6530 keys in conversational mode:

- Return, Shift + Return
- Function keys F1-F16. (For compatibility Alt+F1 - F6 are mapped to F11-F16 as well)

In addition, the following Calculated Keys are supported:

- VK_CLOSE

  This key terminates the session and closes the tn6530 terminal.
- VK_F11 - VK_F16

  These keys provide an alternative to the 6530 Alt+F1 – Alt+F6 key combinations mapped to the F11 – F16 function keys.

Connection options

Connection options are specified in the **Options** field of the **Terminal** step. Options are divided by ampersand (&). The option string can contain percent (%) escape characters.

For example, `NetworkTrace=MyLogfile.log&LineBuffer=2048`

The `tn6530` **Connect** action connects to the Telnet service (port 23) unless another port is explicitly specified in the **Host** field, such as `TerminalHost:11625`.

The following options are supported:

- `NetworkTrace=<logfile>`

  Creates a trace file containing all data exchanged with the host.
- `ColorMap=<color-map>`

  Sets the terminal to color support and optionally provides the default color map.
- `LineBuffer=<lines>`

Sets the number of buffered lines for the terminal. These lines are available for paging/scrolling. Accepted values are between 24 and 65536. If not specified, the terminal keeps a buffer of 1024 lines (conversational mode only).

The `ssh6530` **Connect (SSH)** action connects to the SSH service (port 22) unless another port is explicitly specified in the **Host** field, such as `TerminalHost:11625`. SSH login credentials can be specified using the **Credentials** option in the **Terminal** step.

The following options are supported:

- `SessionFile=<template>`

  Configures the default PuTTY configuration file. If this option is omitted, the `session.plink` file in the bin directory is used.

- `PublicKeyFile=<file>`

  Provides an SSH key file for authentication. This file must be in PuTTY private key format.

- `SSHUser=<user>`

  SSH user. This setting overrides the username in the **User Name** field of the step. If no user is specified, the robot is prompted interactively.

- `SSHPassword=<password>`

  SSH password. This setting overrides the password in the **Password** field of the step. If this option is omitted, the robot is prompted interactively.

- `SSHHostKey=<hostkey>`

  Configures the SSH host key identifying the host. If this setting is used and the value does not match the host key sent by the host, the connection is rejected.

  If this setting is omitted and there are no host keys configured through the PuTTY configuration file, the robot is prompted interactively and the robot must handle this dialog to accept the key. Note that the robot ignores settings in the Windows Registry and does not store host keys.

- `SSHLog=<logfile>`

  Enable SSH-level logging for troubleshooting purposes.

- `NetworkTrace=<logfile>`

  Creates a trace file containing all data exchanged with the host.

- `ColorMap=<color-map>`

  Sets the terminal to color support and optionally provides the default color map.

- `LineBuffer=<lines>`

  Sets the number of buffered lines for the terminal. These lines are available for paging/scrolling. Accepted values are between 24 and 65536. If not specified, the terminal keeps a buffer of 1024 lines (conversational mode only).

Any other options are used to substitute settings in the `SessionFile` template. Options that are not set in the `SessionFile` template are ignored.

The `logfile` parameters perform the following substitutions when creating a log file:

Pattern	Replacement	Examples
$P	PID of the process running the robot (value is platform dependent).	12928
$T	UTC time in Unix epoch format.	1524649335
$D	Local time in ISO 8601 format.	20180425T114215

Pattern	Replacement	Examples
$H	IP address of the host.	127.0.0.1
$P	Port of the host.	22
$$	A single dollar sign ($).	$

Color support

The `ColorMap=` setting switches the terminal from monochrome to non-configurable color mode. The host can update the color map by sending escape codes. The initial color map can be set up in the connection string by providing a string of 32 bytes in hexadecimal format specifying the color mapping for the 32 attribute combinations. See the 6530 documentation for a description of the attribute-to-color mapping scheme.

Known issues and limitations

- Support for Conversational Mode and Unprotected Block Mode is limited.
- Enhanced Color support and EM3270 mode are not available.
- Commands that perform local operations on the terminal are not supported.

Unsupported escape sequences and keys are tracked through the `**ERRORS:` marker in the message line.

Application actions

The **Application Action** menu is available when you right-click the terminal tab in the **Recorder View**. The menu includes the following items for the tn6530 terminals.

Action	Description
Close	Closes the terminal window.
Go To(row, col)	Moves the cursor to the specified location based on the row/col coordinates used in the application tree. Parameters are optional to allow easy horizontal and vertical movement.

tn5250 Terminals

To connect to 5250 terminals, insert the Terminal step, select **iSeries (tn5250)**, and specify all necessary parameters, such as host name, connection type and connection options. To use a port different from default, specify it in the **Host** field, such as `TerminalHost:11625`.

Connection options

Connection options are specified in the **Options** field of the **Terminal** step. Options are divided by ampersand (&). The option string can contain percent (%) escape characters.

`env.TERM` is the value of the client terminal type after the mode is switched by the URL. The default terminal type is "IBM-3179-2". Note that `env.TERM` parameter is valid for the tn5250 terminal only. Kofax RPA supports the following terminal types:

- "IBM-3477-FC"
- "IBM-3477-FG"
- "IBM-3180-2"
- "IBM-3179-2" (Default)
- "IBM-3196-A1"
- "IBM-5292-2"
- "IBM-5291-1"
- "IBM-5251-11"
- "IBM-5555-B01" (DBCS enabled)
- "IBM-5555-C01" (DBCS enabled)

Kofax RPA supports terminals in both 80x24 and 132x27 mode.

The 5250 terminal used a special keyboard with many keys that are not available on present day PC keyboards. To enter such keys, use the following calculated keys in the Press Key.

- VK_RETURN
- VK_ENTER
- VK_TAB
- VK_BACKTAB
- VK_UP
- VK_DOWN
- VK_LEFT
- VK_RIGHT
- VK_CLEAR
- VK_BACKTAB
- VK_F1
- VK_F2
- VK_F3
- VK_F4
- VK_F5
- VK_F6
- VK_F7
- VK_F8
- VK_F9

- VK_F10
- VK_F11
- VK_F12
- VK_F13
- VK_F14
- VK_F15
- VK_F16
- VK_F17
- VK_F18
- VK_F19
- VK_F20
- VK_F21
- VK_F22
- VK_F23
- VK_F24
- VK_ROLLDN
- VK_ROLLUP
- VK_BACK
- VK_HOME
- VK_END
- VK_INSERT
- VK_DELETE
- VK_RESET
- VK_PRINT
- VK_HELP
- VK_SYSREQ
- VK_CLEAR
- VK_REFRESH
- VK_FIELDEXIT
- VK_TESTREQ
- VK_TOGGLE
- VK_ERASE
- VK_ATTENTION
- VK_DUPLICATE
- VK_FIELDMINUS
- VK_FIELDPLUS
- VK_PREVWORD
- VK_NEXTWORD
- VK_PREVFLD
- VK_NEXTFLD

- VK_FIELDHOME
- VK_EXEC
- VK_MEMO
- VK_COPY_TEXT
- VK_PASTE_TEXT
- VK_NEWLINE
- VK_ERASE_EOF
- VK_KANJI

See the 5250 terminal documentation for more information.

To facilitate automation, Kofax RPA has added the following keys:

- VK_VIRTUAL_FIELDEXIT

  This key functions as VK_FIELDEXIT, but only if the cursor is not at the first position of a field. This can be used to exit a field if it is not completely filled, but suppress the FieldExit if the cursor has just wrapped to the next field.
- VK_DBCS_SPACE

  Types a DBCS space.

Character Encoding

To specify character encoding of the host, add a `LineCodePage` parameter to the **Options** field. For example: `LineCodePage=cp838`

For tn5250 terminals, Kofax RPA supports all EBCDIC codepages supported by the ICU library. The `LineCodePage` setting must match the code page setting of the terminal emulator used to connect to the host or the value of the `QCHRID` property of the terminal session.

Some frequently used code pages are listed in this table.

Character set	Host codepage
US/Canada	cp037 (default)
Multinational	cp500
Thai	cp838
Japanese	cp930, cp939
Simplified Chinese	cp935
Korean	cp933
EBCDIC 273 with the euro currency update	cp1141

Device assignment

By default the 5250 host assigns the first available device that is compatible with the requested terminal type or try to create a new device. If the robot needs to connect to a specific device add a `env.DEVNAME=<device>` parameter to the **Options** field. For example, to connect to the DEVROBOT device on the iSeries host:

```
env.DEVNAME=DEVROBOT
```

If the requested device does not exist or does not support the terminal type, or if the device is varied off/in use, the connection fails.

Enhanced 5250 interface support
To enable the enhanced interface for non-programmable workstations on the host, add the `enhanced=on` query parameter to the **Options** field.

This setting enables support for continued-entry fields.

> **Note** The DirectDraw Surface windows and graphical features are not supported when the enhanced interface is enabled.

Application actions

The **Application Action** menu is available when you right-click the terminal tab in the **Recorder View**. The menu includes the following items for the tn5250 terminals.

Action	Description
Close	Closes the terminal window.
Go To(row, col)	Moves the cursor to the specified location based on the row/col coordinates used in the application tree. Parameters are optional to allow easy horizontal and vertical movement.
Enter Field(text)	Enters the specified text in a field and proceeds to the next field using the [FieldExit] key.

3270 Terminals

Kofax RPA supports 3279-4 color 80x43 terminal, which is the default of the underlying terminal emulator.

The 3270 terminal used a special keyboard with some keys that are not available on present day PC keyboards. To enter such keys, the following calculated keys may be used in the Press Key.

- VK_RETURN
- VK_TAB
- VK_BACKTAB
- VK_Up
- VK_Down
- VK_Left
- VK_Right
- VK_F1
- VK_F2
- VK_F3
- VK_F4
- VK_F5
- VK_F6
- VK_F7
- VK_F8
- VK_F9
- VK_F10
- VK_F11
- VK_F12
- VK_F13
- VK_F14
- VK_F15
- VK_F16
- VK_F17
- VK_F18
- VK_F19
- VK_F20
- VK_F21
- VK_F22
- VK_F23
- VK_F24
- VK_ATTENTION
- VK_BACKSPACE
- VK_CLEAR
- VK_DELETE
- VK_DUPLICATE
- VK_HOME

- VK_INSERT
- VK_PA1
- VK_PA2
- VK_PA3

Character Encoding

To specify character encoding of the host, add a `charset` parameter to the **Options** field. For example:

```
charset=cp930
```

Kofax RPA supports the following character sets for tn3270 terminals:

Character name	Host codepage
belgian	500
belgian-euro	1148
bracket	037
brazilian	275
chinese-gb18030	1388
cp1047	1047
cp870	870
finnish	278
finnish-euro	1143
french	297
french-euro	1147
german	273
german-euro	1141
greek	423
hebrew	424
icelandic	871
icelandic-euro	1149
italian	280
italian-euro	1144
japanese-kana	930
japanese-latin	939
norwegian	277
norwegian-euro	1142
russian	880
simplified-chinese	935
slovenian	870

Character name	Host codepage
spanish	284
spanish-euro	1145
swedish	278
swedish-euro	1143
thai	1160
traditional-chinese	937
turkish	1026
uk	285
uk-euro	1146
us-euro	1140
us-intl	037

connecttimeout

To specify the wait time for the driver when the host is temporarily unavailable (that is if the host is unreachable or connection is refused), add the `connecttimeout` parameter to the Options field, indicating the timeout in seconds. For example:

```
connecttimeout=5
```

When this parameter is omitted, the driver waits for 20 seconds.

If the connection fails for reasons that cannot be resolved or the connection attempt cannot be retried (that is the host name cannot be resolved), the driver fails immediately.

Application actions

The **Application Action** menu is available when you right-click the terminal tab in the **Recorder View**. The menu includes the following items for the tn3270 terminals.

Action	Description
Close	Closes the terminal window.
Go To(row, col)	Moves the cursor to the specified location based on the row/col coordinates used in the application tree. Parameters are optional to allow easy horizontal and vertical movement.

SSH / Telnet Settings (vt100 and ANSI)

To use SSH connection, select **Connect (SSH)** for the emulator in the Terminal step. The underlying Kofax RPA SSH client is based on PuTTY and can potentially be configured to access any system that PuTTY can access. You can configure all the same parameters as you can with a PuTTY session, but instead of requiring the actual PuTTY client to define sessions, you can define session settings by adding optional parameters to the **Options** field.

The headless terminal does not automatically detect the code page on the server. This is why sending and receiving non-ASCII characters requires you to specify the code-page to use for character encoding and decoding. If you do not specify a code page, the robot assumes the server's locale is set to UTF-8. You can specify the code page by using the `LineCodePage` parameter in the **Options** field of the step.

The Kofax RPA terminal is built using "libicu" for character encoding. Use the `libicu` converter explorer at http://demo.icu-project.org/icu-bin/convexp?s=ALL to inspect code pages and their aliases. The list of supported character sets equals that of ICU (International Components for Unicode) library. See ICU documentation for details.

The following is a list of the most commonly used parameters for the stream-based terminals.

TermWidth	Sets the terminal width (default is 80).
TermHeight	Sets the terminal height (default is 24).
AutoWrap	Controls automatic line wrapping (default is N). Set to Y to enable line wrapping. This parameter applies to vt100 terminals only.
TerminalType	Sets the terminal type (default is "xterm").
LineCodePage	Specifies the code-page to use for character encoding and decoding (default is UTF-8).

To set the parameters on your session, enter them in the **Options** field of the **Terminal** step. Options are divided by ampersand (&).

The following connection options are supported.

SessionFile	Configures the default PuTTY configuration file. If this option is omitted, the `session.plink` file in the bin directory is used.
PublicKeyFile	Provides an SSH key file for authentication. This file must be in PuTTY private key format.
SSHUser	This setting overrides the username in the `user@host` part of the URI. If no user is specified, the robot is prompted interactively.
SSHPassword	This setting overrides the password in the `user:password@host` part of the URI. If this option is omitted, the robot is prompted interactively.
SSHHostKey	Configures the SSH host key identifying the host. If this setting is used and the value does not match the host key sent by the host, the connection is rejected.If this setting is omitted and there are no host keys configured through the PuTTY configuration file, the robot is prompted interactively and the robot must handle this dialog to accept the key. Note that the robot ignores settings in the Windows Registry and does not store host keys.
SSHLog	Enable SSH-level logging for troubleshooting purposes.

NetworkTrace	Creates a trace file containing all data exchanged with the host.

Application actions

The **Application Action** menu is available when you right-click the terminal tab in the **Recorder View**. The menu includes the following items for the vt100 terminals.

Action	Description
Close	Closes the terminal window.

SSH Authentication and Security Guide

When automating applications using SSH, the user can be authenticated in four different ways.

1. The underlying ssh client will prompt for a password. Enter the password using "Enter Text" step followed by a "Press Key" step (return).
2. An unencrypted private key can be placed on the file system of Design Studio and your RoboServers. Add the `PublicKeyFile=<path to key>` line to the **Options** field of the **Terminal** step.

   It is important that the key file is PuTTY formatted. On Linux you can use **puttygen** to transform an open-ssh private key.
3. An encrypted private key can be placed on the file system of Design Studio and your RoboServers. Add `PublicKeyFile=<path to key>` to the **Options** field. The underlying ssh client should prompt for a password upon connection. To enter the password, use "Enter Text" step followed by a "Press Key" step (return).
4. Use Pageant on Windows and ssh-agent on Linux with an encrypted key on the local file system.
   - On Windows, run Pageant as the same user running the RoboServer and add the key.
   - On Linux, run `ssh-agent` and `ssh-add` and copy the `SSH_AUTH_SOCK` and `SSH_AGENT_PID` environment variables to the RoboServer and Design Studio environment. You can do it by appending the lines to `RoboServer.conf` and `DesignStudio.conf` as follows.
     - `set.SSH_AUTH_SOCK=<value as outputted from ssh-agent>`
     - `set.SSH_AGENT_PID=<value as outputted from ssh-agent>`

     Or by setting the environment variables before starting the RoboServer and Design Studio.

The authentication methods above are listed from the least secure to the most secure as follows.

1. Anyone able to read the robot can extract the password and log onto the system.
2. The attacker would have to have the private key from your file system, so obtaining the robot from a Management Console is not enough.
3. The attacker would need both the robot and the private key file.
4. The attacker would need to obtain the password for the private key to gain access.

Use TLS/SSL

Kofax RPA supports TLS/SSL communication in terminals. To use TLS/SSL, use either **stn3270** or **stn5250** terminal in the Terminal step, specify the host name or IP address with an appropriate port number and select **Secure Connection**. Note that Kofax RPA does not verify the certificate presented by the server.

Basic Terminal Tutorial

See Automate Terminals for information on terminal prerequisites and settings.

In this tutorial, we will connect to a 5250 terminal, login, run some commands, and extract information from the terminal.

1. Open an existing or create a new Desktop Automation robot.
2. Open an existing project or create a new project (in Smart Re-Execution (Full) mode) and add a new robot with a Call Desktop Automation Robot step. Specify the name of the Desktop Automation robot opened by this step.
3. Add variables that will contain a login name and a password to log in to terminal. Also add a variable that will contain the output text. Add the variable with the output to the Return step (`=textVariableName`).
4. Execute the Call Desktop Automation Robot step and click the **Step Into DA Robot** button on the toolbar.
5. In the Desktop Automation workflow, add the Terminal step with necessary parameters. In this tutorial we connect to the 5250 terminal. Generally, the connection string is as follows: `tn5250://<hostname>:<portnumber>?env.TERM=<terminal type>`. Note that `env.TERM` parameter is valid for the tn5250 terminal only.
   - Emulator: iSeries(tn5250)
   - Action: Connect
   - Host: `terminal5250_server:11623`
   - Options: `env.TERM=IBM-3477-FC`

   Where `terminal5250_server` is the terminal name or IP address, and `:11623` is the terminal connection port number. If we omit the `env.TERM` parameter, the emulator connects to a default terminal type (IBM-3179-2). See Supported tn5250 Driver Terminals in the Automate Terminals.

   > **Note** To re-run your Desktop Automation workflow for a terminal with an Terminal step, leave the Desktop Automation robot, refresh your Web Automation robot with the Call Desktop Automation Robot step in Design Studio, and click **Step Into DA Robot**. If you re-run the robot without closing the terminal, another terminal window opens and the robot may fail to execute.

6. If the terminal needs an Enter key press, right-click the terminal in the Recorder View and select the Press Key step. By default it selects Enter as a key.
7. Right-click the User ID field in the terminal in the Recorder View and select **Enter Text** > **From Variable** > **login** and select the variable that contains the user name. Click **Step Into** to type this text into the text field.
8. To shift to the Password field, add the Press Key step and in the Key field select **Standard Keys** > **Tab**. Click **Step Into**. The cursor should move to the Password field.
9. Right-click the Password field in the terminal window and select **Enter Text** > **From Variable** and select the variable with the password. To actually type this text into the text field, click **Step Into**.
10. To log in, right-click the terminal in the Recorder View and select **Press Key** (Enter by default).

11. If the terminal needs an Enter key press, right-click the terminal in the Recorder View and select Press Key. By default it selects Enter as a key.
12. After you execute the required commands, you can extract the information from the terminal window. To extract a text line, right-click a row, select **Extract Context into** > **variableName**. Click **Step Into**. The Variables branch in the Workflow State view shows the value you extracted.

    To take a screen shot of the entire terminal window (you need to add a binary variable to the Return step (`=binaryVariableName`) beforehand), select the screen element in the Recorder View and click **Extract Image Into** > **binaryVariableName**. Later you can convert the information from the binary variable to an image in your Web Automation robot. Click **Step Into**.

Once you extract the information from the terminal, you can return to the Web Automation robot editor window and use the extracted information.

Access Websites

When creating a Desktop Automation workflow, you can open websites in the built-in browser and use action steps to extract information and navigate sites. The built-in browser is based on the selected engine, such as Chromium. To navigate, use Desktop Automation step actions. For more information about choosing a browser for your application, see Choose your browser.

> **Note** Command timeout for browsing websites is set either on the **Desktop Automation** tab of the Design Studio Settings window for executing the workflow in Design Studio, or in the **Desktop Automation** section on the **Security** tab of the **RoboServer Settings** window for RoboServer execution. See Runtime > Security in the *Kofax RPA Administrator's Guide*.

Kofax RPA browser supports the following protocols:

- `http:`
- `about:`
- `https:`
- `ftp:`
- `file:`

To open a website, insert the Browse step, select a browser engine in the **Browser** list, select **Load Page** as an action, and then specify all necessary parameters, such as the address in the **URL** field. Use the Tree Stops Changing guard after any step that includes loading a page or when other changes occur on a web page. See Guarded Choice for more information.

> **Note** Currently, it is not possible to use the system clipboard to copy and paste content inside one or multiple robots while using the built-in browser.

Browser interface
The built-in browser in Desktop Automation contains the following controls.

Button	Description
←	**Go back**: Navigates one page back.

Button	Description
→	**Go forward**: Navigates one page forward.
↻	**Reload**: Reloads the current page. When the page is loading, the button displays a cross to notify that the browser is busy.
http://www.kofax.com	**URL** field: Contains either the URL of the currently loaded page or the URL you pasted before navigating to the page.
>	**Navigate**: Goes to the URL entered in the URL field. If you cannot see the button, scroll the Recorder View window to the right.
⤓	**Save page** button: Saves the currently open page in HTML format. If you cannot see the button, scroll the Recorder View window to the right. For more information, see below.
⇅	**Configure proxy**: Opens proxy configuration dialog box. If you cannot see the button, scroll the Recorder View window to the right. For more information, see below.
PDF	**Print to PDF**: Saves the currently open to a PDF file. If you cannot see the button, scroll the Recorder View window to the right. For more information, see below.

URL in the built-in browser

Browser window contains the URL text field in the toolbar. It shows the URL of the loaded web page. You can select the URL and extract the value into a variable using action steps. The tree view shows the URL as well.

- To select the text in the URL field, click the URL field using the Click.
- To change the address, click the URL field by using the Click step and enter the address using either the Enter Text step or from a variable.
- To go to the entered URL, click the **Navigate** button to the right of the URL field.

Configure proxy

By default, all robots of the Desktop Automation Service use Kofax RPA global proxy settings. The Desktop Automation Service uses the same proxy settings as the Design Studio and Management Console. See Configuring Proxy Servers in Design Studio and Configuring Proxy Servers in Management Console for more information on proxy server properties.

Important Remember that the local proxy settings of the built-in browser in Desktop Automation Service have a higher priority than the Kofax RPA global proxy settings. Make sure the robot uses the Kofax RPA global proxy settings, unless the task requires it to use local proxy settings.

To change proxy settings for the built-in browser in Desktop Automation, click the **Configure proxy** button on the browser toolbar. The following proxy options are available:

- Direct: Proxy is not used.
- Fixed: Specify fixed proxy settings, such as host, port, and a bypass list.
- PAC: Specify the URL of the proxy auto-configuration script file.
- Auto: Click this option if your network provides for automatic proxy configuration, such as the Web Proxy Auto-Discovery Protocol.
- System: select this option to copy proxy settings from the computer running your robot.

Once proxy settings are set, click **OK** to save the settings and close the dialog box.

Print to PDF

You can save a web page in the PDF format using the **Print to PDF** button on the browser toolbar.

1. Click the **Print to PDF** button using the Click step.

   Once you execute the step, the browser opens the **Save page as PDF document** dialog box.

2. Specify the full path including the file name using the Replace Text step. By default, the file is stored in the temporary files folder configured for the current user.

   Click **OK**.

   - Optionally, you can save the file to a robot file system by selecting **Robot File System** in the dialog box. In the **RFS file name** field, enter the path to the configured file system and the file name, such as **myshare/downloaded.pdf**. The file system name must correspond to that specified in the Robot File System section in the Management Console.

3. Click **Save**.

To adjust the PDF file settings, such as page orientation, paper size, and scale, specify necessary options in the **PDF Settings** property of the Browse step.

> **Note** If you want to send an opened HTML page or a document to a hardware printer, due to limitations with handling of pop-up dialog boxes for printer selection, the system automatically opens the **Save page as PDF document** dialog box instead, which enables you to save your document in PDF format on a file system or a Robot File System for further use.

Save page

You can save the currently opened page in HTML format using the **Save page** button on the browser toolbar.

1. Click the **Save page** button using the Click step.

   Once you execute the step, the browser opens the **Save As** dialog box.

2. Specify the full path including the file name using the Replace Text step. By default, the file is stored in the temporary files folder configured for the current user.

   Click **OK**

3. Click **Save**.

Application-level actions

Application-level actions are actions applied to the entire application and available by right-clicking the application tab in the **Recorder View**. The list of application-level actions for the Chromium built-in browser includes:

- **Close Window**: Adds a step to close an opened application window.
- **Execute JavaScript**: Adds a step to execute JavaScript on the current page in the main window context.
  - In the **JavaScript** field, enter the code to execute.
  - In the **JS Execution Result** field, specify a text variable to store the result.
  - When **Wait For Result** is selected, the robot waits until the code is executed and the result is written into the "JS Execution Result" variable.

    The default timeout is 30 seconds. If the action is not completed within this time period, an exception is thrown. Also, if the JavaScript code contains errors, an exception is thrown.

    If you clear this option, the robot proceeds to the next step immediately after the code execution is started (without waiting for the execution to finish).

Component-level actions

Application-level actions are actions applied to the entire application and available by right-clicking an element in the **Recorder View** or the tree view. The list of component-level actions for the Chromium built-in browser includes:

- **Scroll To**: Adds a step that scrolls the page to the selected element, so the element appears in the middle of the window.

Debugging with Chrome Inspector

Kofax RPA supports debugging in the Chromium browser engine using Chrome Inspector in the Device Action step. The Inspector can help you extract and save information necessary to analyze the interaction between the website and the browser and errors occurred during processing of the web content. Information is saved in the HTTP Archive (HAR) format. To save the log to a file, perform the following steps.

1. Open `cef.cfg` file located at `<Kofax RPA installation folder>\nativelib\hub\windows-x32\<hub version number>\node_modules\cef` in a text editor.

   For example: `C:\Program Files (x86)\Kofax RPA 11.1.0.0 x32\nativelib\hub\windows-x32\1267\node_modules\cef`

2. Set `show_dev_tools` property to `true`, save the file, and reload your robot.

   Once you execute the Browse step that loads a web page, the **Chrome DevTools** window attached to the corresponding web page opens.

3. In the **Chrome DevTools** window, select the **Network** tab, click **Preserve log**, and press Ctrl+R. The network trace loads in the window.

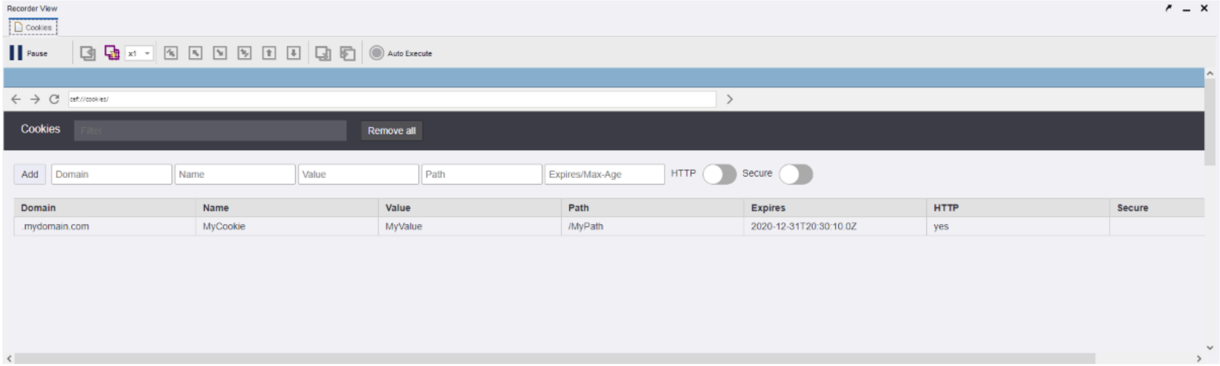
4. Right-click the trace and select **Save all as HAR with content** to save the network trace in a file. Now you can open the file in a HTTP Archive inspector.

Cookie Management in Chromium built-in browser

Use the Desktop Automation built-in browser to monitor, add, or delete cookies from the CookieStore.

Manage cookies on the Cookies page

To manage cookies in the Chromium built-in browser, open the Cookies page by inserting the Browse step. Then select **Chromium** in **Browser**, type `cef://cookies` in the **URL** field and execute the step.



cef://cookies/

Domain	Name	Value	Path	Expires/Max-Age

Domain	Name	Value	Path	Expires	HTTP	Secure
.mydomain.com	MyCookie	MyValue	/MyPath	2020-12-31T20:30:10.0Z	yes	

Add a new cookie to the CookieStore

1. Specify the cookie attributes in the corresponding text boxes:
   - Domain
   - Name
   - Value
   - Path
   - Expires/Max-Age
2. Click the **HTTP** toggle to enable the HTTP restriction.
3. Click the **Secure** toggle to enable an encrypted connection (HTTPS).
4. Click the **Add** button to add the cookie.

Filter the cookies

All cookies on the Cookies page can be filtered by any attribute value.

To filter cookies by attribute, type any attribute value in the Filter field.

Delete the cookies

Use the Cookies page to delete cookies from the CookieStore either separately or all together.

- To delete a cookie from the CookieStore, click the Delete 🗑 button at the end of the line.
- To delete all cookies from the CookieStore, click the Remove all button.

Manage Cookies with the Browse Step

Use the Browse step to add or delete cookies directly by a request.

- To add a new cookie to the CookieStore, type the following text (substituting the values) in the **URL** field of the Browse step:

  ```
  cef://cookies/create?domain=MyDomain.com&name=MyCookie&value=MyValue&path=/
  MyPath&expires=2020-12-31T20:30:10.000Z&http=on&secure=off
  ```

- To delete a cookie from the CookieStore, type the following text in the **URL** field of the Browse step:

  ```
  cef://cookies/remove?domain=.MyDomain.com&name=MyCookie&value=MyValue&path=/
  MyPath&expires=2020-12-31T20:30:10.000Z&http=on&secure=off
  ```

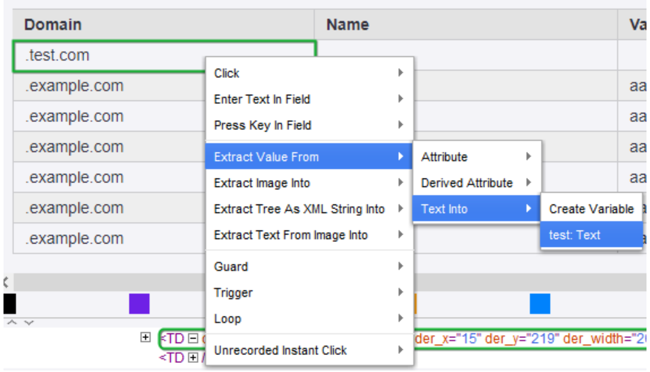
- To delete all cookies from the CookieStore, type the following text in the URL field of the Browse step:

  ```
  cef://cookies/removeall
  ```

Extract Value from a Cookie

On the Cookies page, you can extract a cookie value to a text variable.

Right-click the cookie attribute to extract its value, using the context menu as shown below.



You can also extract cookies from the CookieStore and store them in a JSON variable of complex type.

Built-in Excel Driver

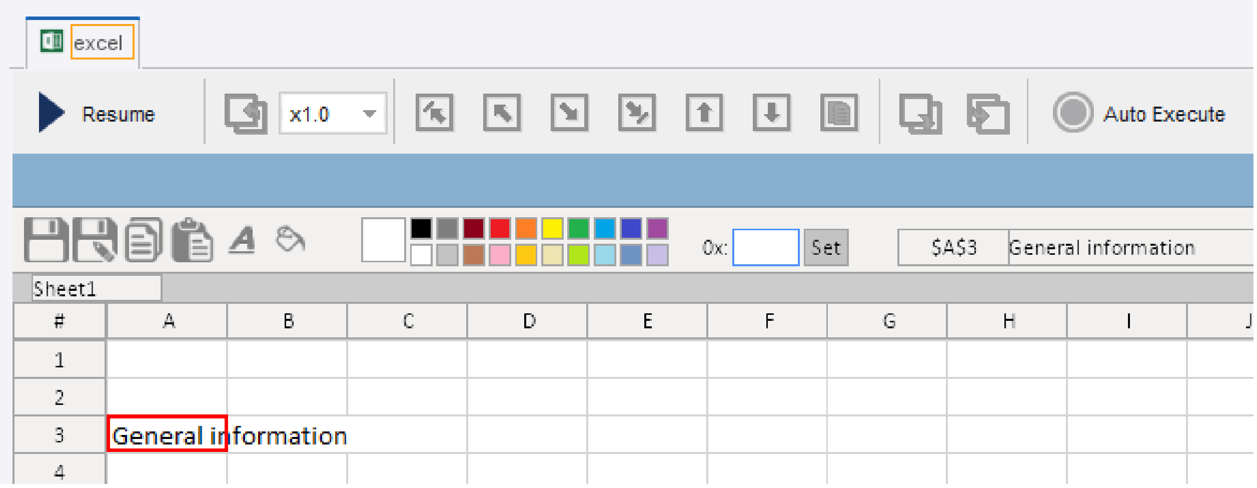
The Desktop Automation workflow includes built-in Excel driver that helps you perform some operations on Excel spreadsheets.

> **Note** To work with built-in Excel driver in Desktop Automation, Microsoft Excel must be installed on the computer where the robot is run.
>
> The built-in Excel driver is available only for local automation device.

To create a new Excel spreadsheet, insert the Excel step and select **Create File** in the **Action** list.

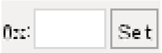
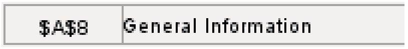
To open an existing Excel spreadsheet, insert the Excel step and select **Open File** in the **Action** list and type the full path to the spreadsheet. For example, `excel://c:/documents/myspreadsheet.xlsx`.

Note that Kofax RPA can create a spreadsheet with only one worksheet. When you open a spreadsheet with several worksheets, you can navigate between them by clicking a button for the respective worksheet. Chart sheets do not contain elements that can be manipulated by a robot and are therefore not shown in the tree of the editor. The Chart sheets are included when the spreadsheet is saved.



Use toolbar buttons in the built-in Excel driver to perform the following operations.

Button	Description
💾	**Save**: Saves the changes in the spreadsheet.
💾	**Save As**: Opens the Save As dialog box to save the current spreadsheet under a different name.
📄	**Mark to copy**: Marks current selection for copy/ paste action. For selection options see "Select cells" subsection below this table.
📋	**Paste**: Copies the previously marked selection to the selected cells. See "Clipboard operations" below for details.
A	**Set text color**: Applies active color to the selected text.
🪣	**Set background color**: Applies active color to selected cells.
⬜ ⬛⬛🟥🟥🟧🟨🟩🟦🟪 ⬜⬜🟫🟥🟧🟨🟩🟦🟪	**Color Picker**: Sets an active color to use for color changing actions.

Button	Description
	Custom color input: Specifies custom color for the Color Picker in hexadecimal (hex) format. For example, pure white is `ffffff`, pure black is `000000`.
 A8 General Information	This text field shows the address of the currently selected cell and its value.

Select cells

You can select one or more cells in the worksheet as follows.

- To select one cell in the worksheet, click it just as you would do in a desktop version of Microsoft Excel.
- To select a row, click the row heading.
- To select a column, click the column heading.
- To select the entire worksheet, click Select All #.
- To select a range of cells in a worksheet, use arrows with Shift in the Press Key. Note that selection is made from left to right and from top to bottom. You cannot select cells by moving your selection up and from right to left.

  For example, to select a range that is five cells wide and three cells in height, do the following.

  1. Click a cell located at the top left corner of your range.
  2. Insert the Press Key step. In the step, specify `excel` as the application name in the finder, select the **Right Arrow** in the **Standard Keys**, select **Shift** as the key modifier, and enter 4 in the **Count** field.
  3. Execute the step.
  4. Insert the Press Key step. In the step, specify `excel` as the application name in the finder, select the **Down Arrow** in the **Standard Keys**, select **Shift** as the key modifier, and enter 2 in the **Count** field.
  5. Execute the step.

  As a result, you should see the selected range of cells in your worksheet.

Change color

To use color changing actions, first either select a color in the Color Picker or specify a custom color in the Custom color input. To specify a custom color, click the text area in the Custom color input, enter the required color in hex format, and click **Set**. Once the active color is set, use "Set text color" or "Set background color" actions.

Clipboard operations

The built-in Excel driver is designed to be used in robots running on RoboServer instances. Do not use Windows Clipboard in the built-in Excel driver, because the clipboard is shared between all running robots.

To copy content within an Excel document, use the **Mark to copy** and **Paste** buttons on the toolbar or Copy/Paste application and component actions.

> **Important** You can only copy and paste between sheets within the same Excel document. If you have more than one Excel documents opened, you cannot copy from one and paste into the other.
>
> If you cut or copy information from an Excel document, the information is not copied to the clipboard and the "Extract clipboard" step does not provide the content you just cut or copied from an Excel sheet.

Freeze Tree operations

You can work with spreadsheets using Excel driver within the Freeze Tree group step. This helps you perform cell loops faster and create a more efficient robot. To use the Excel driver within the Freeze Tree step, insert the Freeze Tree step in your workflow and then insert the Open step that calls the Excel driver as described in this topic.

Application and component actions

The Application Action menu is available when you right-click the **excel** tab in the **Recorder View**. The Component Action menu is available when you right-click a component in the Recorder View or Tree View. The menu includes all items available on the built-in Excel toolbar with the following additional items. "AA" denotes application action menu and "CA" denotes component action menu.

Action	Description
Select AA and CA	Selects a cell in the current worksheet. Specify the cell address in the **Select** property of the Select step. For example, `$b$4` selects the fourth cell in column b. To select a range of cells, use the **Expand From Cursor** option to expand the active selection into a rectangular, which includes both the currently selected cells and a new range or use A1:B2 notation.
Select Sheet AA and CA	Opens the specified worksheet. Enter the worksheet name in the **Select Sheet** property of the Select Sheet step.
Scroll To AA	Scrolls to and makes the selected cell visible on the screen.
Get Value AA and CA	Extracts the content of one or more cells into a text variable. You can use this action to either extract formatted cells, raw content, or formula definitions from the cell. If a range with multiple cells is specified, data from cells is separated by tabs and rows in the result.
Get Number Value AA and CA	Extracts the content of a cell into a Number variable. This action extracts binary data and is not affected by locale settings. If the content of the cell cannot be converted to a number or it is an Excel `#error` value, you can specify to return `Error Value`. Otherwise 0.0 is returned.
Get Hyperlink AA and CA	Extracts the URL of the hyperlink in the specified cell or an empty value if the cell does not contain a hyperlink. **Note** Hyperlinks can be associated with a range that can contain multiple cells; this action works on any cell in the range.

Action	Description
Get Sheet Name AA	Extracts the name of the active worksheet.
Clear AA and CA	Clears a range of cells. Use this action to clear formatting, content, or both.
Set Value AA and CA	Sets a value in a range of cells and optionally applies formatting to the cells. You can use this action to set data or formulas. If you select **Custom** in **Format**, the formatting is set to the value of the Custom Format field.
Set Number Value AA and CA	Sets a Number value in a range of cells and optionally applies formatting to the cells. The copied value is binary, and not affected by locale settings. If you select **Custom** in **Format**, the formatting is set to the value of the Custom Format field.
Set Formatting AA and CA	Sets the formatting for a range of cells without changing the content of the cells. If you select **Custom** in **Format**, the formatting is set to the value of the Custom Format field.
Set Hyperlink AA and CA	Assigns a hyperlink to a range of cells. The link is visually presented in the first cell of the range, but applied to all cells in the range.
Set Sheet Name AA	Changes the name of the active worksheet.
Set Column Width AA and CA	Sets the width of all columns containing cells in the range to a specific width or select to autofit the cells based on content. Note that in Excel the actual width of columns is an approximation of the provided value.
Set Row Height AA and CA	Sets the height of all rows containing cells in the range to a specific height or select to autofit the cells based on content. Note that in Excel the actual height of columns is an approximation of the provided value.
Insert Sheet AA	Inserts a new spreadsheet in the document with the specified name. This action does not change the active spreadsheet.
Insert Rows AA and CA	Inserts one or more rows relative to the first cell in the range.
Insert Columns AA and CA	Inserts one or more columns relative to the first cell in the range.
Delete Sheet AA	Deletes the active worksheet. It is not possible to delete the last worksheet in the workbook.
Delete Rows AA and CA	Deletes all rows containing cells in the range.
Delete Columns AA and CA	Deletes all columns containing cells in the range.
Test Cell Type AA and CA	Performs a test on all cells in the range. Returns true if all cells meet the condition; returns false otherwise. The tests use the equivalent of Excel functions.

Action	Description
Set Named Range AA and CA	Adds a named range to the workbook or worksheet which references the range. Clear the **Visible** option to hide the range in Excel’s GUI after the workbook is saved.
Close	Closes the Excel window discarding any unsaved changes.

Tips for Windows Environment

This section explains how to prevent errors while running robots within a Windows service. In this case, the built-in Excel driver may need some additional configuration in Windows to work correctly.

The following errors may occur during the execution of the Open step:

Device Issue. Error Executing Command: Error Creating Excel Instance: Unknown Error 0x800A03EC

This error is caused by an Excel's headless mode behavioral feature.

To resolve the error, perform the following:

1. Create two empty folders (if they do not exist yet):
   - `C:\Windows\system32\config\systemprofile\Desktop`
   - `C:\Windows\SysWow64\config\systemprofile\Desktop`
2. Make sure the user of Windows service has access rights to these folders. These folders allow Excel to be started from a service.

Error Executing Command: Error Creating Excel Instance: Server Execution Failed and Error Executing Command: Error Creating Excel Instance: Access Denied

These errors can be caused by insufficient Excel COM Launch and incorrectly configured Activation rights. To resolve both errors, perform the following:

1. In the `Run dialog` or `Windows Start menu`, start the `dcomcnfg.exe` application.
2. In the opened window, navigate to **Component Services > Computers > My Computer > DCOM Config > Microsoft Excel Application**.

   > **Note** If there is no **Microsoft Excel Application** entry, you may need to start the `dcomcnfg.exe` application for a different architecture.
   >
   > To do this, open a command-line window, navigate to `C:\Windows\SysWOW64`, and execute the following command: `mmc comexp.msc /32`

3. Right-click the **Microsoft Excel Application** entry and select **Properties** .
4. On the **Security tab**, select **Customize** for **Launch and Activation Permissions** and click **Edit**.
5. Click **Add** to add a user or a desired user group if they are not on the list.
6. Make sure the check boxes **Local Launch** and **Local Activation** are selected.
7. Save all the changes and restart the Windows service.

Attended Automation

Attended Automation is a way to automate your remote computers by creating a trigger robot that reacts to an event on a remote device. The following steps help you set up your development and production environment to use robots with triggers.

> **Important** Starting from Kofax RPA version 10.7, using robots with triggers requires additional configuring. See "Windows tab" section of the Configure Desktop Automation Service for more information.

Prerequisite: During robot development, set the Desktop automation service on the remote device to the **Single User** mode to be able to access the device from Design Studio.

1. Create a simple, direct connection Device mapping for your trigger robot.
2. Add the created mapping as a "Trigger reference" to the **Required Devices** in the Call Desktop Automation Robot step using the **Add Device** dialog box.
3. Add Trigger Choice to your Desktop Automation robot.
4. Once the development is complete, open the Desktop automation service configuration on the remote device, clear the **Single User** option and specify the Management Console where the robot is deployed.

After uploading a robot with a trigger to a Management Console, map the robot to users and labels in the Trigger mappings section. After that the Management Console provides a list of triggers to the Desktop Automation Service based on created mappings. When a trigger event is detected on a remote device, the Desktop Automation Service sends a notification to the Management Console and the robot performs some programmed steps. For example, you can program the robot to insert or extract some data when a certain application is opened. Use the Trigger Choice step to define triggers and actions that start when a certain event is detected.

Don'ts for robots with triggers

- Do not add robots with triggers to schedules and do not start robots with triggers manually. The robot started by schedule or manually will wait forever (or some specified RoboServer timeout), because the trigger is never activated by any remote Desktop Automation Service.
- Do not add robots with triggers to Kapplets.
- Do not use Intelligent Screen Automation (ISA) in robots with triggers.

When configuring a Desktop Automation Service, you can use labels to distinguish the automation devices. For example, if a Desktop Automation Service is installed on a computer that is used by an active user and that should run attended robots to assist the user with manual tasks, you can add a respective label to such Desktop Automation Service, like "interactive." If the computer is not used actively and non-attended (usual) robots can run on it, you can add a label similar to "non-interactive."

Trigger mappings are used to assign users and labels to robots with trigger events in the Management Console.

Also, you can assign mappings, suspend and activate triggers in the Repository > Robots section in Management Console.

When a trigger event is detected, the robot might prevent the user from using the mouse and keyboard. To inform the user of the action performed by the robot, use the Notify step.

If triggers are suspended in a Management Console, the robot is not triggered by the remote Desktop Automation Service. Note that the refresh of trigger information runs during the connection to a remote device and then every minute. There might be a situation when a trigger is shown as suspended, but is still running in the Management Console. In this case, a trigger event may still be detected.

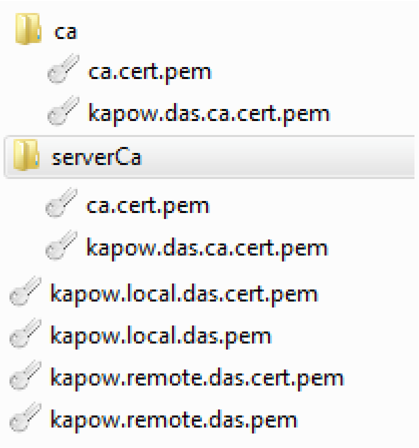
Use TLS Communication

Kofax RPA provides a means for setting up TLS communication between Automation Device and RoboServer or Design Studio. The communication uses certificates for encrypting the communication. The encryption uses a public-private key structure for securing the connection.

Note Password protected certificates are not supported.

The key files must be in the "pem" format, which is the most common format for openSSL.

The installation package for Desktop Automation Service includes six files and two folders.



```
ca
    ca.cert.pem
    kapow.das.ca.cert.pem
serverCa
    ca.cert.pem
    kapow.das.ca.cert.pem
kapow.local.das.cert.pem
kapow.local.das.pem
kapow.remote.das.cert.pem
kapow.remote.das.pem
```

The `ca.cert.pem` is a public key file signed by a private key created by Kofax RPA. It acts as the root certificate for this trust chain of keys. The `kapow.das.ca.cert.pem` is another signed certificate that is signed by the root private key. These two files exist in both the `ca` and the `serverCa` folder. If you do not have any specific security requirements, these files can be used out of the box.

The `kapow.local.das.pem` is the private key file used by the local hub that exists on the RoboServer and Design Studio. The `kapow.local.das.cert.pem` is the public key signed by the underlying private key for `kapow.das.ca.cert.pem`.

The `kapow.remote.das.pem` is the private key file used by the Desktop Automation Service. The `kapow.remote.das.cert.pem` is the public key signed by the underlying private key for `kapow.das.ca.cert.pem`.

The files have the same code for Automation Device and RoboServer or Design Studio.

The Automation Device must have the kapow.remote.* files along with the serverCa folder containing the ca.cert files.

The RoboServer or Design Studio must have kapow.local.* files and the ca folder containing the ca.cert files.

When using homemade certificates, the certificates (signed public key files) of trusted authorities must be in the ca folder (serverCa for Automation Device). Node.js checks the certificates from the Automation Device for trust (certificates are trusted if there is a chain of signed certificates to a certificate in the ca/serverCa folder). If the certificate from the Automation Device is trusted, the Desktop Automation Service verifies the certificates from the RoboServer or Design Studio in the same manner.

The RoboServer or Design Studio requires just the certificates to be trusted. The same certificate is used from multiple Automation Devices, so the Automation Device name does not have to match the "common name" in the certificate.

If you want to change the certificates to your own validated or homemade certificates, you can do it in two ways.

- Recommended
  1. Get new server certificates and install them on the Automation Device by copying the private key and the public key to a folder on the device. Use the **Certificates** tab in the Desktop Automation Service configuration window to change the path of the certificates to the new ones.
  2. Get the new client certificates for Design Studio and install them on a computer running Design Studio. Open the **Desktop Automation** tab in Design Studio Settings window and specify the paths to the client certificates.
  3. Get the new client certificates for the RoboServer and install them. On the **Security** tab of the RoboServer dialog box, specify the path to the new certificates.
- Alternative
  - Rename custom certificates to appropriate Kofax RPA names, such as kapow.local.das.pem, kapow.local.das.cert.pem, and so on. Overwrite the supplied certificates with the new ones.

Data Conversion

The conversion functionality of Desktop Automation facilitates data conversion and robot development. With this functionality, you can automatically convert manually entered data and, together with extract steps, convert extracted data and then store the result in variables.

Collectively, the group of the following steps is referred to as a converter group:

- Convert Value
- Extract Value
- Extract Clipboard
- Extract Tree as XML

You can use these steps to extract and/or convert data. To these steps, you can add an Evaluate Expression step that performs the actual data conversion.

As opposed to other steps in Desktop Automation, if you need to undo a change to workflow state inside a converter group or see the state at a certain preceding flow point, you can simply go back inside the step by executing the required preceding flow point. Also, if you change the workflow before the current flow point, the workflow state is adjusted automatically, so you do not need to reexecute the steps.

The conversion and extraction are performed with the help of the following built-in variables:

- The **$current** variable contains the initial value. The value of this variable is changed during execution. The type of the value can also be changed, such as a number value can be converted to a text value.
- The **$initial** variable also contains the initial value, but the value remains unchanged during execution, which enables you to reuse the initial value any time, such as to convert it again.

Expressions

This section describes expressions in Desktop Automation and how they are edited and evaluated.

Many properties on Desktop Automation steps can either be specified as plain value (for instance, a number), or as an expression. An expression is evaluated and then the result of this evaluation is used for the property where plain values are used directly as the value of the property. For instance, the Count property on the Click step could be specified as a number, such as 2, but could also be specified as an expression, such as `clickCount` where `clickCount` is a variable defined in the scope of the step and given a value somewhere else in the Desktop Automation robot.

Expressions in Desktop Automation are very similar to expressions in most common programming languages, such as Java, C#, JavaScript, etc. They consist of constants, variables, operations (addition, subtraction, multiplication, comparison operations, logic operation, and etc.), and functions. The following are a few examples of expressions:

- `(1 + 2)*3`
- `x > 0 || x <= 6`
- `max(x, 10)`
- `"hello".substring(3)`

Expressions are typed and these types are the same as variables have. This means that operation and function may only be applied to operands of a certain type. Depending on the operand type, it returns a value of a given type when evaluated. For example, addition of two operands of type Integer provides a result of type Integer, such as `1+2` evaluates to `3`. If the type of the operands is Number, then the result is of type Number, such as `1.0+2.0` evaluates to `3.0`. The type of an expression is statically checked before the expression is evaluated and a type error in an expression is reported as an error in the Desktop Automation robot.

Evaluation of an expression cannot change the state of the workflow, that is you cannot assign a value to a variable inside the expression. Only steps can do this, for instance the Assign step assigns a value to a variable and this value may come from evaluating an expression.

The following sections explain different components of expressions.

Constants

Constants are values of the following simple types.

Type	Example
Integer	42 -17
Number	3.14159 -.33
Boolean	true false
Text	"Hello" "First"

Text values must not contain double quotes (") as this terminates the Text value. Instead use \" when you need a double quote in your Text value. The backslash sign (\) is generally used for special characters in Text values that you cannot write directly in expressions. The special characters are:

Character	Description
`\n`	line break **Note** To use line break in the built-in browser in Desktop Automation, use `\r\n`
`\r`	carriage return
`\f`	form feed
`\"`	double quote
`\t`	tab
`\b`	backspace
`\\`	the backslash character itself
`\uXXXX`	Unicode character coded in a hexadecimal number, for example `"\u002A"` is an alternative way of writing `"*"`

Variables

Variables in an expression can be any variables or input parameters defined in a workflow that are in scope at the location of the expression. Input parameters are always in scope, because their scope is the entire workflow. Variables are in scope if they are defined at the top level of the workflow or inside a Group step.

Operations

An operation is an expression consisting of an operator and some operands. In the expression `1+2`, `+` is the operator and `1` and `2` are the operands. So the operator defines an operation that should be performed on the value of the operands when the expression is evaluated. In this section we describe the operators that can occur in expressions. In most cases the operation that these operators perform is straightforward. If you are familiar with expressions in programming languages, you can skip this description and consult the summary table below.

Arithmetic operations

Expressions support normal arithmetic operations, such as addition (+), subtraction (-), multiplication (*), division (/), and modulo (%). Each of the operations take two operands that can have any combination of type Integer and Number. If at least one of the operands is of type Number, the result type is also Number. Otherwise it is of type Integer.

When using an addition operation (+), if one of the operands is Text and the other operand is of type Integer, Number, Boolean or Text, the result type is Text. For example, `"a" + 1` evaluates to the text `"a1"`. The value of the operand that is not of Text type is converted into text and then the values of the two operands are concatenated into the resulting text. The subtraction operation - can also be used as negation of numbers, such as `-x` where x is of type Integer or Number. The operator `%` is called a modulo, or remainder operator. It returns the remainder after division of two operands, for example, `5 % 2` returns `1`. More precisely it is defined mathematically as follows:

```
x % y = x - trunc(x / y) * y
where
trunc(x) = sgn(x) * floor(|x|)
```

Evaluation of an arithmetic operation may result in an exception thrown. This can happen for addition, subtraction, multiplication, and division operations if the result is outside the limit for numbers, such as overflow if a Number value is too big, in which case an `OverflowIssue` exception is thrown. The division and modulus operators throw a `DivisionByZeroIssue` exception if the value of the second operand is zero. For example:

- `17 % 2` evaluates to `1`
- `-17.3 % 2.0` evaluates to `-1.3`

Equality operators

There are two equality operators in workflow expressions.

- `==` Determines if the value of one operand is equal to another
- `!=` Determines if the value of one operand is not equal to another

These operators work on operands of all types, but the type of the operands must be the same, for instance, you cannot compare a Number to an Integer.

Relational operators

Relational operators determine if one operand is less than or greater than another operand. The operands must be numbers, that is of type Integer or Number and the types of the operands in an expression must be the same. There are four relational operators:

Operator	Description
<	less than
<=	less than or equal to
>	greater than
>=	greater than or equal to

Logical operators

There are two binary (taking two operands) logical operators: AND (`&&`) and OR (`||`), and one unary (takes one operand): NOT (`!`). They are defined for Boolean operands and their return type is also Boolean. The `&&` operator returns true if the value of both of its operands is true and false in any other case. The `||` operator returns true if the value of at least one of its operands is true and false in case they are both false. The `!` operator returns true if the value of the operand is false and returns false if the operand is true.

Evaluation of the `&&` and `||` operators is slightly different from evaluation of most other operators. Normally all operands are evaluated before the operator is evaluated, but for the `&&` and `||` operators the first operand is evaluated first and if this is enough to determine the result of the operation, the second argument is not evaluated. For example, in `x==1 || x==2` if `x` is 1, the second part of the expression (`x==2`) is not evaluated.

Conditional operator
Conditional operator takes three operands and has the following form:

`<Op>?<Op>:<Op>`

where `<Op>` can be any operand with some restrictions. For example, `x==1?0:1` evaluates to `0` if the value of `x` is `1` and to 1 otherwise. The type of the first operand must be Boolean and the other two operands can be of any type, but must be the same.

Evaluation of the conditional operator is also slightly different from the evaluation of most other operators. For the conditional operator, the first operand is evaluated first and then depending on its value, only one of the other two operands is evaluated. If the first operand is true (or false) then the second (or third) operand is evaluated and the result is the result of this evaluation. This also means that even if an evaluation error occurs in the operand that is not evaluated, this does not lead to an exception thrown. For example, in `x == 0.0? 1.0: 1/x` if `x` has value `0.0`, `1/x` is not evaluated and no `DivisionByZeroIssue` exception is thrown.

Summary of operators
The table below lists the expression operators.

Operator	Description	Examples

`+`	Addition or text concatenation	`1+2`
		`"hello " + name`

`-`	Subtraction or negation	`1-2`
		`5-2.9`
		`-5`

`*`	Multiplication	`42*2`
		`1.0*17`

`/`	Division	`1/2`
		`1/2.0`

`%`	Modulus	`x % 2`
		`2.5 % 1.0`

`==,!=`	Equals, not equals	`true == false`
		`x != 0`

`<,<=`	Less than, less than or equals	`0 < 1`
		`1.0 <= 0.0`

`>,>=`	Greater than, greater than or equals	`0 > 1`
		`1.0 >= 0.0`

`&&,||`	Logical AND, logical OR	`true || x`
		`false && y`

Operator	Description	Examples
`!`	Logical NOT	`!true`
`_?_:_`	Conditional operator	`x>0? x: 0`

Parenthesis

You can use parenthesis to determine the order of evaluation in an expression and alter the result obtained from the expression. For example, the expression `1+2*3` evaluates to 7, but if you insert a parenthesis as follows: `(1+2)*3`, the result changes to 9, because the content of the parenthesis is evaluated before the surrounding operator.

Function

Expressions can also contain function calls. There are two ways to call a function. The first is called a direct function call and looks like this: `f(<Op>,…,<Op>)`, such as `max(1,2)`. The other is called a method function call and it looks like this: `<Op>.f(<Op>,…,<Op>)`, for example, `1.max(2)`. The two ways are related as follows:

<Op1>.f(<Op2>,...,<Opn>) is the same as f(<Op1>,...,<Opn>).

Functions are similar to operators in that the operands must have certain types and the result type depends on the types of the operands. For example, the function `max` that determines the maximum of two numbers can be called with operands of type Integer or Number and the return type is the same as the type of the operands.

If during the evaluation a function gets an operand value that is incorrect, such as outside the expected range, an `IncorrectValueIssue` exception is thrown.

Kofax RPA provides the following functions.

Numeric functions

Function	Result type
abs(Integer)	Integer
abs(Number)	Number
ceil(Number)	Integer
computeMD5(binary: Binary)	Text Computes an MD5 checksum of the binary input.
floor(Number)	Integer
round(Number)	Integer
trunc(Number)	Integer
max(Integer, Integer)	Integer
max(Number, Number)	Number
min(Integer, Integer)	Integer
min(Number, Number)	Number
random()	Number Returns a random number greater than or equal to 0.0 and less than 1.0.

Function	Result type
random(Integer, Integer)	Integer Returns a random integer greater than or equal to the value of the first operand and less than or equal to the value of the second operand.

Examples	Evaluates to the following result
abs(-2)	2
1.5.round()	2
random(1,6)	an integer value between 1 and 6

Text functions

Function	Result type
length(Text)	Integer
substring(Text, Integer)	Text
substring(Text, Integer, Integer)	Text
indexOf(Text, Text)	Integer
contains(Text, Text)	Boolean
trim(Text)	Text
capitalize(Text)	Text
startsWith(Text, Text)	Boolean
endsWith(Text, Text)	Boolean
toLowerCase(Text)	Text
toUpperCase(Text)	Text
matches(text: Text, regex: Text)*	Boolean Checks that the text matches the regular expression.
removeNonPrintable(text: Text)	Text Removes non printable characters.
replaceAll(text: Text, regex: Text, replacement: Text)*	Replaces all sub-text that matches the regular expression with the given replacement.

Function	Result type
unquote(text: Text)	Text Removes any quotes from a text. For example, both `"hello"` and `'hello'` produces `hello` without quotes.

Note * This function may run for a long time depending on the used text and regular expression. For example, entering a lot of extra zeros in the text causes the `matches` function to run very long. Adding a single `A` at the end makes it return true almost immediately such as `matches("0000000000000000000000", "(0*)*A")`. Every time you change the expression, the previous evaluation is cancelled (if it is still running) and a new evaluation is started.

Example	Evaluates to the following result
"workflow".substring(5)	low

Conversion functions

Conversion functions convert values from one type to another. Conversion may fail if the value of the operand does not represent a value that can be converted into a value of the result type.

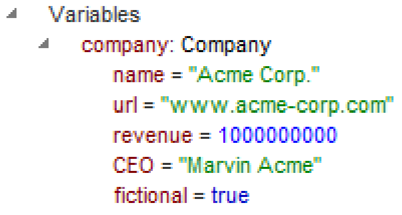
Function	Result type
ampersandEncode(text: Text)	Text Ampersand encode a text.
ampersandDecode(text: Text)	Text Ampersand decode a text.
base64Encode(binary: Binary)	Text Base64 encode using Base64 transfer encoding for MIME (RFC 2045).
base64Decode(text: Text)	Binary Base64 decode using Base64 transfer encoding for MIME (RFC 2045).
boolean(text: Boolean)	Boolean The text must match either "true" or "false".
integer(Number)	Integer The number must be an integer value, for example 1.0
integer(Text)	Integer The text must be a text representation of an integer, such as "42"
number(Integer)	Number
number(Text)	Number Text must be a text representation of a number, such as "17.7"
text(Integer)	Text

Function	Result type
text(Number)	Text
text(Boolean)	Text
text(binary: Binary, charsetName: Text)	Text Converts a binary representation of a text into a value of type Text. Specify a character set, such as `UTF8` as an argument.
binary(text: Text, charsetName: Text)	Binary Converts a text value into a binary value. Specify a character set, such as `UTF8` as an argument.
toJSON()	Converts any type of value except Password to a text value formatted as JSON object. Record types that contain a Password type attribute cannot be converted to JSON. **Converted value examples** • 5 becomes "5" • 1.2 becomes "1.2" • true becomes "true" • "Hello" becomes "\"Hello\"" • A binary value becomes a base 64 encoding of the binary value • A record value, R(a = 5, b = true, t = "Hello") becomes "{\"a\":5,\"b\":true,\"t\":\"Hello\"" **Note** Workflow State view does not show reverse slashes before quote marks. The way the converted record value is shown above is how you have to write it in an expression. toJSON function cannot generate an array, such as [1, 2, 3]. When you use a JSON value in a Desktop Automation robot, you can create the list yourself by string concatenation. This way you can return values from the Desktop Automation robot the Web Automation robot. See Work with JSON for more information.

The following table lists and describes examples of conversion functions written in two ways: direct function call such as text(2), and method call such as 2.text(). While in the first variant the function is called directly, the second variant enables you to use text completion when writing expressions in the expression mode. For more information, see **Expression Mode** later in this section.

Examples	
Function	**Evaluates to the following result**
ampersandEncode("<b/>") or "<b/>".ampersandEncode()	&lt;b/&gt;
ampersandDecode("&lt;b/&gt;") or "&lt;b/&gt;". ampersandDecode()	<b/>

Examples	
Function	**Evaluates to the following result**
base64Encode(bin) or bin.base64Encode() where bin is a variable containing a Binary value resulting from "Hello".binary("ASCII")	SGVsbG8=
base64Decode("SGVsbG8=").text("UTF8") or "SGVsbG8=".base64Decode().text("UTF8")	Hello
• "true".boolean() • "false".boolean() • "False".boolean()	• Boolean true • Boolean false • Throws a ConversionIssue exception as it does not must match neither "true" nor "false"
• integer(2.0) or 2.0.integer() • integer("2") or "2".integer	2
• number(2) or 2.number() • number("2") or "2".number()	2.0
number(".1E10") or ".1E10".number()	1.0E9
"Hello".binary("UTF8").text("UFT8") where "Hello".binary("UTF8") evaluates to a binary value encoded using the UTF8 character encoding. When the binary value is converted to text, as in this example, the binary value must be encoded using the same character encoding (UTF8 in this case).	Hello
"hello".toJSON()	"hello"
17.7.toJSON()	17.7
true.toJSON()	true

Company.toJSON()
where Company is a variable of a record type containing Text, Integer, and Boolean field types.

Variables
company: Company
name = "Acme Corp."
url = "www.acme-corp.com"
revenue = 1000000000
CEO = "Marvin Acme"
fictional = true
	a Text value with the following value:
	{"name":"Acme Corp.", "url":"www.acme-corp.com", "revenue":1000000000, "CEO":"Marvin Acme", "fictional":true}

Limits for number values

Integers

The largest integer that can be represented is 1E34-1. This is a number with 34 nines. If you have this number and add 1 to it, you get an `OverflowIssue` exception. Likewise the smallest integer value that can be represented is -1E34+1. So all integer values must be in the interval from -1E34+1 and to 1E34-1 (both included).

Numbers
Representation of numbers matches the IEEE 754R Decimal128 that uses 34 decimal digits and a representation of the exponent in the range of −2147483648 to +2147483648. If evaluation of an expression leads to a number outside the range, you get an `OverflowIssue` exception.

See Limits in Numbers for more information.

It is possible to enter numbers with an exponent as follows:

1.234E5, 0.1E-2, 1000.0e42, etc.

A base number is optionally followed by an `E` or `e` and an integer exponent. When numbers are displayed in the Desktop Automation robot State view or in the Expression editor, they are normalized according to the following rules:

- One trailing zero is shown after the decimal point if there are no other decimal digits, such as 1.0E12
- The exponent is only shown if it is numerically larger than or equal to 9, that is:
- - 1.0E8 is shown as 100000000, but 1.0E9 is shown as 1.0E9
  - 1.0E-8 is shown as .000000001, but 1.0E-9 is shown as 1.0E-9
- All digits up to the maximum number of 34 are shown (numbers are rounded off to have a maximum of 34 digits). For example, number 9.999999999999999999999999999999999 is shown as is. If you add one more 9 to this number, it is shown as 10.0. Actually the number value is rounded off so that internally it is also represented as 10.0. Therefore, the numbers can be entered in one way and displayed in another way, which adds some flexibility in how numbers may be entered, such as 10.0 or 0.1E2.

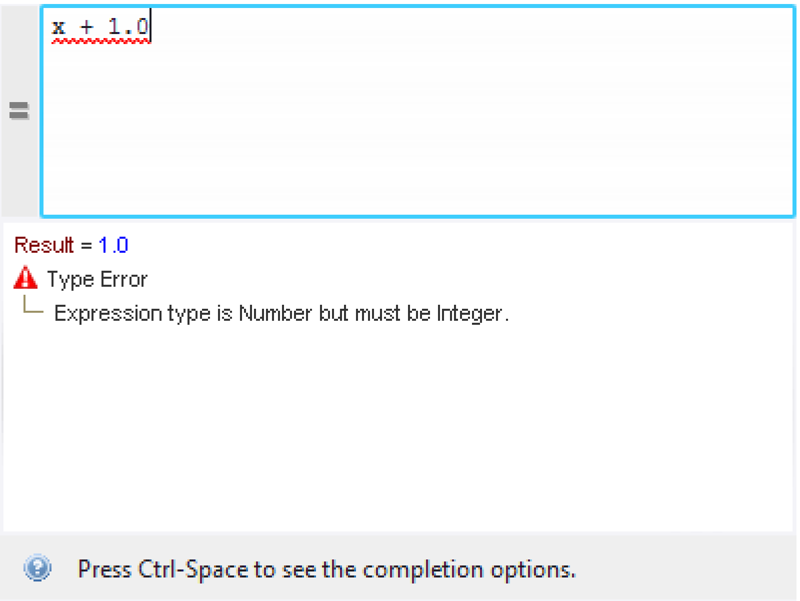
The conversion function `text(Number)` also returns the number written according to the above rules.

Expression Editor

The Expression Editor is an interactive editor that opens when you click an input field in the Desktop Automation robot and if the given field supports entering an expression. The editor consists of two horizontal panes. The upper pane is the input pane to enter and edit the expression and the lower pane is the result pane. The result pane can show the result of evaluating the expression, an error message, or both. The following errors can be shown:

- Parse errors: The syntax of the expression is incorrect.
- Type errors: There is a type error in the expression or the result does not have the correct type.
- Evaluation errors: Some error occurred while evaluating the expression, such as division by zero.

In the following example, the expression is `x + 1.0` where `x` is a variable of Integer type. The expression is correctly typed, but as the result is assigned to the variable of Integer type, an error message shows that the result type of the expression is not the expected one.



```
x + 1.0
```

Result = 1.0

Type Error

Expression type is Number but must be Integer.

Press Ctrl-Space to see the completion options.

You can copy the result of the expression and the error message from the lower pane of the editor by right clicking it and selecting **Copy Value**. If the value is a record type, the result is shown as a tree and each attribute value can be copied separately. Password and Binary values cannot be copied.

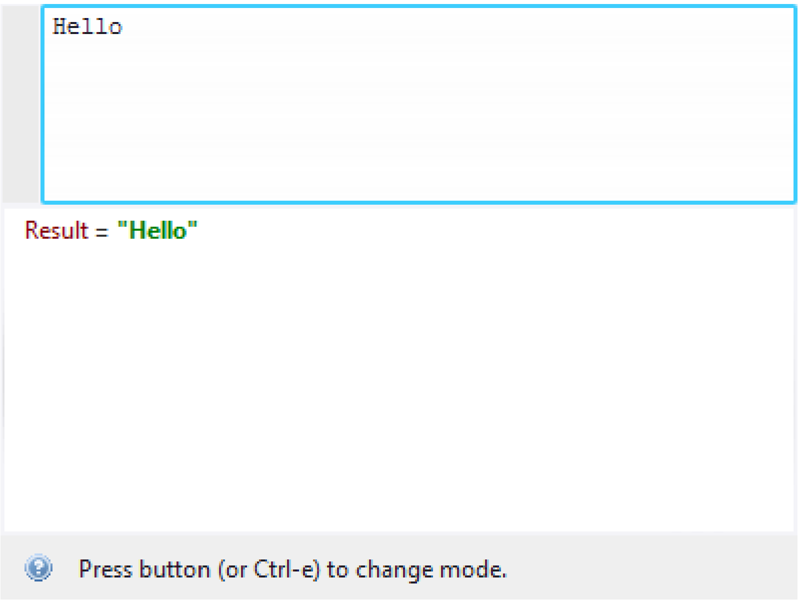
To close the Expression Editor, click outside the editor or press Esc.

Editor Modes

The editor has a mode button to the left of the upper pane that switches between an expression and a value mode. The editor in the figure above is in the Expression mode. When the editor is in the Value mode, the mode button is blank. When the editor is in the expression mode, the button shows an equals sign (=). You can use the keyboard shortcut Ctrl-E to switch between the two modes.

Value Mode

In the value mode, the entered value is simply interpreted as a value, such as a Text, a Boolean, a Number, etc. and no evaluation takes place. The result panel shows the result. The only error that can be shown is when the result type is incorrect.



Hello

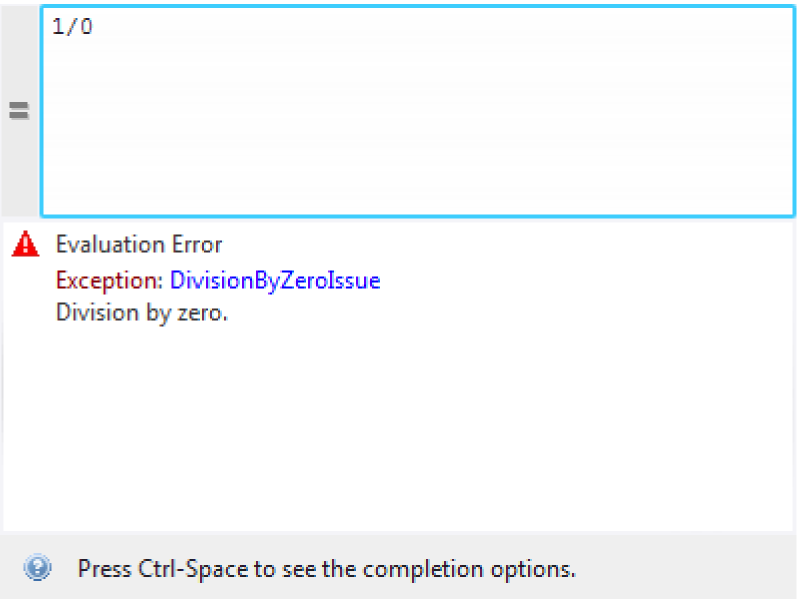
Result = "Hello"

Press button (or Ctrl-e) to change mode.

Expression Mode

In the expression mode, everything you type in the input pane is interpreted as an expression and checked for syntax and type errors. If you are editing an expression at the current flow point and there are no errors, it is evaluated and its result is displayed. The evaluation happens while you are typing so that you always know what the state of your expression is. If you are editing an expression that is not at the current flow point, it is evaluated if it does not contain any variables, that is if its value does not depend on the current state.

If there is an error during evaluation of an expression, for example divide by zero, the Result pane reports this by showing the name of the exception and a message describing the issue. You can copy the name of the exception by right-clicking it and selecting **Copy Value**.



```
1/0
```

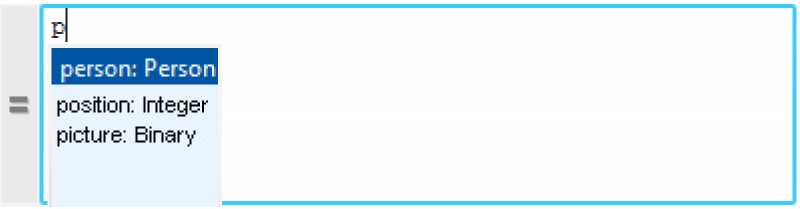
Evaluation Error

Exception: DivisionByZeroIssue

Division by zero.

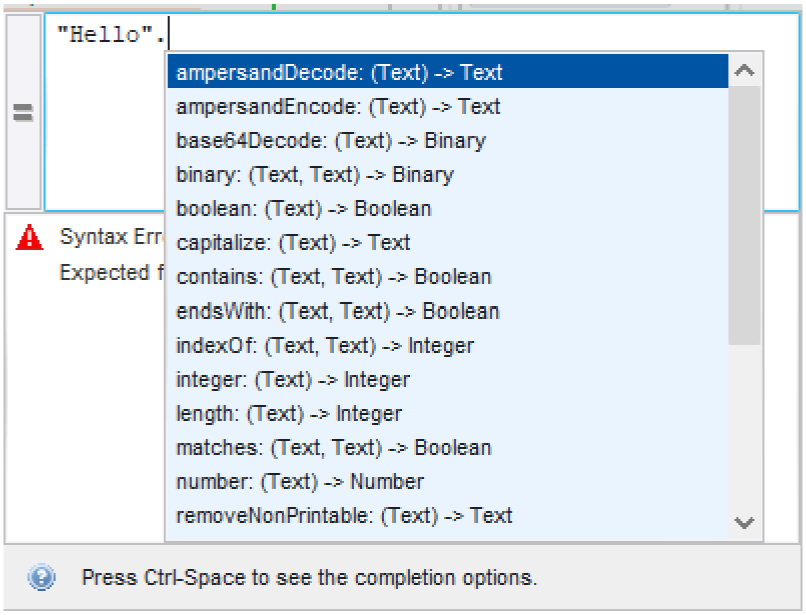
Press Ctrl-Space to see the completion options.

Text completion in the expression mode helps entering names of variables, field names, and function names. The text completion window automatically appears when you type something that has a completion help. For example, if you start typing a variable name and there is a variable starting with what you have already typed, the completion window appears. If you press a dot (.) after a variable of record type, the completion window shows a list of completion options corresponding to the fields of the record type. The following example shows completion help after typing "p".



```
p
```

If you press a dot (.) after a sub-expression of simple type, the completion window shows a list of completion options corresponding to the function for which the first argument has the same simple type.



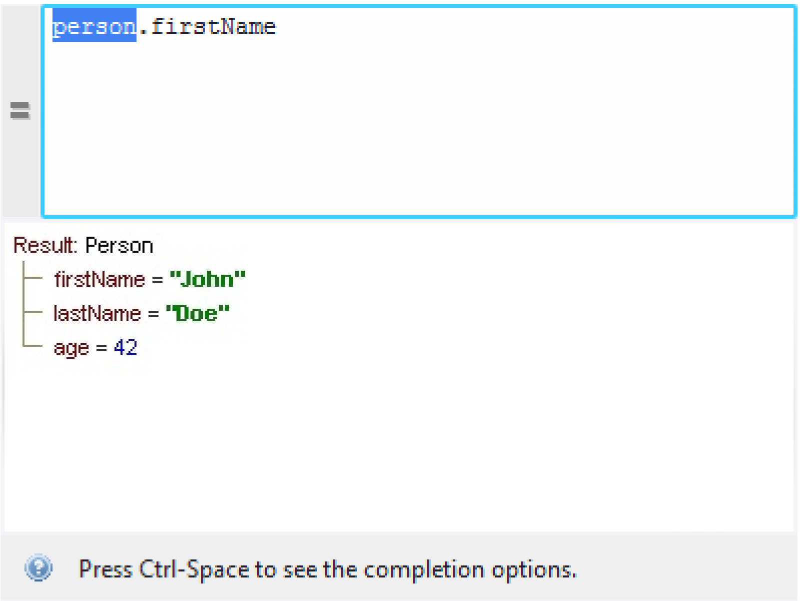
```
"Hello".
```

ampersandDecode: (Text) -> Text
ampersandEncode: (Text) -> Text
base64Decode: (Text) -> Binary
binary: (Text, Text) -> Binary
boolean: (Text) -> Boolean
capitalize: (Text) -> Text
contains: (Text, Text) -> Boolean
endsWith: (Text, Text) -> Boolean
indexOf: (Text, Text) -> Integer
integer: (Text) -> Integer
length: (Text) -> Integer
matches: (Text, Text) -> Boolean
number: (Text) -> Number
removeNonPrintable: (Text) -> Text
Syntax Err
Expected f

Press Ctrl-Space to see the completion options.

To navigate the options in the completion list, use the arrow keys or a mouse. To select an option in the completion list, press Enter, Tab, or double-click the option. To open the completion window without typing anything, press Ctrl+Space. To close the completion window, click outside the window or press Esc.

If a part of an expression is selected in the input pane and this selection corresponds to a proper sub-expression (one that may be evaluated on its own), the value of this sub-expression is shown in the result pane as in the following figure.



```
person.firstName
```

Result: Person
- firstName = "John"
- lastName = "Doe"
- age = 42

Press Ctrl-Space to see the completion options.

If you select the name of a function in the Input pane, a description of this function is shown in the Result pane as shown below.

Limits in Numbers

Desktop Automation and Web Automation robots have different number formats. In Web Automation robots, the value stored in a variable is a double (binary64 as specified by the IEEE 754 standard). In Desktop Automation, variables store numbers in IEEE 754R decimal128 format, which uses 34 decimal digits and an exponent range of −2147483647 to +2147483648 (= 2^31). When storing values in a Desktop Automation robot, rounding may occur and you can expect some loss of precision. For example, if a number has more than 34 digits and the last one is `.5`, it is rounded off, so the `.5` becomes `.0`. For example, 1234567890123456789012345678901234567890.0 becomes 1234567890123456789012345678901235000000. Integers can get large, but the number of significant figures can only be 34 digits.

You can use numbers up to: 9.9999999999…E2147483647 and numbers as small as 0.1E-2147483646. They are the largest and smallest numbers that you can convert from Text to Number, such as by writing `“0.1E-2147483646”.number()` in an expression. You can get higher numbers using multiplication, but it might cause an overflow error. The limits for our representation of Numbers gives you the following:

`“9.9999999999…9E2147483647”.number()` converts to 1.0E2147483648 if there are more than 34 digits in the number, but 9.9999999999…9E2147483647 if there are fewer than 34 digits.

`“0.1E-2147483646”.number()` converts to 1.0E-2147483647.

Use RDP Connection

You can connect to devices over an RDP connection if session management via Remote Desktop Sessions is preferred to normal login sessions.

Prerequisites
To use RDP your environment must comply with the requirements for the Lock Screen feature.

Steps to use RDP

1. Install Desktop Automation Service on remote devices. Configure the service to work in single user mode and specify a token. See Configure the Desktop Automation Agent.
2. In the Desktop Automation robot, insert the Initiate Session step.
3. Specify all necessary options for the RDP connection.

   The Initiate Session step waits until the connection is established before continuing robot execution. If remote connection fails, an error message is provided.

If the system you are connecting to is configured to show an extra screen when the user logs in, set a number of seconds to wait before dismissing the extra screen during login in the "Logon dialog dismissal timeout" option of the Initiate Session step. If this screen is not dismissed, the action may fail.

Keep RDP alive
Windows may be configured to disconnect and terminate RDP sessions that are idle for too long. To prevent from disconnecting, the RDP connection service in the robot sends a dummy keystroke signal to the remote session every several seconds. The default value for the time interval between the key strokes is 30 seconds. To change the time interval, specify it in the **Keep-awake key press interval** option of the Initiate Session step in seconds. To turn off this option, specify zero (0).

> **Note** Only one RDP connection to a specific device can exist at a time, but you can have several RDP connections to different devices. A new RDP connection to the same device closes any existing RDP connection on this device.

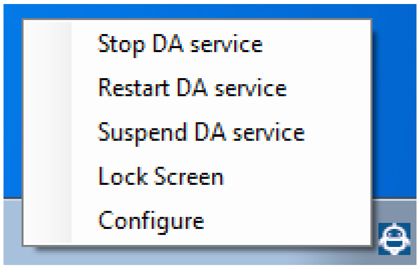
If the device is a static reference, you can see available applications in the Recorder view after RDP connection is established.

If the device is a dynamic reference, a Connect To Device is needed to automate the remote device. We recommend setting up timeout and some retry attempts to make sure the RDP connection is established. Once this step is executed, you should see available applications in the Recorder view.

> **Note** To check that the RDP connection is established, look for the `kapowlock` process in the Windows Task Manager. If the process is present in the list of processes on the computer executing your robot, the connection is active.

Manage Remote Desktop

You can perform the following actions using the Desktop Automation Service shortcut menu.



Manage the Desktop Automation Service

The following commands help you manage the Desktop Automation Service running on a remote computer.

- **Stop DA service**: Stops the service, which makes the remote device unavailable. The computer running the Desktop Automation Service is removed from the list in the Management Console.
- **Restart DA service**: Stops and starts the service. A robot or Design Studio loses the connection to the device and must be reloaded to restore it.
- **Suspend DA service**: Suspends the device. If suspended, the service is displayed as suspended in the Management Console. To restore the service operation, a user or an administrator needs to manually start the Desktop Automation Service on the device.

  The suspended state makes the DA service unavailable for robots to use, but the state information is send to the Management Console via the ping mechanism and the devices is displayed in the **Admin** > **Devices** section. This command is useful if for some reason the service or the computer running it needs some configuration changes.
- **Lock Screen**: Locks the screen on the remote device.
- **Configure**: Opens the Desktop Automation Service configuration dialog box.

Chapter 6

Management Console

Management Console is an application with a web-based interface providing monitoring and management capabilities for Kofax RPA.

In Management Console, you can, among other things:

- Enable collaboration and sharing using the repository
- Centrally administer users, user roles, and permissions
- Manage license information and passwords
- Deploy robots from Design Studio to the repository
- Schedule robots from the repository
- Browse and export extracted data
- Access detailed logs of production results and errors
- Configure clusters of multiple RoboServers
- Connect to other Kofax products and services

Introduction to Management Console

Management Console is an application with a web-based user interface, which makes it easy to access from any computer on the network. The address to connect to a Management Console consists of a computer name and a port, such as `http://computername:port`.

Management Console can be deployed in two different ways.

- **Embedded mode**: A way to quicky start Management Console after installation for demonstration and testing purposes. See Configure Embedded Management Console below for information on the embedded mode.
- **Standalone mode**: A standalone version of Management Console deployed on Tomcat in a production environment. For this mode, Kofax RPA also provides tools for Docker deployment. For information about a standalone Management Console, see "Tomcat Management Console" in the *Kofax RPA Administrator's Guide*.

Starting from Kofax RPA version 10, all RoboServers must auto register to Management Console. Therefore, the URL and credentials for the Management Console along with the cluster name must be specified when starting the RoboServer (either at the command line or on the **General** tab of the RoboServer Settings application). See Start RoboServer for more information.

Note Some features such as high availability may not be available, depending on your license key.

Configure Embedded Management Console

In embedded mode, Management Console and RoboServer start simultaneously and by default, the web server interacts with port 50080. The web interface port and other parameters are configured on the **Management Console** tab in the RoboServer Settings application.

Protocols and Ports

You can configure the web server to be accessible through HTTP and HTTPS on separate ports. If a protocol is enabled, a port number must be chosen; the default ports are 50080 (HTTP) and 50443 (HTTPS).

To enable HTTPS, a server certificate in JKS format must be stored in a file called `webserver.keystore`.

You can also restrict who is allowed to upload JDBC driver to the embedded Management Console. Possible choices are "**Not Allowed**", where no one can upload JDBC drivers, "Admin from localhost," which means that the administrator can upload drivers when accessing the Management Console from the local machine; and finally, "Admin from any host," which means the administrator can always upload JDBC drivers.

User Management

One of the points of having a Management Console is to coordinate execution of robots, and thus it is typically accessed by many clients. To mitigate the potential security risk of having an unauthorized access to a Management Console from other machines, user management is enabled by default in embedded mode and the default admin superuser password is available (user name - `admin`, password - `admin`). See Users & groups for more information.

When you log in to a Management Console for the first time, use the default `admin` user password. Once you are logged in, you can change the password and create other users and groups on the **Users & groups** page under **Admin** in the Management Console. Use these credentials both when you publish a robot to the Management Console from Design Studio and when you access the web interface from a browser. To change the `admin` user password, do the following:

1. Expand the **Admin** item on the left pane and click **Users & groups**.
2. On the **Users** tab, select the `admin` user and click 👤 above the tab.
3. Type the new password, type it again to confirm, and click **OK**.

   See Users & groups for more information.

Start Embedded Management Console

Before starting a Management Console, perform the following:

1. Start the RoboServer Settings application from the Start menu.

2. On the **General** tab, select **Register to a Management Console**, and supply all necessary information including the superuser name and password to connect to the Management Console. The default superuser name and password:
   - User name: `admin`
   - Password: `admin`

Note For production scenarios, we recommend that you set up a Tomcat version of the Management Console. For information on starting and using a Tomcat version of the Management Console, see the *Kofax RPA Administrator's Guide*.

In embedded mode, start Management Console as follows.

Windows

Use the **Start Management Console** item on the Start menu.

To start the Management Console from the command line, run the following command in the `bin` subfolder of the installation folder.

```
RoboServer.exe -p 50000 -MC
```

Linux

Start Management Console from the command line. It is part of the RoboServer program, which is found in the bin directory under the installation directory.

```
$./RoboServer -p 50000 -MC
```

This starts a RoboServer listening on a socket at port 50000, and providing Management Console functionality via a web interface on a configured port (the port is configured in the RoboServer Settings application on the **Management Console** tab). See the Configure Embedded Management Console for details.

You can also use the command line to start a RoboServer and register it to a Management Console:

`RoboServer.exe -p 50000 -mcUrl http://username:password@ServerName:port -cl "Production"` command starts a RoboServer on port `50000` and registers it to the Management Console at `ServerName:port` under the `Production` cluster with the specified user name and password.

Once the Management Console starts, open it in a browser. On Windows, click the Management Console item on the Start menu. On all platforms, you can open a browser and go to `http://computername:port/`. When started for the first time, log in using the default credentials, accept the license terms, and enter your license information, including your license keys. If you need to change the license information later, you can do so in **Admin** > **License**.

After setting up the Management Console, you can start as many RoboServers as you like using the parameter `-p 50000`. These are the only RoboServers that should be added to the RoboServers section.

Management Console Configuration and User Interface

Management Console is divided into six functional areas.

Schedules

Use this menu to create and manage schedules. A schedule is a plan for running one or more robots, typically at pre-planned points in time and in a repeating fashion. A schedule provides the plan for when the robots should be run, which is done by passing them to the configured servers.

Repository

Use this menu to manage your work objects, connect to other components by means of mappings, add Robot File Systems, and other. Robots, type definitions, and resources can be uploaded from Design Studio to the Management Console repository or uploaded manually through the web interface of the Management Console. Uploaded robots can be executed as part of a schedule, Kapplet, or through client code that executes robots using the Kofax RPA Java or .NET APIs. You can also use the APIs to programmatically query or update the repository.

Data view

Use this menu to see data your robots have stored in databases or to export this data to Excel, XML, or CSV.

Log view

Use this menu to check the RoboServer, schedule, robot, and Desktop Automation logs.

Admin

Use this menu to configure settings for Management Console related to administrative tasks. It also enables you to manage clusters of RoboServers and their settings, as well as manage projects and permissions. This is also where you configure the license and create/restore backups and projects.

Settings

Use this menu to configure other settings for Management Console, such as related to server and database configuration, Process Discovery, and KTA.

Most sections in Management Console display items of a certain type in a table. When many items are available for display, they are divided into pages. You can use the **Items per page** setting to select how many items to display at the same time. This number does not adapt automatically to the height of the browser window, so the empty space below the table does not always mean that you are seeing the last items. To select columns shown on the page, click ▥ on the right and select columns in the list.

The tables support sorting and filtering. Click any column heading to sort by that column. Click the same column heading again to reverse the sorting order. Sorting is performed on the entire table, not on each page individually. If you have more than one "page," you may see a completely different set of rows if you change the sorting order. Depending on the opened page you can filter the list by typing a name in the **Filter** field or first click **Filter**, select an item to filter by, and type its name. See Filtering for details.

User menu

In the upper right corner, the user menu is located containing the following functions:

- **Profile**

  Contains general information about the current user.

- **Settings**
  Contains the user interface language setting and the option to reset all custom settings to their default values. Note that local user interface language settings have a higher priority than those assigned by the administrator.
- **About**
  Contains general information about this version of Management Console, server time, and local time.
- **Log out**
  Click to log out from the current user profile.

Additionally, a help button is available to the left of the user menu to open *Help for Kofax RPA*.

Schedules

The Schedules menu enables you to manage schedules in Management Console. A schedule denotes a selection of robots and plans for running them. Running the schedule means running the selected robots (in parallel or sequence), optionally executing pre- and post-run robots.

> **Important** When selecting the interval between schedule runs, if a schedule run time comes when it is still running, this schedule will not start.

At the top of the Schedules menu, in the Projects drop-down list, you can select the project with schedules to display. You can change the way the information for each schedule is presented as follows:

- Filter the list of schedules in the table by typing a name or a part of the name in the Filter text field. See Filtering for more information.
- Select the table columns to display for a schedule using the ▥ menu icon on the right.
- Refresh the displayed information by clicking the ↻ refresh icon on the right.

By default, the following table columns are displayed for each schedule.

Column	Description
Active	When active, the schedule runs as planned. You may need to make a schedule inactive for several reasons, such as: • Because the function performed by the schedule currently is not needed. • Because errors have been found in the robots and you do not want the schedule to run before you fix these errors. • Because you want to trigger the schedule manually each time it should run. This may be appropriate for some robots and schedules, such as for preparation or cleanup tasks.
Name	Name of the schedule.
Project name	Name of the project that the schedule belongs to (useful when viewing all projects).
Job count	Combination of total and active jobs. If all jobs are active, it simply lists the number of active jobs; if two of three jobs are active, it lists 2 (3).
Next run	Time when the schedule is planned to run next time.
Previous run	Time when the schedule was last run.
Interval	Planned interval between two consecutive runs of the schedule.

Column	Description
Total runs	How many times the schedule has been run.
Errors	Number of schedule errors during the last run of the schedule. Schedule errors do not include robot errors. **Note** To see errors and robot errors, make sure that the log database is set up on the General tab in Settings->RoboServer log database and that database logging is enabled on the RoboServers in RoboServers->Settings->Logging.
Robot errors	Number of robot errors that occurred in robots run by this schedule. **Note** When the RoboServer works with a scheduled robot, and the robot reports an error, the scheduler and RoboServer are not stopped as the RoboServer tries to recover from the error.
Optional Columns	
Total jobs	Total number of jobs in the schedule.
Active jobs	Number of active jobs in the schedule.
Created by	Name of the user who created this schedule.
Modified by	Name of the user who last modified this schedule.
Commit message	Summary describing the commit.
Revision number	Number of the schedule revision.
Cluster	Cluster that the schedule executes on.

Create New Schedule

1. To create a new schedule, click the plus sign in the upper left corner.

   The "Select project" dialog box appears.

2. Select a project and click **OK**.

   Several new tabs open.

3. The **Basic** tab contains everything necessary for setting up a schedule.
   - Name: Name of the schedule.
   - Active: Select this option to make this schedule active.
   - Simple / Cron: Select a way of defining the time plan for a schedule.
   - Every: Available only for simple schedules. Desired time interval between two consecutive runs of the schedule. It is entered as an integral number with a unit, such as "1 minute" or "3 hours."
   - Pattern: Available only for cron schedules. Pattern defining when the schedule should be run.
   - Start time: Local time when the schedule must start.
   - Jobs priority: Execution priority for this schedule jobs over the other schedule jobs queuing to be given access to a certain resource. The priority applies to all robot jobs in the schedule. For more information, see Queuing of schedule jobs.
   - Jobs timeout: Timeout for the queuing of schedule jobs. If a schedule job has not been given access to a certain resource and executed by the time the timeout is reached, the schedule job

stops queuing. The timeout applies to all robot jobs in the schedule. For more information, see Queuing of schedule jobs.

- Pre-processing robot: Name of a robot that will be run before the schedule is started.
- Post-processing robot: Name of a robot that will be run after all other robots in the schedule are run.
- Run on cluster: Name of the cluster to run this schedule.
- Commit message: Summary describing the commit.

4. On the **Advanced** tab, you can configure runtime constraints.
   - Execution time limit: Set the maximum execution time for each robot in the schedule. When a robot has executed for this period of time, the server stops it, and an error is logged. The default value is `-1`, which means the time is not limited.
   - Extracted values limit: Select the maximum number of values each robot may output. If the robot produces more than this number of values, the server stops it, and an error is logged. The default value is `-1`, which means the number is not limited.
   - Run jobs sequentially: Select to execute robots in the order listed on the Jobs in schedule tab.
   - Use email notification: Check to receive an email whenever a robot fails. If several robots in a schedule fail, you will get one email for each robot each time the schedule runs. Email notification works only if you configure an SMTP server and enter the desired email addresses in the Email addresses field.
   - Email addresses: Comma-separated list of email addresses to which notifications are sent. The first listed address is used as both a sender and receiver address. The field can contain up to 255 characters. If you exceed the number of characters in this field, the schedule is not saved.
5. On the **Jobs in schedule** tab, you see the list of jobs that will run when the schedule is triggered.
   - Job name: Display name of the job. This is the name specified in the display name attribute when uploading the robot.
   - Active: When Active, the job runs when the schedule is run. You may want to make a single job within a schedule inactive for several reasons, such as:
     - Because the function performed by the job is currently not needed.
     - Because errors have been found in the robots and you do not want the schedule to run before you have fixed these errors.
     - Because you want to trigger the job manually each time it should run.

   Add Jobs to Schedule

   To add jobs to a schedule, do the following.

   a. Click the plus sign in the upper left corner.

      A dialog box appears to guide you through the job creation.

   b. In the **Select job type** step, select an option. Available options include:
      - A single robot
      - A group of robots based on name

Job Type	Description
Single robot	Adds a job that runs a single robot. You can add several robots one by one. To pass input to a robot, you can specify the necessary parameters.

Job Type	Description
Group of robots	Adds a job that runs any number of robots which path name matches a given criteria. The robot groups are evaluated every time the schedule starts; therefore, new robots matching the selected criteria run automatically.

Actions on Schedules

When you click a ⋮ context menu for a robot, the following actions are listed:

- **Create copy**: Opens the wizard for schedule creation. Specify a new name, make any changes to the schedule, and click **OK**.
- **Deactivate**/**Activate**: Turns off/turns on the schedule.
- **Run**/**Stop**: Click to manually run the schedule. This is especially useful for inactive schedules. If the schedule is already running, the Stop function is active and the schedule is stopped when you click it. That is, all of its running robots are stopped as quickly as possible. The schedule is displayed as "running" until all of its robots stop executing.
- **Edit**: Opens the same wizard as for schedule creation with filled parameters.
- **View**: Multiple view functions are available to see information related to schedule runs as also appears in the Log view.

Alternative schedule creation

It is also possible to create a schedule from the Robots section. This is done by selecting any number of robots, right-clicking and choosing **Create schedule** from the context menu.

Add a single robot

The wizard for adding a single robot contains one to four steps, depending on the robot you select.

1. In the **Select robot** step, specify a display name and select a robot in the drop-down list.
2. If all snippets and types used by the robot are already uploaded, and the robot does not have any input variables, click **Next** and then click **Save**.

   > **Note** Ensure that all snippets and/or type files associated with this robot are present in the Management Console repository. If not, make sure you upload them now.
   >
   > In robots added to schedules, only default values of input types are uploaded. Any changes to default type values in variables made in Design Studio can be used only in Design Studio. Check the values of variable types and change if needed when uploading robots to schedules.

3. Configure the robot input used when it runs as part of this schedule.
4. If an attribute is of a binary, image, PDF, or Excel type, you can use the drop-down list to select a resource that is already uploaded, or click **Upload** to upload one.

   If an attribute is required, it is marked red.

5. Complete all required fields and click **Save**.

   Additionally, you can perform the following actions on jobs from the ⋮ context menu:

   - **Move up**/**Move down**: Moves the selected job in the list.
   - **Create copy**: Creates a copy of the existing job.
   - **Edit**: Edits the job.
   - **Deactivate**/**Activate**: Turns off/turns on the job.

   To delete a job, select it in the table and click the 🗑 bin icon in the upper left corner.

Add a group of robots

You can select to add a single job that runs all robots with path names that match the given criteria. The criteria is evaluated when the schedule is run, so any robots uploaded after the job is created are included if they match the configured criteria.

Because the criteria is evaluated at runtime, any errors such as missing snippets/types are logged as errors in the Schedule runs log in the Log view. The same is true if a robot that uses input variables is run because it matches the criteria.

Note In robots added to schedules, only default values of input types are uploaded. Any changes to default type values in variables made in Design Studio can be used only in Design Studio. Check the values of variable types and change if needed when uploading robots to schedules.

1. In the **Configure criteria** step, specify a display name and select the matching criteria: "Robot path contains," "Robot path matches pattern," or "Robot path starts with."

   The list at the bottom displays one or more robots matching the selected criteria. As long as you do not edit the **Display name** field, it changes depending on your criteria selection, but once you start editing it, the automatic naming is disabled.
2. Specify a pattern to match against.
3. Click **Next**.

   You can click **Next** even if no robots match the configured criteria, as you can later upload a robot that matches.

   If you have many robots, it may take some time to refresh the list of robots matching the selected criteria.

   Note When this job type is added to a schedule configured to run its jobs sequentially, you cannot control the order within the group of robots matching the criteria (but they are executed sequentially).

Cron schedule

A "cron" schedule is made up of six or seven sub-fields that together describe when the robots should be run. The sub-fields are separated with one or more spaces and represent:

1. Seconds
2. Minutes
3. Hours

4. Day-of-Month
5. Month
6. Day-of-Week
7. Year (optional field)

An example of a complete cron schedule is "0 0 12 ? * WED", which means "every Wednesday at 12:00 pm".

Individual sub-fields can contain ranges and/or lists. For example, the Day-of-Week field in the previous example (that is, "WED") could be replaced with "MON-FRI", "MON, WED, FRI", or even "MON-WED,SAT".

Wildcards (the "*" character) can also be used, meaning "every possible value of this field". For example, the "*" character in the Month field of the previous example would mean "every month". A "*" in the Day-of-Week field would mean "every day of the week".

The set of valid values for each field is:

- Seconds: The numbers 0 to 59.
- Minutes: The numbers 0 to 59.
- Hours: The numbers 0 to 23.
- Day-of-month: The numbers 1 to 31 (take into account many days a given month has).
- Month: The numbers 1 to 12, or the strings JAN, FEB, MAR, APR, MAY, JUN, JUL, AUG, SEP, OCT, NOV, and DEC.
- Day-of-week: The number 1 to 7 (1 is Sunday), or the strings SUN, MON, TUE, WED, THU, FRI, and SAT.
- Year: Any number.

In addition, a number of special characters may be used:

	Definition
/	Specifies increments to values. For example, if you put "0/15" in the Minutes field, it means "every 15 minutes, starting at minute zero". If you use "3/20" in the Minutes field, it means "every 20 minutes during the hour, starting at minute three". In other words, it is the same as specifying "3,23,43".
?	Allowed only in the Day-of-Month and Day-of-Week fields and means "no specific value". This is useful when you need to specify something in one of these two fields, but not the other. See the examples below for clarification.
L	Allowed only in the Day-of-Month and Day-of-Week fields. "L" is short-hand for "last", but its meaning differs between the two fields. In the Day-of-Month field, "L" means "the last day of the month" - day 31 for January, day 28 for February on non-leap years, and so on. If used in the Day-of-Week field by itself, it means "7" or "SAT", but if used in the Day-of-Week field after another value, it means "the last xxx day of the month" . For example, "6L" or "FRIL" both mean "the last Friday of the month". When using the "L" character, it is important not to specify lists or ranges of values, as you will get confusing results.
#	Used in the Day-of-Week field to specify "the nth" XXX weekday of the month. For example, the value of "6#3" or "FRI#3" means "the third Friday of the month".

	Definition
W	Specifies the weekday (Monday-Friday) nearest the given day. As an example, if "15W" is specified as the value for the Day-of-Month field, the meaning is: "the nearest weekday to the 15th of the month".

Examples

Cron schedule	Explanation
0 0 12 * * ?	Fire at 12pm (noon) every day
0 15 10 ? * *	Fire at 10:15am every day
0 15 10 * * ?	Fire at 10:15am every day
0 15 10 * * ? *	Fire at 10:15am every day
0 15 10 * * ? 2005	Fire at 10:15am every day during the year 2005
0 * 14 * * ?	Fire every minute starting at 2pm and ending at 2:59pm, every day
0 0/5 14 * * ?	Fire every 5 minutes starting at 2pm and ending at 2:55pm, every day
0 0/5 14,18 * * ?	Fire every 5 minutes starting at 2pm and ending at 2:55pm, AND fire every 5 minutes starting at 6pm and ending at 6:55pm, every day
0 0-5 14 * * ?	Fire every minute starting at 2pm and ending at 2:05pm, every day
0 10,44 14 ? 3 WED	Fire at 2:10pm and at 2:44pm every Wednesday in the month of March.
0 15 10 ? * MON-FRI	Fire at 10:15am every Monday, Tuesday, Wednesday, Thursday and Friday
0 15 10 15 * ?	Fire at 10:15am on the 15th day of every month
0 15 10 L * ?	Fire at 10:15am on the last day of every month
0 15 10 ? * 6L	Fire at 10:15am on the last Friday of every month
0 15 10 ? * 6L	Fire at 10:15am on the last Friday of every month
0 15 10 ? * 6L 2002-2005	Fire at 10:15am on every last Friday of every month during the years 2002, 2003, 2004 and 2005
0 15 10 ? * 6#3	Fire at 10:15am on the third Friday of every month

Queuing of schedule jobs

When creating a schedule, you schedule jobs to run in a queue upon availability of required resources, such as Desktop Automation Service, license units, or RoboServer execution slots.

During the schedule creation, set the **Jobs priority** parameter to the most suitable priority level: Minimum, Low, Medium, High, or Maximum. Jobs from schedules that have a higher priority are provided access to the required resources and are executed sooner than those having a lower priority. Note that jobs with a certain priority that have been queuing for some time will be executed sooner than the newer jobs with the same priority. For example, a high priority job that has been queuing for several minutes will be executed (provided access to the needed resource) sooner than another high priority job that has just entered the queue.

Also, you need to set the **Jobs timeout** parameter to determine when the schedule jobs are to stop queuing. If a schedule job has not been given access to the required resource and executed by the time the timeout is reached, the job stops queuing.

The two parameters apply to all robot jobs in the schedule.

You can observe the status and history of queuing tasks in the Task view and using the **Task messages** log in the Log view.

Repository

Management Console keeps a repository of robots, types, snippets, resources, device mappings, OAuth credentials, and other work objects. The Repository menu helps you manage your objects.

Robots

The Robots section lists robots uploaded to the Management Console repository and helps you manage robots in the repository on a per project basis.

To run in a schedule, robots need to be uploaded to the Management Console repository from Design Studio using the Upload function or directly from Management Console using the "Upload a robot" button. When the robot is uploaded, it is copied into the repository. Thus, if changes are later made to the robot in Design Studio, it needs to be uploaded again. Schedules that the robot is associated with will use the new version of the robot the next time they run.

Each robot belongs to a project. Within a given project, you cannot have two different robots with the same name in the repository. Different projects can contain robots with the same name. In Design Studio, you can assign tags to your robots that can be the same and can be used to filter the list in Management Console.

At the top of the Robots section, in the Projects drop-down list, you can select the project with robots to display. You can change the way the information for each robot is presented as follows:

- Filter the list of robots in the table by typing a robot name or a tag name in the Filter text field. See Filtering for more information.
- Select the table columns to display for a robot using the ▥ menu icon on the right.
- Refresh the displayed information by clicking the ↻ refresh icon on the right.

By default, the following table columns are displayed for each robot.

Column	Description
Folder	Name of the folder specified to store the robot files. By default, the files are stored in the Root folder. You can create a new folder to store the files, and the folder name must be unique. The folder name is displayed in the column if the selected folder is other than Root. Folders are also shown as part of the robot name in the robot runs log in the Log view. When deleting a robot, you can delete an empty folder.
Name	Name of the robot. If the robot uses a type or a snippet that is not present in the repository, the name is marked in red.
Type	Types associated with the robot.
Project name	Name of the project that the robot belongs to (useful when viewing all projects).
Tags	Tags assigned to the robot.
Version	Kofax RPA version last used when editing the robot.

Column	Description
Size	Size of the robot in bytes.
Schedules	Names of the schedules that run the robot.
Input types	Types used in input variables in the robot. To execute the robot, .type files corresponding to each of these types must be present.
Returned types	Types of values returned by the robot. When executing the robot through the API, it may return values of these types. To execute the robot, .type files corresponding to each of these types must be present.
Stored types	Types of values stored in a database by the robot. To execute the robot, .type files corresponding to each of these types must be present.
Triggers	Trigger names the robot is mapped to.
Labels	Label names the robot is mapped to.
Snippets used	Names of the snippets used by this robot. If a robot uses snippet A and snippet A uses snippet B, only snippet A is listed here.
Mappings	Shows robot user and label mappings.
Last modified	Date of the most recent modification of the robot.
Imported by	User name of the user who imported the robot that is part of the imported project or restored backup.
Imported at	Date of the importing of the robot that is part of the imported project or restored backup.
Optional Columns	
Robot ID	Automatically generated ID of the robot.
Created by	User name of the user who first uploaded the robot.
Modified by	User name of the user who last modified the robot.
Commit message	Summary describing the commit.
Revision number	Number of the robot revision.

Upload Robot

1. To add a robot to the Management Console, click the plus sign in the upper left corner.

   The "Upload Robot file" dialog box appears.

   An alternative way of uploading a robot is to use the Upload function in Design Studio. This works in exactly the same way, except that Design Studio also uploads the necessary types and snippets. If your Management Console repository contains multiple projects, you are prompted to choose the project to upload the robot.

2. Click the 📎 paper clip sign to select a robot file to upload, select the file on your computer, and then click **Open**.

   - If you are uploading a robot with the same name as an existing one, select **Override if exists** to replace the existing robot.
   - Select the folder to upload the robot. By default, all files are stored in the Root folder.
   - In the **Commit message** field, you may add a description for the commit.

3. Click **Submit**.

   The robot appears in the table.

   Once a robot is uploaded to Management Console, it can be executed in several different ways: Directly using the "Run now" option, as part of a schedule, through the Java/.NET API or as REST services. See "Actions on Robots" below.

   To execute a robot as part of a Kapplet, see Kapplets.

Actions on Robots

When you click the ⋮ context menu for a robot, the following actions are listed:

- **Run now**: Starts immediate execution of the robot on RoboServer. This feature is not available for robots that take input.
- **Set folder**: Adds the selected robot to a folder in the Management Console.
- **Create schedule**: Opens a wizard for schedule creation. If any robot added this way requires input, you need to add it later.
- **API**: Opens a window with a sample Java or C# code for executing the robot on RoboServer.
- **REST**: Opens a window that enables you to invoke the robot as a REST service.

  > **Note** When you start a robot through the Java/.NET API or as a REST service and a Management Console does not have enough slots to run it, a message appears that no slots are available. The robot is not queued. Therefore, when you start a robot using API, create a way to schedule robots such that they can always run when enough slots are available. For example, you can keep a count of the number of running robots and start a new robot only when enough slots are available, or you can have loops that attempt to run a robot and if the lack of slots prevents it from running, the loop should wait and try again.

- **SOAP**: Opens a window that enables you to invoke the robot with SOAP.
- **Add/edit password access for robot**: Opens a dialog box where you can create or edit the access token for a robot. The token corresponds to the particular version of a robot. Here, you can also edit a description of the token and assign a password entry to the token from the list of available password entries. You can assign password entries to the token directly on the Password entries tab in the Password store.
- **Get resource access token**: Opens a dialog box where you can copy the access token for this robot. For example, you can use this access token to allow the robot access to the Password store or Robot File System.
- **Map to user**: Available for robots with triggers in a Desktop Automation robot. Enables you to select a user to map the robot to.
- **Map to label**: Enables you to select a label to map the robot to. You can type a new label here or select an existing one.
- **Suspend triggers**: Available for robots with triggers in a Desktop Automation robot. Click to disable triggers in the robot.
- **Activate triggers**: Available for robots with triggers in a Desktop Automation robot. Click to enable triggers in the suspended robot.
- **Download**: Downloads a copy of the robot from the repository and saves it to your file system.
- **View robot runs**: Shows logged information on the robot runs as also appears in the Log view.
- **View robot errors**: Shows logged information on the robot errors as also appears in the Log view.

To remove a robot from the repository, select it in the table and click the 🗑 bin icon in the upper left corner. You are prompted to confirm the deletion. The robot is automatically removed from any schedules used to run it. If you do not have a copy of the robot in the file system, it is irrevocably lost.

API

In the **Repository** > **Robots** section, from the ⋮ context menu for a robot, click **API** to access the code generation window where you can generate code for Java or .NET.

Before you start using the API to execute robots, we recommend that you read the relevant chapters in the *Kofax RPA Developer's Guide* to understand how the API works.

REST

You can execute robots as REST services, which enables you to invoke a robot from any programming language, or directly from a browser using JavaScript.

In the **Repository** > **Robots** section, from the ⋮ context menu for a robot, click **REST**.

- Select the method to use, GET or POST, and configure the format for the request and response. Robots that require input must be invoked using POST. Robots without input may be invoked using either GET or POST.
- The format buttons help you configure the formats of the request and response while testing, but when you call the service from a code, the format is controlled by the `Accept` and `Content-Type` HTTP headers. The `Accept` header specifies the desired response format, and the `Content-Type` header specifies the request format.
- Use the Request pane of the service window to construct a request. Click **Test service** to execute the robot. The result is then displayed in the Response pane of the window.

REST services are easily invoked from a robot by using the Call REST Web Service action.

If the project or robot name contains any non-ASCII characters, make sure that the URL is encoded properly (UTF-8 URL encoding). This is automatically done in robots, but if the service is called from a code, the developer is responsible for encoding the URL.

Note Robots that run as services stop the first time the robot generates an API exception. This is different from scheduled robots, which continue to run regardless of any API exception generated by the robot.

Each robot that is run as a service uses a request thread. When a Management Console is running embedded in a RoboServer, a maximum of 100 request threads is available. These 100 threads are used for all types of HTTP requests, such as users accessing Management Console, uploads from Design Studio, and the Repository API. If you need to run a higher number of concurrent REST services, install a standalone version of Management Console on Tomcat so that you can control the number of request threads.

SOAP

Robots can initiate SOAP requests to communicate with programs installed on other computers, pass necessary information, and return a response.

In the **Repository** > **Robots** section, from the ⋮ context menu for a robot, click **SOAP** to access a window for editing and testing your SOAP request.

Input format

"Normal" or "flat" refers to the structure of a SOAP request message. For example, if a robot myRobot expects input variables var1 and var2, both of a type that has attributes attr1 and attr2, then "normal" would expect a SOAP message that looks similar to the following.

```
 <myRobot>
    <var1>
      <attr1>Some value</attr1>
      <attr2>Another value</attr2>
    </var1>
    <var2>
      <attr1>More input</attr1>
      <attr2>and some more</attr2>
    </var2>
  </myRobot>
```

The "flat" structure would require the SOAP message to look as follows:

```
<myRobot>
    <var1__attr1>Some value</var1__attr1>
    <var1__attr2>Another value</var1__attr2>
    <var2__attr1>More input</var2__attr1>
    <var2__attr2>and some more</var2__attr2>
  </myRobot>
```

The flat structure was introduced for compatibility reasons.

WSDL URL

The URL for the WSDL of the project that this robot belongs to. Note that this URL is identical for all robots of the same project.

Request URL

When running a robot, an HTTP POST request should be sent to this URL.

SOAP action

When running a robot, a HTTP header called SOAPAction should be present with the value shown.

Request

This field is pre-filled with an example SOAP message. All input attributes have default/test values. It can be edited before clicking **Test service**.

Response

A non-editable field which contains output from a robot run.

If there are errors in the input parameters or errors during the robot run, a SOAP Fault message is shown (containing a reason and some details for the error).

Important notes

- Project names can contain characters that are not allowed in WSDL; therefore, project names might be different in WSDL/SOAP messages. More specifically, all characters that are not alphanumeric (a-z, A-Z, 0-9) will be replaced by _.
- Similarly, robot names may appear different. They are converted similarly to project names, but when a robot name is changed, a special suffix (such as _1234) is also added.
- Currently, SOAP 1.1 is supported.

Types

The Types section lists types uploaded to the Management Console repository. When a schedule runs, the robots linked to it are executed on RoboServer. Many robots require types as definitions of either input, output, or both. These types must be added to the repository to the same project as the robot.

When you upload a type to the repository, it is copied into the repository. Thus, if changes are made later to the type in Design Studio, it needs to be uploaded again. As each type name must be unique within each project, multiple types with the same name can be uploaded only if they are placed in separate projects.

At the top of the Types section, in the Projects drop-down list, you can select the project with types to display. You can change the way the information for each type is presented as follows:

- Filter the list of types in the table by applying filters in the Filter text field. See Filtering for more information.
- Select the table columns to display for a type using the ▥ menu icon on the right.
- Refresh the displayed information by clicking the ↻ refresh icon on the right.

By default, the following table columns are displayed for each type.

Column	Description
Folder	Name of the folder specified to store types. By default, the files are stored in the Root folder. You can create a new folder to store the files, and the folder name must be unique. The folder name is displayed in the column if the selected folder is other than Root. When deleting a type, you can choose to delete an empty folder.
Name	Name of the type.
Size	Size of the type in bytes.
Project name	Displays the project the type belongs to (useful when viewing all projects).
Last modified	Timestamp for the last change of this type.
Optional Columns	
Created by	User name of the user who first uploaded the type.
Modified by	User name of the user who last modified the type.
Commit message	Summary describing the commit.
Revision number	Number of the type revision.
Imported by	User name of the user who imported the type that is part of an imported project or a restored backup.
Imported at	Date of the importing of the type that is part of an imported project or restored backup.

Upload Type

1. To add a type to the Management Console, click the plus sign in the upper left corner.

   The "Upload Type file" dialog box appears.

   Also, types can be implicitly uploaded from Design Studio. This happens when you use Design Studio to upload a robot that uses the type. As Design Studio knows about the dependencies between robots and types, it always uploads the necessary types along with the robot.
2. Click the paper clip sign to select a type file to upload, select the file on your computer, and then click **Open**.
   - If you are uploading a type with the same name as an existing one, select **Override if exists** to replace the existing type.
   - Select the folder to upload the type. By default, all files are stored in the Root folder.
   - In the **Commit message** field, you may add a description for the commit.
3. Click **Submit**.

   The type appears in the table.

Additionally, you can perform the following actions on types from the ⋮ context menu:

- **Set folder**: Change the folder where the type is located.
- **Create database table**: Create a database table to store extracted values in a database.
- **Download**: Download a copy of the type to your computer.

To remove a type, select it in the table and click the bin icon in the upper left corner. You are prompted to confirm the deletion. If the type is used by a robot or a snippet, the confirmation message includes the usage count. If you delete a type used by a robot or snippet, the robot (or robots using the snippet) will no longer be able to execute. If you do not have a copy of the type on your computer, it is irrevocably lost.

Create database table from types

To store extracted variable values in a database, you can create a matching database table. To create a database table from one or more types uploaded to Management Console, follow this procedure.

1. Select a type in **Repository** > **Types**.
2. Click the ⋮ context menu for the type and click **Create database table**.

   The **Create table** dialog box appears.
3. In the **Create table** dialog box, select one of the configured database mappings defined in a project and select whether to generate SQL code to drop tables. Click **Generate SQL**.

   The SQL editor opens, containing SQL code to generate (and drop if you selected **Generate SQL to drop tables**) the tables from the selected types on the selected database. The SQL shown is a recommended suggestion: You can change the statement to fit your needs, if required. When ready, click **Execute SQL**.

   After the SQL code is executed, a message is issued with an execution state (success or errors with description). Click **OK** to dismiss the message and close the windows.

Snippets

The Snippets section lists snippets uploaded to the Management Console repository. When a schedule runs, the robots linked to it are executed on RoboServer. Some robots use snippets that must be available in the repository in the same project as the robot.

When you upload a snippet to the repository, it is copied into the repository. Thus, if changes are later made to the snippet in Design Studio, it needs to be uploaded again. As each snippet name must be unique within each project, multiple snippets with the same name can be uploaded only if they are placed in separate projects.

At the top of the Snippets section, in the Projects drop-down list, you can select the project with snippets to display. You can change the way the information for each snippet is presented as follows:

- Filter the list of snippets in the table by applying filters in the Filter text field. See Filtering for more information.
- Select the table columns to display for a snippet using the ▥ menu icon on the right.
- Refresh the displayed information by clicking the ↻ refresh icon on the right.

By default, the following table columns are displayed for each snippet.

Name	Description
Folder	Name of the folder specified to store snippets. By default, the files are stored in the Root folder. You can create a new folder to store the files, and the folder name must be unique. The folder name is displayed in the column if the selected folder is other than Root. When deleting a snippet, you can delete an empty folder.
Name	Name of the snippet.
Project name	Name of the project the snippet belongs to (useful when viewing all projects).
Input types	Types used in input variables in the snippet. To use this snippet in a robot, the .type files corresponding to each of these types must be present.
Returned types	Types of values returned by the snippet. To use this snippet in a robot, the .type files corresponding to each of these types must be present.
Stored types	Types of values stored in a database by the snippet. To use this snippet in a robot, the .type files corresponding to each of these types must be present.
Snippets used	Name of the snippets used by this snippet. If a snippet uses snippet A and snippet A uses snippet B, only snippet A is listed here.
Size	Size of the snippet in bytes.
Created by	User name of the user who first uploaded the snippet.
Commit message	Summary describing the commit.
Revision number	The number of the snippet revision.
Last modified	Timestamp for the last change of this snippet.
Optional Columns	
Modified by	User name of the user who last modified the snippet.
Imported by	User name of the user who imported the snippet that is part of the imported project or restored backup.

Name	Description
Imported at	Date of the importing of the snippet that is part of the imported project or restored backup.

Upload Snippet

1. To add a snippet, click the plus sign in the upper left corner.

   The "Upload Snippet file" dialog box appears.

   Also, snippets can be implicitly uploaded from Design Studio. This happens when you use Design Studio to upload a robot that uses the snippet. As Design Studio knows about the dependencies between robots and snippets, it always uploads the necessary snippets along with the robot.
2. Click the 📎 paper clip sign to select a snippet file to upload, select the file on your computer, and then click **Open**.
   - If you are uploading a snippet with the same name as an existing one, select **Override if exists** to replace the existing snippet.
   - Select the folder to upload the snippet. By default, all files are stored in the Root folder.
   - In the **Commit message** field, you may add a description for the commit.
3. Click **Submit**.

   The snippet appears in the table.

Additionally, you can perform the following actions on snippets from the ⋮ context menu:

- **Set folder**: Change the folder where the snippet is located.
- **Download**: Download a copy of the snippet to your computer.

To remove a snippet, select it in the table and click the 🗑 bin icon in the upper left corner. You are prompted to confirm the deletion. If the snippet is used by a robot or snippet, the confirmation message includes the usage count. If you delete a snippet that is used by a robot, the robot will no longer be able to execute. When deleting a snippet, you can delete an empty folder.

Resources

The Resources section shows the resources uploaded to the Management Console repository. These resources can be used as input for scheduled robots that have an input variable with binary attributes. See Using Local Files in Robots for information on how to add and load such a variable to the robot.

At the top of the Resources section, in the Projects drop-down list, you can select the project with resources to display. You can change the way the information for each resource is presented as follows:

- Filter the list of resources in the table by applying filters in the Filter text field. See Filtering for more information.
- Select the table columns to display for a resource using the ▥ menu icon on the right.
- Refresh the displayed information by clicking the ↻ refresh icon on the right.

By default, the following information is displayed for each resource.

Column	Description
Name	Name of the resource.
Project name	Name of the project the resource belongs to (useful when viewing all projects).
Folder	Name of the folder specified to store resources. By default, the files are stored in the Root folder. You can create a new folder to store the files, and the folder name must be unique. The folder name is displayed in the column if the selected folder is other than Root. When deleting a resource, you can delete an empty folder.
Size	Size of the resource in bytes.
Last modified	Timestamp for the last change of the resource.
Modified by	User name of the user who last modified the resource.
Revision number	Number of the resource revision.
Commit message	Summary describing the commit.
Created by	User name of the user who first uploaded the resource.
Imported by	User name of the user who imported the resource that is part of an imported project or a restored backup.
Imported at	Date of the importing of the resource that is part of an imported project or a restored backup.

Upload Resource

1. To add a resource, click the plus sign in the upper left corner.

   The "Upload Resource file" dialog box appears.

2. Click the 📎 paper clip sign to select a resource file to upload, select the file on your computer, and then click **Open**.

   - If you are uploading a resource with the same name as an existing one, select **Override if exists** to replace the existing resource.
   - Select the folder to upload the resource. By default, all files are stored in the Root folder.
   - In the **Commit message** field, you may add a description for the commit.

3. Click **Submit**.

   The resource appears in the table.

Additionally, you can perform the following actions on resources from the ⋮ context menu:

- **Download**: Download a copy of the resource to your computer.

To remove a resource, select it in the table and click the 🗑 bin icon in the upper left corner. You are prompted to confirm the deletion.

Device mappings

The "Device mappings" section shows mapped automation devices available for robots. You can create mappings for these devices in Design Studio.

How it works

When a Desktop Automation robot starts, it does not have direct access to the IP addresses or host names of the devices listed under Management Console > Admin > Devices.

The robots use the device mapping name and label. If multiple devices with the same label are available, the robot picks a random one from the ones that are currently listed in the "Device mappings" section.

The call asking for an available device is made through Hazelcast that might return the same device each time if it is available. Management Console or robot do not force or control the order. The use of labels offers redundancy because the robot is not tied to a single device, which would lead to an error if the device is down or in use, but it can use any of the devices associated with the label from the device mapping specified in the robot.

You can change the way the information for each device mapping is presented as follows:

- Filter the list of device mappings in the table by applying filters in the Filter text field. See Filtering for more information.
- Select the table columns to display for a device mapping using the ▥ menu icon on the right.
- Refresh the displayed information by clicking the ↻ refresh icon on the right.

By default, the following information is displayed for each device mapping.

Column	Description
Name	Name of the device mapping.
Labels	Labels of the device mapping. You can use the labels to filter the devices to automate with your robot.

Create New Automation Device Mapping

1. To create a new device mapping, click the plus sign in the upper left corner.

   The "Add device mapping" dialog box appears.
2. Specify a name and a label (or multiple labels) for the device mapping.

   The labels must be separated by commas.
3. Click **Save**.

Additionally, you can perform the following actions on device mappings from the ⋮ context menu:

- **Edit**: Contains the same fields as the "Add device mapping" dialog box.

To remove a device mapping from the Management Console, select it in the table and click the 🗑 bin icon in the upper left corner. You are prompted to confirm the deletion.

Database mappings

The "Database mappings" section lists database mappings of the selected project. You can link your robot to different databases in a cluster by using mappings, and you can create new mappings in this section. After you install Kofax RPA, a default database mapping "`objectdb`" is added to the default project, which is pointing to the Production cluster default development database.

For more information on database mappings, see Map Databases in the Design Studio chapter.

When creating a database mapping, note the following:

- You cannot have database mappings with the same name in one project.
- You can use space, parentheses, and hyphens for database mapping names. For example, `"Development Database (MySQL)"` is a valid database mapping name.

At the top of the "Database mappings" section, in the Projects drop-down list, you can select the project with databases to display. You can change the way the information for each database mapping is presented as follows:

- Filter the list of database mappings in the table by applying filters in the Filter text field. See Filtering for more information.
- Select the table columns to display for a database mapping using the ▥ menu icon on the right.
- Refresh the displayed information by clicking the ↻ refresh icon on the right.

By default, the following information is displayed for each database mapping.

Column	Description
Name	Database mapping name.
Project name	The name of the project that the mapping is assigned to (useful when viewing all projects).
Database	Name of the database that the mapping is used for.
Cluster	Name of the cluster that the database mapping is related to.

Create New Database Mapping

1. To create a new database mapping, click the plus sign in the upper left corner.
   The "Add database mapping" dialog box appears.
2. Specify a project to assign the mapping to.
3. Select a cluster and a database to map the database mapping to.
4. Click **Save**.

Additionally, you can perform the following actions on database mappings from the ⋮ context menu:

- **Edit**: Contains the same fields as the "Add database mapping" dialog box.

To remove a database mapping from the Management Console, select it in the table and click the 🗑 bin icon in the upper left corner. You are prompted to confirm the deletion.

Note If you delete the last mapping of a database, this database will not be available in Design Studio even if a cluster is selected to provide Design Studio with shared databases.

Email triggers

This "Email triggers" section lists email triggers set up for the email trigger functionality. Use this section to add new triggers and delete triggers that you no longer need.

The robot specified for the email trigger must contain one variable of complex type with a long text attribute. The robot can read all headers, body, and one or more attachments of an email. All attachments are encoded to Base64 format. If the robot does not contain any variable, the email is not processed and used just as a trigger to start a robot.

At the top of the "Email triggers" section, in the Projects drop-down list, select the project with email triggers to display. You can change the way the information for each email trigger is presented as follows:

- Select the table columns to display for an email trigger using the ▥ menu icon on the right.
- Refresh the displayed information by clicking the ⟳ refresh icon on the right.

By default, the following information is displayed for each email trigger.

Column	Description
Email trigger	Name of the email trigger.
Assigned email folder	Folder on the email server. Email messages that appear in this folder initiate the trigger.
Robot	Robot that uses the email trigger.
Project	Project that contains the email trigger.

Create New Email Trigger

1. To add a new email trigger, click the plus sign in the upper left corner.
   The "Create email trigger mapping" dialog box appears.
2. Specify the following parameters:
   a. Trigger name: Enter a name for the email trigger.
   b. Select project: Select a project to contain the email trigger.
   c. Select email account: Select an email account for the trigger. See Email accounts for details.
   d. Select robot: Select a robot that will use the trigger.
   e. Select email folder: Select a folder on the email server. Email messages that appear in this folder initiate the trigger. If the email server provides a list of subfolders under **Inbox** and you select one of the subfolders, only messages that appear in the selected subfolder trigger robot execution. In this case, messages placed to **Inbox** do not trigger robot execution.
   If the email server does not provide a list of subfolders, select **Inbox** and all messages that appear in **Inbox** and its subfolders will trigger a robot.
3. Click **OK**.

Additionally, you can perform the following actions on email triggers from the ⋮ context menu:

- **Edit**: Contains the same fields as the "Create email trigger mapping" dialog box.

To delete an email trigger from the Management Console, select it in the table and click the 🗑 bin icon in the upper left corner. You are prompted to confirm the deletion.

Trigger mappings

The "Trigger mappings" section shows available trigger mappings. Triggers are part of the Attended Automation functionality in Desktop Automation robots. A trigger robot is a robot that reacts to an event on a remote device. After uploading a robot with a trigger to a Management Console, you can map the robot to users and/or labels. After that the Management Console provides a list of triggers to the Desktop Automation Service based on the created mappings. When a trigger event is detected on a remote device, the Desktop Automation Service sends a notification to the Management Console and the robot performs some programmed steps. For example, you can program the robot to insert or extract some data when a certain application is opened.

Kofax RPA supports user and label mappings for robots. User mapping is only available for robots with triggers. You can enable or disable triggers in a robot in the Robots section.

The "Trigger mappings" section contains two tabs: Users and Labels. The Users tab shows robots with assigned user mapping. The Labels tab shows robots with assigned labels. You can only assign user mappings to robots with triggers.

You can change the way the information for each mapping is presented as follows:

- Filter the list of trigger mappings in the table by applying filters. You can filter by robot name and user/label name, depending on the selected tab. When creating a trigger mapping, you can filter the list of robots.
- Refresh the displayed information by clicking the ↻ refresh icon on the right.

Users

The following information is displayed for each user mapping.

Column	Description
Robot name	Name of the robot that the user mapping is based on.
Triggers	Names of the associated triggers.
Project name	Name of the project that the user mapping belongs to.
User name	Users that the robot is mapped to.

Labels

The following information is displayed for each label mapping.

Column	Description
Robot name	Name of the robot that the label mapping is based on.
Project name	Name of the project that the label mapping belongs to.
Label	Labels that the robot is mapped to.

Create User Mapping

1. Click the plus sign on the Users tab.

   The "Add trigger-user mapping" dialog box appears.

2. On the Robots tab, select robots that the mapping will be based on. On the Users tab, select users to map the robots to. Click **OK**.

   The new mapping is added to the table. If you selected multiple robots, multiple new mappings are added.

Create Label Mapping

1. Click the plus sign on the Labels tab.
   The "Add trigger-label mapping" dialog box appears.
2. Select robots that the mapping will be based on and then add labels to map the robots to. Click **OK**.

   The new mapping is added to the table. If you selected multiple robots, multiple new mappings are added.

Additionally, you can perform the following actions on trigger mappings from the ⋮ context menu:

- **Edit**: Contains the same fields as the "Add trigger-user mapping" dialog box.

To remove a trigger mapping from the Management Console, select it in the table and click the 🗑 bin icon in the upper left corner. You are prompted to confirm the deletion.

OAuth

This section contains the OAuth applications and users that are authenticated using Management Console. The user credentials can be used as input for robots in a schedule, allowing them to access APIs on behalf of the authenticated user without having access to the user name and password.

See the OAuth section for more information on how to create and manage robots that access APIs that are protected by OAuth.

The OAuth section contains two tabs: Applications and Users.

- To add a new OAuth application, see Add applications.
- To add a new user for an existing OAuth application, see Add users.

At the top of the OAuth section, in the Projects drop-down list, you can select the project with OAuth applications to display.

Applications

By default, the following information is displayed for each application.

Column	Description
Name	Name of the application.
Service provider	Provider of the web service.
Project name	Name of the project the application belongs to (useful when viewing all projects).
Modified by	User name of the user who last modified the application.
Commit message	Summary describing the commit.
Revision number	Number of the application revision.

Users

By default, the following information is displayed for each user.

Column	Description
Name	Name of the user.
Application	Application the user belongs to.

Password store

Password store is a password repository designed to grant access to different systems without disclosing sensitive information. The "Password store" section shows available and assigned password store entries. Here you can create new and edit existing entries.

The section contains two tabs:

- Passwords
- Password access

At the top of each tab, in the Projects drop-down list, you can select the project with password store entries to display.

You can change the way the information for each password store entry is presented as follows:

- Filter the list of entries in the table by applying filters in the Filter text field. See Filtering for more information.
- Select the table columns to display for an entry using the ▥ menu icon on the right.
- Refresh the displayed information by clicking the ↻ refresh icon on the right.

Use Password store

The following are general steps for using the password store.

1. In the selected Management Console, navigate to **Repository** > **Password store** and create a password entry on the Passwords tab.
2. In Design Studio, navigate to **Settings** > **Design Studio Settings**, and on the **Management Consoles** tab, specify the Management Console to store passwords and select **Use as Primary**.
3. Once a robot is ready for deployment, it needs to be uploaded to the Management Console. When the robot is uploaded, in the Management Console, on the Password access tab, create a new password access entry for the uploaded robot using an access token for a robot (can be obtained by clicking **Get resource access token** in the robot context menu) or a Design Studio instance (can be copied from the About window in the Design Studio). Also, use the password entry created in Step 1.

Important Every time you upload your robot or any of its components, such as types, snippets, and so on, to Management Console, a new password access entry must be created for the robot. Previous entries are kept in the password access list and you can delete them manually.

Passwords

On this tab, you can create password entries for accessing a particular target system.

By default, the following table columns are displayed for each password entry.

Column	Description
User name	User name of the user accessing the target system.
Target system	Description of the target system to access.
Project name	Project containing the password entry.

Create New Password Entry

1. To create a new password entry, click the plus sign.

   The "Password entry" dialog box appears.

2. Fill in the following fields:
   - Projects: Select a project to contain the password entry.
   - User name: Enter a user name to access the target system that you specify below.
   - Target system: Provide a description of the system to access. This is required to distinguish this target system from other target systems.

     Note When you insert the Lookup Password step, the step's **Target System** property value must match the **Target system** value of the password entry.

   - Password: Enter a password.
   - Retype password: Re-enter the password.

3. Click **Save**.

   The new password entry appears in the table.

Additionally, you can perform the following actions on password entries from the ⋮ context menu:

- **Edit**: Contains the same fields as the "Password entry" dialog box.
- **Move password entry**: Moves a password entry from one project to another. The password store access is project-based. You can see password entries assigned to projects that you have access to. Management Console Administrator can move password entries between projects for Project Administrators to manage password entries and grant access to target systems. Use the Merge entries option to merge entries with identical user names and target systems.

To remove a password entry, select it in the table and click the 🗑 bin icon. You are prompted to confirm the deletion.

Password access

On this tab, you can create password access entries, for example, to grant a particular robot or a particular Design Studio instance access to the password store.

By default, the following table columns are displayed for each password access entry.

Column	Description
Password access token	Access token from Design Studio.
Description	Description identifying an entry in the table.

Column	Description
Project name	Project containing a password access entry.
Password entries	Password entries that the access is granted to.

Create New Password Access Entry

1. To create a new access entry, click the plus sign.

   The "Password access entry" dialog box appears.
2. Fill in the following fields:
   - Projects: Select a project to contain the password entry.
   - Password access token:
     - To grant access to a Design Studio instance, enter the access token copied from the **Help** > **About** window in Design Studio.
     - To grant access to a particular robot (it must be uploaded to Management Console), go the Robots section, and from the context menu for the required robot, click **Get resource access token**, copy the token, and then enter it here.
   - Description: Specify a description to identify an entry in the list.
   - Password entries: Select a user/target system to grant access to the password store.
3. Click **Save**.

   The new password entry appears in the table.

Additionally, you can perform the following actions on password access entries from the ⋮ context menu:

- **Edit**: Contains the same fields as the "Password access entry" dialog box.

To remove a password access entry, select it in the table and click the 🗑 bin icon. You are prompted to confirm the deletion.

CyberArk

CyberArk is a third-party information security software supported by Kofax RPA. It functions as an external password manager. Use CyberArk as an alternative to the Kofax RPA built-in password manager.

To install CyberArk, see the *Kofax RPA Installation guide*.

To configure CyberArk, see CyberArk configuration and CyberArk applications.

As well as the password store, CyberArk is designed to grant access to different systems without disclosing sensitive information. The CyberArk section shows available and assigned password store entries. Here you can create new and edit existing entries.

The section contains two tabs:

- Passwords
- Password access

Use CyberArk

The following are general steps for using CybeArk.

1. In the selected Management Console, navigate to **Repository** > **CyberArk** and create a password entry on the Passwords tab.
2. In Design Studio, navigate to **Settings** > **Design Studio Settings**, and on the **Management Consoles** tab, specify the Management Console to store passwords and select **Use as Primary**.
3. Once a robot is ready for deployment, it needs to be uploaded to the Management Console. When the robot is uploaded, in the Management Console, on the Password access tab, create a new password access entry for the uploaded robot using an access token for a robot (can be obtained by clicking **Get resource access token** in the robot context menu) or a Design Studio instance (can be copied from the About window in the Design Studio). Also, you the password entry created in Step 1.

Important Every time you upload your robot or any of its components, such as types, snippets, and so on, to Management Console, a new password access entry must be created for the robot. Previous entries are kept in the password access list and you can delete them manually.

Passwords

On this tab, you can create password entries for accessing a particular target system.

By default, the following table columns are displayed for each password entry.

Column	Description
User name	User name of the user accessing the target system.
Target system	Description of the target system to access.
Project name	Project containing the password entry.
Account name	Account name corresponding to the required CyberArk account name.
Application ID	Application from the CyberArk applications.
Safe	CyberArk account entry safe name.

Create New Password Entry

1. To create a new password entry, click the plus sign.

   The "Password entry" dialog box appears.

2. Fill in the following fields:
   - Projects: Select a project to contain the password entry.
   - User name: Enter a user name to access the specified target system.
   - Target system: Provide a description of the system to access. This is required to distinguish this target system from other target systems.

     > **Note** When you insert the Lookup Password step, the step's **Target System** property value must match the **Target system** value of the password entry.

   - Application ID: Select an application from the CyberArk keystore list.
   - Safe: Enter a CyberArk account entry safe name.
   - Account name: Enter an account name corresponding to the required CyberArk account name.

3. Click **Save**.

   The new password entry appears in the table.

Additionally, you can perform the following actions on password entries from the ⋮ context menu:

- **Edit**: Contains the same fields as the "Password entry" dialog box.
- **Move password entry**: Moves a password entry from one project to another. The password store access is project-based. You can see password entries assigned to projects that you have access to. Management Console Administrator can move password entries between projects for Project Administrators to manage password entries and grant access to target systems. Use the Merge entries option to merge entries with identical user names, target systems, application IDs, account names, and safes.

To remove a password entry, select it in the table and click the 🗑 bin icon. You are prompted to confirm the deletion.

Password access

On this tab, you can create password access entries, for example, to grant a particular robot or a particular Design Studio instance access to the password store.

By default, the following table columns are displayed for each password access entry.

Column	Description
Password access token	Access token from Design Studio.
Description	Description identifying an entry in the table.
Project name	Project containing a password access entry.
Password entries	Password entries that the access is granted to.

Create New Password Access Entry

1. To create a new access entry, click the plus sign.

   The "Password access entry" dialog box appears.
2. Fill in the following fields:
   - Projects: Select a project to contain the password entry.
   - Password access token:
     - To grant access to a Design Studio instance, enter the access token copied from the **Help** > **About** window in Design Studio.
     - To grant access to a particular robot (it must be uploaded to Management Console), go the Robots section, and from the context menu for the required robot, click **Get resource access token**, copy the token, and then enter it here.
   - Description: Specify a description to identify an entry in the list.
   - Password entries: Select a user/target system to grant access to the password store.
3. Click **Save**.

   The new password entry appears in the table.

Additionally, you can perform the following actions on password entries from the ⋮ context menu:

- **Edit**: Contains the same fields as the "Password access entry" dialog box.

To remove a password access entry, select it in the table and click the 🗑 bin icon. You are prompted to confirm the deletion.

Robot File System

The Robot File System section lists file systems configured to share and store data used and/or produced by robots. You can add a configuration for a file system by providing the required credentials and defining a list of robots and users to access this file system.

The file system can be a local Windows folder, Windows file share, SFTP, and FTP server. For FTP, FTPS is supported. The maximum size of the file that can be uploaded to the Robot File System is 100 MB.

For information on how to configure a server for the Robot File System, see "Set up Robot File System server" in the *Kofax RPA Administrator's Guide*. The guide also contains an example of general configuration and usage steps.

Important By default, only users with the **admin**, **RPA Administrators**, **Project Administrator** role can manage robot file systems. See Users & groups for more information on user roles.

At the top of the Robot File System section, in the Projects drop-down list, you can select the project with file systems to display. You can change the way the information for each configure file system is presented as follows:

- Filter the list of file systems in the table by applying filters in the Filter text field. See Filtering for more information.
- Select the table columns to display for a file system using the ▥ menu icon on the right.
- Refresh the displayed information by clicking the ↻ refresh icon on the right.

By default, the following information is displayed for each configured file system.

Column	Description
Name	Name of the file system configured to store and share robot data.
Project name	Project that the file system is associated with.
Path	Path to the file system.
User name	User that has access to the file system.
Project scope	If selected, the file system is accessible to all robots of the project. Otherwise, the file system is only accessible to particular robots selected during configuration.
Optional Columns	
ID	ID of the file system configured to store and share robot data.

Configure New Robot File System

1. To add a new file system, click the plus sign in the upper left corner.

   Several new tabs open.

2. On the **General information** tab, specify the following information:

   - **Project**: Specify a project to contain the file system.
   - **Project scope**: Select this option to make the file system accessible to **all robots** of the project. Alternatively, you can make the file system accessible only to particular robots as shown in Step 3.
   - **File system name**: Type the name for the new file system configuration.

     Example: `RFS1`
   - **Path**: Specify the path to the file system.
     - For Windows, the path must be similar to the following:

       `Folder\Subfolder1\Subfolder2`
     - For a Windows file share, the path must be similar to the following:

       `\\WindowsServer\FileShareName\Folder`

       This file share is mapped to the Robot File System as specified in the **Name** field.
     - For an SFTP, FTP, or FTPS server, the path must start with `sftp://`, `ftp://`, or `ftps://`, respectively, and be similar to the following:

       `sftp://website.com:9551/guest`
   - **User name** and **Password**: Type the credentials to access the file system.

     For Windows, the user name must conform to: `username@domain`

     The credentials are *only* used by a secure service to allow robots to read and/or write to the file system.
   - Save the changes.

3. To make the file system accessible to **particular robots** within the project, you can map them by names or add access tokens for those robots. This method works as long as the robot name remains unchanged. Click ⋮ to open the context menu for the file system you are working on and click **Edit**.
   - Use the robot name to ensure that all versions of the robot (current and future) have access to the file system.

     On the **Robot mapping** tab, select which robots have access to the file system. The robots must be synchronized with the Management Console to appear on this tab. Save the changes.
   - Use the robot access token to ensure that only the current version of the robot has access to the file system and that no changes made to the robot are written into the file system. As the token corresponds to the particular version of a robot, if the robot changes, it will no longer be allowed to the file system.
     - **a.** In the Management Console navigation menu, in the **Robots** section, click ⋮ to open the context menu for the required robot and then click **Get resource access token**.

       A new dialog box appears.
     - **b.** Copy the token and close the dialog box.
     - **c.** Return to the **Robot File System** section, click to edit the file system, and on the **Authorized access tokens** tab, add the copied access token.
     - **d.** Save the changes.
4. Additionally, you can configure Design Studio instances that are allowed to the file system, so that users can design a robot to read and/or write to the file system.

   To make the file system accessible to a particular Design Studio instance, on the **Authorized access tokens** tab, add an access token for that instance.

   The token must copied from the **Help** > **About** window in Design Studio and shared with the administrator.
5. Save the changes.

To delete a file system, select it in the table and click the 🗑 bin icon in the upper left corner. You are prompted to confirm the deletion.

Data view

The "Data view" menu helps you view database data for projects and export data extracted by robots. You can also use it to see database tables created from types using Design Studio or Management Console.

At the top of the menu, in the Projects drop-down list, you can select the project with the data to display. On the left side, you can see the Database navigation tree displaying database mappings defined for the selected project.

If no database connection is established, or if tables have not been created for a database, the database mapping name under the navigation tree is not active (not expandable).

When you click a database mapping name, the tree is expanded, and you can see the various tables contained in the database. When you select a table, the type data is displayed in the right pane, including the names of types, robots associated with those types, and types attributes. You can copy data directly from the table. You can select the columns to display for a database table using the ▥ menu icon on the right.

Once the data is loaded, a number of filters become available that work in the same way as the filters in the Log view. For example, you can filter by the following conditions: Contains, Not contains, One of, Starts with, and so on. `Binary` and `longtext` attributes are not filterable.

Export data

Above the data grid, in the **Export to** drop-down list, you can select the format to export data extracted by robots: **XML**, **CSV**, or **EXCEL**.

To see previous exports and/or download them again, click **Exports** ▦.

By default, the following information is displayed for each export.

Column	Description
Created	Date when the export was created.
Table	Name of the table that the export was created for.
Database	Name of the database that the table belongs to.
Catalog	Database catalog that the table belongs to, if applicable.
Download	Click to download the export.

The exports are automatically deleted the next time the Management Console is started. The oldest exports are deleted when the number of exports exceeds 100. There is no limit to the number of rows you can export to CSV or XML, but Excel files are limited to 10000 records, to prevent the system from running out of memory.

The list of schemas/catalogs available under each database in the navigation tree is controlled by the permissions of the user whose credentials are used in the cluster settings database configuration.

Log view

The "Log view" menu contains logs created by Management Console and by the database logging of RoboServers. Both Management Console logs and RoboServers logs are written to the logging database, thus requiring a logging database to be set up in the Management Console settings and the database logging to be enabled on RoboServers with the logging cluster settings.

The **Schedule runs** and **Schedule messages** are Management Console logs that report information on schedules. The remaining logs, **Robot runs**, **Robot messages**, **Robot summary**, **RoboServer messages**, **DAS messages**, and **Task messages** are RoboServer logs containing information on the status of RoboServers and on robots and robot runs.

The page layout for each log is identical, but you can change the way the information for each log is presented as follows:

- Filter the lists of messages in the table by applying filters in the Filter text field. When you click the filter ▼ icon, the "Column filters" dialog box appears with a number of filters.

  For example, depending on the log type, you can filter by existing projects and orphan projects, filter by a time frame, total execution time, execution result, severity of error messages, and so on.

  Also, see Filtering.
- Select the table columns to display for a log using the ▥ menu icon on the right.

- Refresh the displayed information by clicking the ↻ refresh icon on the right.

If there are more results than the number selected per page, you can navigate to the next page using the controls under the data grid.

You can set up the retention policy for your logs in the RoboServer log database by specifying a number of days to keep the logs and a number of messages in the robot run. The logs are cleaned on a daily basis by deleting the oldest messages first. The default values are 10 days and 500 messages.

Schedule runs

Displays execution information for each schedule that executes, such as when the schedule started and when it finished.

Use the ⋮ context menu to navigate to the individual schedule message, or to the robots that were executed as part of this schedule run.

You can delete a schedule run using the context menu. When deleting, you always have to delete the messages, but you can keep the run information if you need. You can delete this run and messages, or all runs or messages matching the current filter. Deleting many runs or messages might take some time.

Schedule messages

Displays individual message entries for a given schedule run. It gives you the ability to see detailed information about why a schedule failed to run.

To find robot errors for a given schedule run, select **View robots with errors** from the ⋮ context menu.

You can delete one or more records. When deleting, you can select to delete only this message, all messages matching the current filter, or all RoboServer log messages (due to performance, the option to delete message matching a filter is disabled if there are more than 500 matching results).

When selecting a value in the Severity filter, note that the result will be greater or equal to the selected value. For example, if you set the Severity filter to "Warning," the result will also contain errors and fatal errors, apart from warnings.

Robot runs

Displays information for each robot run. The ⋮ context menu also enables you to view all runs for the same robot, or view the input this robot was given when this run was executed.

You can delete a run and messages. When deleting, you always have to delete the messages, but you can keep the run information if you like. You can delete this run and messages, or all runs or messages matching the current filter. Deleting many runs or messages might take some time.

Robot messages

Displays individual error messages belonging to a robot run. Using the ⋮ context menu, you can navigate to the run this message belongs to.

For robot errors, the Location code column in the data grid contains a link. When you click the link, a small .robotDebug file is downloaded. If you open the file with Design Studio, the robot is loaded, the input from the run is set, and the robot navigates to the location of the error, enabling you to quickly debug errors.

When selecting a value in the Severity filter, note that the result will be greater or equal to the selected value. For example, if you set the Severity filter to "Warning," the result will also contain errors and fatal errors, apart from warnings.

Robot summary

This is a simple summary view of all robots ever run (for as long as data is available). Using the ⋮ context menu, you can navigate to all runs for the given robot.

RoboServer messages

Displays general messages from RoboServer or Management Console.

You can delete one or more records. When deleting, you can select to delete only this message, all messages matching the current filter, or all RoboServer log messages (due to performance, the option to delete message matching a filter is disabled if there are more than 500 matching results).

When selecting a value in the Severity filter, note that the result will be greater or equal to the selected value. For example, if you set the Severity filter to "Warning," the result will also contain errors and fatal errors, apart from warnings.

DAS messages

Displays events received from Desktop Automation Service. See Logging for Desktop Automation Service for more information.

When selecting a value in the Severity filter, note that the result will be greater or equal to the selected value. For example, if you set the Severity filter to "Warning," the result will also contain errors and fatal errors, apart from warnings.

Task messages

Displays individual messages informing of the state of the queuing tasks. The following message types can be shown for a task: queuing, running, completed, timed out, and removed. All of the messages are supported by a description that appears in the Details column.

The task is timed out if it did not get the resource it required, such as Desktop Automation Service, license units, or RoboServer execution slots. The task is removed if the schedule or the respective task in the "Task view" was stopped, if the RoboServer was stopped, or if the Management Console went offline.

See Queuing of schedule jobs for more information.

Admin

Use this menu to administer Management Console.

Task view

Display the robots and other tasks executed by Management Console.

RoboServers

Add or delete RoboServers and clusters, and configure cluster settings. View running robots.

Projects

Create and delete projects.

High Availability nodes

View configured high availability nodes.

Devices

View available automation devices registered with Management Console.

Users & groups

Create new users and groups. View information on the Management Console users.

Backups

Create and restore backups and import/export projects.

License

Enter license information or view information on the current license and Design Studio seats in use.

Task view

The "Task view" gives an overview of the current activity across all servers and robots. This section shows scheduled robots as well as pre- and post-processing tasks started by the Management Console scheduler. To learn about the outcome of previous robot runs, view the logs.

Note Due to update delays, short-running robots may never (or almost never) appear in this view.

At the top of the "Task view" section, in the Clusters drop-down list, you can select the cluster with robots and tasks to display. You can change the way the information is presented as follows:

- Select the table columns to display using the ▥ menu icon on the right.
- Refresh the displayed information by clicking the ↻ refresh icon on the right.

By default, the following table columns are displayed for all currently running robots.

Column	Description
Name	Name of the task that is running.
Schedule	Name of the schedule that the task is running within.
Project name	Name of the project that the task belongs to.
Status	The task may be in one of the following states: **Queued** The task is queued and waiting for execution. A task may be queued for the following reasons: • **No servers** There are currently no servers in the cluster, or all servers are offline. The task will start to execute when a server is added, or an offline server comes online. • **No capacity** All servers are executing at maximum capacity. The task will start to execute when other tasks finish and capacity becomes available, or if additional server are added to the cluster. • **Waiting for other task** The task may be waiting for another task to finish. Post-processing may not be run until all robots have finished, and robots may not be run before pre-processing has finished. Also, if robots on a schedule are configured for sequential execution, robots will remain queued until the previous robot completes. **Running** The task is currently executing. **Acquired** The task is sent to RoboServer but the execution has not started yet.

Column	Description
Last status change	Time of the last status change.
Input	Summary of input for the running robot.
Priority	Execution priority for the task, Minimum, Low, Medium, High, or Maximum, in the queue of other tasks.
Expiry	Time remaining before the timeout for task queuing is reached.

RoboServers

The RoboServers section enables you to manage clusters and RoboServers known to the Management Console. When in embedded mode, by default, the list contains one cluster with one RoboServer, which is the one that also runs the Management Console functionality. In a larger setup with multiple RoboServers and clusters, we recommend that you deploy Management Console on a standalone web container if your license permits. See the *Kofax RPA Administrator's Guide* for additional information about configuring the Management Console.

You can change the way the information for each cluster/RoboServer is presented as follows:

- Select the table columns to display for a cluster/RoboServer using the ▥ menu icon on the right.
- Refresh the displayed information by clicking the ↻ refresh icon on the right.

By default, the following table columns are displayed for each cluster/RoboServer.

Column	Description
Cluster	Name of the cluster. The name is suffixed by SSL if the cluster uses SSL.
Action	This column lists available actions. **For a cluster** • Cluster settings: Opens the tabs with the cluster settings. • Add server: Opens a dialog box to add a RoboServer to this cluster. Specify a host and port and click **OK**. • Assign CRE: Opens the Assign license units pane. • Delete: Deletes a cluster. **For a RoboServer** • Dump thread: Sends a request to the selected RoboServer to perform a full thread dump. • Stop RoboServer: Opens a dialog box where you can stop/restart the selected RoboServer. Select how you want to shut down the RoboServer, specify a shutdown timeout if required, and then click **OK**. The timeout is measured in milliseconds, and if it is set to -1, the shutdown is forced immediately. • Stop all robots: Stops all running robots in the selected RoboServer. • Delete RoboServer: Deletes the selected RoboServer.
Server	Name or IP address and port of the RoboServer.
Version	Version of the software on the running RoboServer.

Column	Description
Status	**For a cluster**, shows the cluster state: • Initializing • Shutting down • Running • Paused • Invalid settings • Sending settings to RoboServer • Wait for schedules and robots to finish • Wait for currently running robots to finish • Stop all robots and schedules and apply settings immediately After you have changed the cluster settings, the "Apply cluster settings" dialog box is shown with options to proceed. **For a RoboServer**, shows if the server is online or offline.
Assigned license units	For information on the CRE and KCU licensing, see the sections "Concurrent Robot Execution License" and "Kofax RPA Compute Units" in the *Kofax RPA Installation Guide*. **For CRE-based licences** In the **Static** license distribution mode, this column indicates how many robots can run concurrently in this cluster. CREs are distributed evenly between online RoboServers in the cluster. A CRE is an integral unit, and you cannot split one CRE among multiple RoboServers. For example, if you have six CREs, but five RoboServers in a cluster, each RoboServers gets one CRE; therefore, one CRE remains unused. **Note** The number of CREs in a cluster must be equal to or bigger than that of RoboServers. If you assign fewer CREs to a cluster than the number of RoboServers present in the cluster, the cluster is disabled. In the **Dynamic** license distribution mode, this column indicates the total number of CRE licenses assigned to the cluster. RoboServers receive the licenses from the cluster per request. A RoboServer can get as many licenses as it requests if they are available. In this mode, RoboServers communicate only with the Management Console and block other requests, such as API calls. The number of concurrent robots a RoboServer can execute also depends on the amount of CPU available and the speed required to get the data for the RoboServer to process. For more information, see the section "Production Configuration" in the *Kofax RPA Administrator's Guide*. To adjust the number of CREs, in the **Action** column, from the ⋮ context menu for a cluster, click **Assign CRE**. This action opens the Assign license units pane where you can adjust the number of license units and see how many units are available in total and how many remain. **For KCU-based licences** Indicates the number of KCUs assigned to this cluster. KCUs in a cluster are distributed evenly among online RoboServers in the cluster. To adjust the KCUs on a cluster, in the **Action** column, from the ⋮ context menu for a cluster, click **Assign KCU**. This action opens the Assign license units pane where you can adjust the number of license units and see how many units are available in total and how many remain.
License type	License type of the cluster: Production or Non-Production.

Column	Description
Optional Columns	
Running robots	Number of robots currently running on the RoboServer.
Queued robots	Number of queued robots on the RoboServer.
Max robots	Maximum number of concurrent robots on the RoboServer. Can be configured in the cluster settings.
Uptime	Uptime of the RoboServer. Enables you to see when the server was started or restarted.
Command line	Command line the RoboServer was started with.
CPU count	Number of CPUs assigned to the RoboServer process. For example, if CPU affinity is assigned.
Memory limit	Maximum amount of memory assigned to the JVM the RoboServer runs in.
Duration (accum.)	Shows the total time the RoboServer has been in the "Above limit" state.
Above limit	Shows whether the server is operating above its memory threshold, which is 80% by default. If this limit is reached, the RoboServer queues the robot instead of starting it.
Max queue	Maximum number of robots that can be queued on the RoboServer. Can be configured in the cluster settings.
Last updated	Shows the time when the Management Console received the latest status update from the RoboServer.
Temp profiling	Indicates whether profiling is temporarily enabled for a given RoboServer. The setting will be cleared upon RoboServer restart.

Create New Cluster

1. To create a new cluster, click the plus sign in the upper left corner.

   The "Add cluster" dialog box appears.

2. Specify a name for the cluster and the cluster type.

   If you create a production cluster, you can assign license units from the production license. Similarly, if you create a non-production cluster, you can assign license units from the non-production license.

3. Select a license distribution mode.

4. If you select **Use SSL**, all RoboServers in the cluster must use the SSL RQL service.

5. Click **Submit**.

   The new cluster appears in the table.

After creating a cluster, you can add RoboServers to the cluster, using the ⋮ context menu in the **Action** column for the cluster. A cluster can include RoboServers of different Kofax RPA product versions, making it possible to gradually update the robots. In the cluster, robots of older versions are forwarded to the nearest available version of RoboServer. For an example, see Threshold RoboServer version.

Load Distribution and Failover

When a cluster needs to execute a robot, it finds the RoboServers with the highest number of available slots. Available slots are calculated based on how many robots are already running on the RoboServers and how many robots it can run concurrently (the maximum concurrent robots as set in the cluster settings.)

If a RoboServers in a cluster goes offline, the KCU is automatically distributed evenly among the remaining RoboServers.

Actions on RoboServers and Clusters

To perform an action on a RoboServer or a cluster, such as add a RoboServer to a cluster, use the ⋮ context menu in the **Action** column. For a description of the actions, see the **Action** row in the preceding table.

Running robots view

When you click a cluster or a RoboServer name, a view opens containing detailed information about running robots in the selected cluster (from all RoboServers in this cluster) or in the selected RoboServer.

You can change the way the information in this view is presented as follows:

- Filter the list of robots in the table by applying various filters. You can filter by robot name, project name, execution ID, and robot URL. Filter matching is case-sensitive and the filter selects those robots that contain the entered text as a substring in either the robot name, project name, execution ID, or robot URL.
- Select the table columns to be displayed using the ▥ menu icon on the right.

By default, the following table columns are displayed for each running or recently completed robot.

Column	Description
Robot name	Name of the robot.
Server	Name of the RoboServer that runs the robot.
Project name	Name of the project that the robot belongs to. View the list of projects in the **Repository** > **Robots** section.
Robot URL	A URL that the robot is identified by. When you build an execute request for a RoboServer, you can specify a file://URL or a Library:/, which indicates whether the robot should be loaded from the file system or the library. • File system URL: `file://C:/Kofax RPA/Robots/Library/Input.robot` • Library URL: `Library:/Input.robot` The execute request for a robot may look similar to the following example: `Request request = new Request("Library:/Input.robot")`
Start time client	Time when the robot was started. The time is displayed in time zone of the browser running the Management Console.
Execution ID	Robot execution ID.
Current step	Step the robot is currently executing.

Column	Description
Status	Robot current status. - Running - Currently running - Queued - Queued for running when possible - Completed - Finished executing on RoboServers. This status is assigned to robots that: - Completed successfully - Completed with errors - Failed - Were forced to stop Robots with the Completed status are removed from the table after one minute from completion.
Optional Columns	
Location code	Code assigned to a step that you can view in Design Studio.
Step execution time	Current step execution time in seconds.
Executed steps limit	Shows the maximum number of steps the robot is allowed to execute. If the limit is reached, the robot is stopped.
KCU wait	Amount of time the robot has been unable to execute because the KCU points (for this second) had already been spent.
Loaded bytes	Bytes loaded during a robot execution.
Last output time	Time when the last extraction was performed.
Emails sent	Number of emails sent by the robot.
Executed steps	Number of steps the robot has executed.
Execution path	Sequence of steps the robot has performed.
KCU point cost	KCU points spent for running the robot. KCU point cost is equal to the KCU usage shown in Design Studio.
Extracted values limit	Upper limit on the number of object extractions. If the robot extracts more objects than indicated by this property, an error message is generated or the robot is stopped.
Execution time limit	Upper limit on the total robot execution time. If the robot does not complete within this time limit, an error message is generated and the robot is stopped. The property value is specified in seconds.
Output count	Number of objects that the robot generated.
Robot library	Type of robot library. The following types exist: - Design Studio robot library - Embedded file-based robot library - Repository robot library - URL file-based robot library - URL folder-based robot library See the Kofax RPA *Developer's Guide* for more information. Also, see Robot Libraries in this help system.

Column	Description
Stop if connection lost	When set, the robot will stop if it loses the connection to the Management Console. The flag is only used when running robots with the Java API. For more information, see "setStopOnConnectionLost" in the *Kofax RPA Developer's Guide.*
Stop on API exception	When set, the robot will stop if it generates an API exception.
Stopping	Shows whether the robot is in the process of shutting down.

Change cluster state

In the **Status** column, you can set a cluster to one of the following states: Running or Paused.

Cluster State	Description
Running	Cluster operates normally with schedules and individual robots executed as expected. Cluster settings are not applied to the RoboServers, so that settings are not changed in the middle of schedule runs.
Paused	Cluster is not allowed to accept new execution requests and the RoboServers is not allowed to finish all current executions. Cluster settings can be changed. For example, a database mapping can be changed.

When you apply new settings to a cluster, the "Apply cluster settings" dialog box with options to proceed is shown.

Apart from Running and Paused, a cluster can have one of the following states:

- Initializing
- Shutting down
- Invalid settings
- Sending settings to RoboServer
- Wait for schedules and robots to finish
- Wait for currently running robots to finish
- Stop all robots and schedules and apply settings immediately

The cluster states are only relevant on a cluster level. As such, they are a way for the Management Console to control when tasks can be executed on clusters. They do not, however, control the individual RoboServers. This means that robots started from the API are not stopped when going into Paused state. Therefore, the settings of a RoboServer can be updated while API robots are running. It is guaranteed, though, that robot settings are not updated during its execution. For example, if databases are updated while one or more API robots are running, the robots uses the databases that were configured when they started. Next time they are run, they use the new settings.

Configure cluster settings

The cluster settings are sent to each RoboServer in the cluster. For example, with the cluster settings, you can configure which databases are available for robots executing on the RoboServers. In the **Action** column for a cluster, you can open the cluster settings from the ⋮ context menu.

Note To send settings to the RoboServers, they must first be applied. After you apply the settings, the "Apply cluster settings" dialog box with options is shown. You need to manually move the cluster to the Running state after applying the settings.

General

Use this tab to define the CRE license distribution and a threshold RoboServer version.

Important Both License and Threshold version options are disabled if KCU licensing is used or Management Console is run in JMS mode. In JMS mode, the license distribution is Static and the threshold RoboServer version is set to 11.1.0.

License Distribution

Select the CRE license distribution mode.

Option	Description
Static	In this mode, CREs are distributed evenly between online RoboServers in the cluster. A CRE is an integral unit, and you cannot split one CRE among multiple RoboServers. For example, if you have six CREs but five RoboServers in a cluster, each RoboServer gets one CRE; therefore, one CRE remains unused. **Note** The number of CREs in a cluster must be equal to or bigger than that of RoboServers. If you assign fewer CREs to a cluster than the number of RoboServers present in the cluster, the cluster is disabled. To adjust the number of CREs, in the **Action** column for the cluster, from the ⋮ context menu for a cluster, click **Assign CRE**. This action opens the Assign license units pane where you can adjust the number of license units and see how many units are available in total and how many remain.
Dynamic	In this mode, RoboServers receive the licenses from the cluster per request. A RoboServer can get as many licenses as it requests if they are available. In this mode, RoboServers communicate only with the Management Console and block other requests, such as API calls. **Important** Dynamic license distribution mode is supported by Kofax RPA version 10.3 and later. Version 10.7 and later support this mode immediately after installation. To use dynamic license distribution, in versions 10.3 to 10.6, install the latest fix pack for the corresponding version. See Enable Dynamic License Distribution Mode in the *Kofax RPA Upgrade Guide*.

Threshold RoboServer version

Defines the Kofax RPA version number, after which legacy robots should not be upgraded beyond. Helps perform a smooth upgrade of robots and block robots from running on a RoboServer they have not been tested on.

Robots with the product version below the threshold value are forwarded to the nearest newer version of a RoboServer. All the robots above the threshold value are distributed between RoboServers in accordance with strict version matching. That is, a robot with a product version higher than the threshold value can only be run on the RoboServer of the same product version.

For example, if a threshold value is set to 10.2.0.0, distribution of several random robots between several random RoboServers uploaded to the cluster looks similar to the following pattern:

	10.2.0.0 RoboServer	10.3.0.0 RoboServer	10.5.0.0 RoboServer	10.7.0.0 RoboServer
9.7.0.0 robot	Matching	-	-	-
10.1.0.0 robot	Matching	-	-	-
10.3.0.0 robot	-	Strict version matching	-	-
10.4.0.0 robot	-	-	-	-
10.7.0.4 robot	-	-	-	-

All robots with product versions below the threshold value are successfully matched with a RoboServer of the equal or higher version.

The 10.3.0.0 robot has a strict version matching as its product version is above the threshold value and it has a RoboServer of the same version in the cluster.

The 10.4.0.0 robot has no matching and cannot be run as its product version is above the threshold value and no RoboServer of the same version in the cluster is available.

The 10.7.0.4 robot has no matching and cannot be run as its product version is above the threshold value and no RoboServer of the same version in the cluster is available.

By default, threshold value is set to the current Management Console version.

Databases

You can add or delete a database on the Databases tab in the cluster settings.

The databases defined on a cluster are available to robots running on the cluster RoboServers via **Repository** > **Database mappings**Database mappings. Database types are defined in the Settings menu.

If a cluster is selected to provide Design Studio with shared databases and database mappings pointing to these databases exist in the repository, the databases that are defined on the cluster are also available in Design Studio.

See Database Connections for more information on database properties.

> **Note** To connect to a specific database, Kofax RPA needs the JDBC driver for this database type. The driver can be downloaded from the provider of the database. For example, `sqljdbc4.jar` that is used with a Microsoft SQL Server database can be downloaded from the Microsoft website. To provide the driver for Kofax RPA to use, upload it to Management Console under **Settings** > **Database drivers** > **Upload driver JAR**.
>
> By default, you can upload driver JAR files to Management Console when you connect to it as admin user from localhost (otherwise, the Upload Driver JAR button is disabled). This setting can be changed in **RoboServer Settings** > **Management Console** > **JDBC Driver Upload**. Important: Start the RoboServer Setting as the user running the Management Console. After making changes in the RoboServer Settings, restart Management Console for the changes to take effect.

Actions

- The **New development database** button creates a database connection to the pre-installed Kofax RPA development database.
- The **New database** button creates a new database connection.
- For users upgrading from pre 9.2.0 versions, it is possible to import databases from the legacy db.properties files in which databases were previously defined. This is done by clicking **Import databases from file**. This action opens a window in which the contents of the file can be pasted.

New Database

1. On the **Databases** tab in the cluster settings, select **New database**.
2. Complete the following database details.
   - Name
   - Host
   - Port
   - Schema

     > **Note** If the concepts of "'schema" and "database" are equivalent (used interchangeably) in your database provider, specify the name of the schema in this field. Otherwise, specify the name of your database. In this case, the default schema is used.

   - Type
   - User name
   - Password
   - Max active connections
   - Max idle connections
3. Click **OK**.

Proxy servers

You can configure a list of proxy servers to use on the RoboServers. If only a single proxy server is defined, that one is used. If a list of proxy servers is defined, for the first connection a random proxy server is selected.

See Use Proxy Services for more details on using proxy and Design Studio, Proxy Servers for more information on proxy server properties.

You can import proxy servers from legacy properties files by selecting the proxy servers category and clicking **Import proxy servers from file**. This action opens a window in which you can paste the contents of the file to import. The file must have the format described at the bottom of the Proxy Servers topic.

1. On the **Proxy servers** tab in the cluster settings, select **New proxy**.
2. Complete the following proxy server details.
   - Host
   - Port
   - User name
   - Password
   - Exclusions (excluded host names)
3. Click **OK**.

Logging

You can enable or disable logging options for cluster RoboServers.

To enable logging options for cluster RoboServers, navigate to **Settings** > **RoboServer log database** and select **Log to database**. See RoboServer log database for more information on logging.

Log to database

If database logging is enabled, the RoboServers logs to the RoboServer log database defined in the Design Studio settings. If **Log robot input to database** is selected, the robot inputs are also logged to the database.

Log to Log4J

Using log4j2, you can control where the log goes using an ordinary log4j2.properties file. The log4j2.properties file is located in the Configuration directory of the Kofax RPA Application Data Folder, for example, `C:\Users\name.lastname\AppData\Local\Kofax RPA\11.1.0\Configuration`. See "Application Data Folder" in the *Kofax RPA Installation Guide* and "Audit Log for Management Console" in the *Kofax RPA Administrator's Guide*.

The default log4j2.properties file logs all robot run information, robot messages and server messages into a file. The logs are placed in the Logs folder in the application data folder. More advanced setups include storing in the Windows event log, rotating files, and the syslog.

When this option is enabled, RoboServer logs using three different loggers.

Logger	Description

`logger.servermessagelog.name = kapow.servermessagelog`
`logger.servermessagelog.level = {TRACE,INFO,DEBUG,ERROR...}`
	Used for server logging that is not tied to a particular robot run.

`logger.robotrunlog.name = kapow.robotrunlog`
`logger.robotrunlog.level = {TRACE,INFO,DEBUG,ERROR...}`
	Used to log information about starting and stopping robots, including the execution time.

Logger	Description

`logger.robotmessagelog.name = kapow.robotmessagelog`
`logger.robotmessagelog.level = {TRACE,INFO,DEBUG,ERROR...}`
	Used to log robot messages, such as information related to error handling. Also includes profiling, if it is set to be output to the log.

If **Log robot input to Log4j** is selected, the robot inputs are also logged using the log4j2.properties file.

Log to email

When enabled, an email is sent whenever a robot logs an error message or the server has an error message.

Property	Description
Email from	The "from" address to be used on the emails.
Email to	Who should receive the email. You can add multiple addresses separated by commas.

Profiling

If you enable profiling for robots, you can see the execution time for the individual steps and the entire robot. This can be useful if you have a slow running robot and want to improve performance.

Profiling is configured using the following properties.

Property	Description
Output type	If you choose **Summary**, only one statement summarizing the execution is written to the profiling log for each robot execution. If you choose **Detailed**, a detailed statement is written to the profiling log for every step executed in a robot, provided the execution time of the step is above the threshold defined by the **Threshold** property. The detailed log contains the following information that can be useful to analyze and improve robot performance. • Wait time for the remote device in milliseconds. • Name of the executed step. For the Desktop Automation steps, the statements are prefixed with **DA**. • Step execution time in milliseconds. • KCU points cost. • Wait time for available KCU points. • Robot execution time in milliseconds. • Robot loading time in milliseconds. • Summary of the total wait time.
Output target	Controls where the profiling information is sent.
Threshold	This threshold defines the smallest execution time (in milliseconds) for a step to warrant inclusion in profiling information.
Log URL	If checked, the page URL is printed before the page load wait time.

Robot execution

Use this tab to define properties for RoboServer robot execution. Specify how many robots can run concurrently on individual RoboServers, and how many can be queued. If you are using the CRE type of license, the maximum number of robots that can run concurrently in clusters is defined by the license.

Property	Description
Max concurrent robots	Maximum number of robots allowed to execute concurrently on each RoboServer. Lower the number if your robots are CPU or memory intensive and you often reach the memory threshold.
Max queued robots	Maximum number of robots allowed to reside in the queue on each RoboServer. If robots are started via API calls, the value determines how many robots can be queued on each RoboServer. **Note** This setting is only used with deprecated methods such as SocketBasedObjectRQLProtocol in the Java API to call RoboServers directly without calling the cluster. We recommend that you do not use deprecated methods.

Projects

Use the Projects section in the Admin menu to configure and create projects for the Management Console.

Projects are a way to segment robots, types, snippets, resources, schedules, and other work assets. Robots have access only to types, snippets, and resources contained in the project they belong to. Also, names of robots, types, and other objects need to be distinct within a project.

By default, Management Console contains a single project called "Default project."

You can change the way the information for each project is presented as follows:

- Select the table columns to display for a project using the ▥ menu icon on the right.
- Refresh the displayed information by clicking the ↻ refresh icon on the right.

By default, the following table columns are displayed for each project.

Column	Description

Name	Name of the project. Project name must be unique.
	The project name is used as a foreign key in the log tables to determine who has permission to view the log files. If you rename a project, all existing robot runs and robot messages belonging to this project must be updated to reflect the new name. Otherwise, they are not shown when the logs are filtered by project.
	If Management Console is connected to the logging database, when you rename a project, Management Console automatically renames the robot runs/messages entries in the log database. However, if it is not connected, or the connection is lost during the update, the administrator must manually run the following SQL to update the log tables.
	`UPDATE ROBOT_RUN SET projectName = '<newName>' where projectName = '<oldName>';`
	`UPDATE ROBOT_MESSAGE SET projectName = '<newName>' where projectName = '<oldName>';`
	Where `<oldName>` is the old project name, and `<newName>` is the new project name.

Description	Description of the project.
Authenticate REST	Indicates if the authentication for REST/SOAP requests is enabled.
Optional Columns	
Imported by	User name of the user who imported the project or restored it as part of a backup.
Imported at	Date when the project was imported or restored as part of a backup.
Permissions	Any configured project permissions for this project.
REST cluster	Cluster used to execute robots for this project when robots are invoked as REST services.

Create New Project

1. To create a new project, click the plus sign in the upper left corner.
   Several new tabs appear.
2. On the **Basic** tab, specify a name for the project and its description.
3. On the **Permissions** tab, set project permissions.
   a. To add a permission, click the plus sign.
   b. Specify a project role and optionally a security group that it belongs to.
      For a description of project roles, see Users & groups.

   When deploying Management Console on a standalone web container, you can also configure project permissions for users based on their group membership (for example, LDAP groups). For more information, see the "Tomcat Management Console" section of "Project Permissions" in the *Kofax RPA Administrator's Guide*.

4. On the **Services** tab, you can optionally select a service cluster to execute robots for this project when robots are invoked as REST services.
   - **Use only service cluster in project**: Service cluster is a cluster used to run REST services in the current project. REST services always use the selected service cluster. If you select this option, Management Console hides all other clusters and use the service cluster for all schedules, when generating code, and when running robots from the robot menu. When you select this option, you also need to select a **Service cluster**.
   - **Authenticate REST/SOAP requests**: By default, REST/SOAP services are protected by basic authentication. When calling the services directly from a browser using XMLHttpRequest, disable the authentication, as you would otherwise be exposing login credentials in the JavaScript source files. When calling REST/SOAP services from a programming language like Java, Ruby, C#, and other, enable the authentication to protect the services, assuming that you can keep the credentials stored in a secure manner.
   - **Access-Control-Allow-Origin**: Specify a header for the client to be allowed to process a resource from another domain.

     There are certain restrictions when calling a REST/SOAP service from a browser, unless the service is located on the same web server as the web page from which it is invoked. When calling a REST/SOAP service from another domain (referred to as CORS Cross-Origin Resource Sharing), you have to include certain headers for the client to be allowed to process a resource from another domain. The Access-Control-Allow Origin is one such header. If you call a REST/SOAP service in a cross-domain fashion, you must specify the domain, from which the page that generated the request was loaded. If a page on http://example.com contains a page with JavaScript that generates a request to a service located on http://kofax.com, then the service response from http://kofax.com must contain the header "Access-Control-Allow-Origin: http://example.com" or built-in security mechanisms in the browser will prevent it from processing the cross-origin response. You may use * as a wildcard, which means that any domain can invoke your REST/SOAP service in a cross-domain fashion.
5. On the **Repository** tab, you can optionally configure the Robot Lifecycle Management feature that enables you to control work objects in the Git version control system. For a detailed example of using this feature, see the *Kofax RPA Best Practices Guide for Robot Lifecycle Management*.

   Here, specify the following information:

   a. **URL**: Type the path to the Git repository.

      Example: /gitrepos/example.git/

   b. **Branch**: Type the name of the branch to use.

      Example: If you have a setup with two Management Consoles, one for production and one for development, you can have a dedicated "development" branch.

   c. To make the repository the only source for object changes, select **Read-only**. We recommend that you select this option to avoid any changes to objects belonging to the synchronized project in the production Management Console.

   d. To enable the configuration specified above, select **Enable configuration**.

   e. Under **Objects to synchronize**, select the objects to include in the synchronization.
6. Click **Submit** to save the changes.

Additionally, you can perform the following actions on projects from the ⋮ context menu:

- **Edit**: Contains the same fields as when creating a new project.

To delete a project, select it and click the 🗑 icon.

> **Note** If you delete a project, all of the robots, types, snippets, resources, and schedules associated with it are deleted as well.

High availability nodes

This section lists all configured high availability nodes. For information on how to enable high availability mode and configure multiple Management Consoles to work together as a cluster, see "High Availability" in the *Kofax RPA Administrator's Guide*.

You can change the way the information for high availability nodes is presented as follows:

- Select the table columns to display using the ▥ menu icon on the right.
- Refresh the displayed information by clicking the ↻ refresh icon on the right.

By default, the following table columns are displayed for all configured nodes.

Column	Description
Node ID	ID of the configured Management Console node.
Interface	IP address of the Management Console node including port.
Status	Status of the node: Running, Shutting Down, or Shut Down.
Connected to	Shows "true" or "false" depending on whether you are currently connected to this node.

Devices

This section shows available automation devices that are registered with the Management Console. For more information, see Automation Device Mapping.

To register an automation device in Management Console, specify the Management Console details in the Desktop Automation Service. Leave the Single User option empty.

The following information is displayed for each cluster/device.

Column	Description
Cluster	Name of the cluster.
Device	Either IP address or name of the automation device and a port used to connect to the device.

Column	Description
Status	**For clusters** Current cluster state. See Change cluster state for more information. **For devices** Current status of the device. The status can be as follows. • Available: The Desktop Automation Service is running and no one is connected to the device. • In use: The Desktop Automation Service is running and either a Design Studio or a RoboServer is connected to the device. Note that only one connection can be established with the automation device. • Offline: Either the remote device is off or the Desktop Automation Service is not started.
Labels	Label of the device mapping. You can use the labels to filter the devices to automate with your robot.
User name	Name of user running the Desktop Automation Service.
Domain	Domain name of user running the Desktop Automation Service.
Version	Kofax RPA version last used when editing the robot.
Execution ID	Robot run ID using a given automation device.
Robot name	Name of the running robot.

Users & groups

Use this section to manage users and groups that can be given access to the Management Console and projects. The security model is role-based; after you create a user, you need to add the user to one or more groups associated with specific roles in one or more projects.

Tip We recommend that you create groups first, because a user cannot log in until the user is added to a group that is granted a Viewer role inside at least one project.

The section contains two tabs:

- The Users tab helps you create new users and edit, remove, and reset passwords for selected users.
- The Groups tab helps you create, remove, and edit groups.

Note User and group names in Kofax RPA must follow the Rules for Logon Names for Microsoft Windows. Such as the names must not contain the following characters:

" / \ [ ] : ; | = , + * ? < >

For information, see Creating User and Group Accounts on technet.microsoft.com.

You can change the way the information for each tab is presented as follows:

- Filter the lists in the table by applying filters in the Filter text field. See Filtering for more information.
- Select the table columns to be displayed using the ▥ menu icon on the right.
- Refresh the displayed information by clicking the ↻ refresh icon on the right.

Users

By default, the following table columns are displayed for each user.

Column	Description
User name	Name of the user.
Full name	Full name of the user.
Email	Email address of the user.
Login count	Number of sessions done by this user.
Last login	Date and time when the user last logged in.
Last IP address	Last IP address that the user logged in from.
Groups	Groups that the user belongs to.

Create New User

Tip If you do not have required groups yet, we recommend that you create them first, because a user cannot log in until the user is added to a group that is granted a Viewer role inside at least one project.

1. On the **Users** tab, click the plus sign.
   The "Create new user" dialog box appears.
2. Specify a user name, password, full name, and email for the user and then select a group or multiple groups that the user will belong to.
3. Click **OK**.
   The user appears in the table.

You can edit a user from the ⋮ context menu.

Groups

By default, the following information is displayed for each group.

Column	Description
Group name	Name of the group.
Description	Description of the group.
Number of users	Number of users contained in the group.
Used in projects	Projects using this group.

Create New Group

1. On the **Groups** tab, click the plus sign.
   The "Create new group" dialog box appears.
2. Specify a group name, description, and select the users to be included in this group.
3. Click **OK**.
   The group appears in the table.

You can edit a group from the ⋮ context menu.

Reset Password for User

1. On the Users tab, select the user and click in the upper left corner.

   The "Reset password for" dialog box appears.

2. Type the new password, type it again to confirm, and then click **OK**.

   You may select to send an email so the user receives a notification about the password change. The "From address" needs to be pre-configured to send notifications.

Built-in roles

Management Console provides some built-in roles that users can have. Roles are mapped to a user of a security group. User permissions are calculated based on the roles that are mapped to security groups the user is a member of. You can modify built-in roles or add additional roles.

- **Project Administrator**: A user with this role administrates one or multiple projects and has a right to assign a role to a group for these projects. This user has a view right to view RoboServer and cluster settings without changing them. Project Administrator is not a member of the RPA Administrators group (for more information, see later in this section).
- **Developer**: Developers have a right to upload, download, and view all resource types in the repository. A user with this role can create, edit, and delete schedules, run robots, view run logs and clusters.
- **Viewer**: Viewers have similar view rights as developers and the rights to change or run anything.
- **API** (This user logs in only as a service authenticating via the API): A user with this role can use the repository API to read from and write to the repository. A user with this role cannot run robots using REST but is able to run robots using RQL.
- **RoboServer** (This user logs in only as a service authenticating via the API): A restricted user that can only read from the repository. This role is used by RoboServers when accessing a cluster, retrieving repository items, and requesting passwords from the password store.
- **Kapplet Administrator**: A user who can create, view, run, and edit Kapplets.
- **Kapplet User**: A user who can view and run Kapplets. A user with this role cannot access Management Console if this user has no other rights.

  For more information on Kapplet user roles, see Kapplets User Roles.
- **Password Store client**: A user with this add-on role can access the password store. The role is provided on top of other roles, just like the Developer role. This role only provides access to the password store in Management Console.
- **DAS Client User** (This user logs in only as a service authenticating via the API): This is a user that is created for remote Desktop Automation Service (DAS) clients, and can only access the DAS API. The DAS client user has a right to announce a DAS to Management Console, and retrieve DAS configuration.
- **VCS Service User** (This user logs in only as a service authenticating via the API): The version control service user is granted a special set of rights for the Synchronizer. This role has a right to add, modify, and delete resources. This is the only role that can deploy on behalf of another user to use the "deployer" feature in the version control service.
- **Process Discovery Client** (This user logs in only as a service authenticating via the API): This role allows Process Discovery components interact with Management Console.

Built-in "admin" user

admin is a superuser that has access to everything. It is not a member of the RPA Administrators group and cannot be a member of any group. The default admin user password is available (user name - `admin`, password - `admin`). You can change the admin user password as described in "Reset password for user" in this section.

In an LDAP integration setup, the **admin** group is defined as part of the LDAP configuration. **admin** can then log in and define which LDAP groups should be mapped to the Developer, Project Administrator, RoboServer, and other roles.

In an internal user setup, the **admin** user is created at first start and can then login and create Administrators, Developers, and other users.

Built-in "admin" user special rights

Beside being the initial user, **admin** also has special rights, such as:

- In the RoboServers section, **admin** can click a RoboServer node and request a stack trace from the corresponding RoboServer.
- Only **admin** can create and import backups.
- In the password store, **admin** can move passwords to another project.

Built-in group

RPA Administrators: Users belonging to this group have all rights for all projects (excluding special **admin** user rights), such as creation of new administrators and users in any project. To make a user an administrator, the user has be added to this group.

Note

- The RPA Administrators group is visible when the internal user management is enabled, and it is empty by default.
- When restoring a backup created in version prior to 10.7, users with Administrator role become members of the RPA Administrators group.

User management principles

The following information can help you understand some Kofax RPA user management principles.

There are two ways to run Management Console: embedded in a RoboServer with any license and on a standalone Tomcat server (requires enterprise license). For information about Management Console on Tomcat, see "Tomcat Management Console" in Kofax RPA *Administrator's Guide*.

When Management Console runs in embedded mode, user management is turned on by default to mitigate the potential security risk of having an unauthorized access to a Management Console from other computers. The default administrator name and password are set as follows:

- User name: `admin`
- Password: `admin`

To change the admin name and password, do the following:

1. On the **Users** tab, select the user and click 🔒 in the upper left corner.

   The "Reset password for" dialog box appears.

2. Type the new password, type it again to confirm, and then click **OK**.

   You may select to send an email so the user receives a notification about the password change. The "From address" needs to be pre-configured to send notifications.

   Once you change the password, update the login information in the RoboServer settings.
3. Start the RoboServer Settings application either by clicking RoboServer Settings on the Windows Start menu or by double-clicking `RoboServerSettings.exe` in the `/bin` subfolder of the Kofax RPA installation folder.
4. On the **General** tab under **Register to a Management Console**, specify the new password for Management Console to register to. Click **OK** to save the changes.
5. Restart Management Console for the changes to take effect.

   Depending on your license and the way you run Management Console, you can manage user access as follows:

   - Internal user management: Available in both Embedded and Standalone mode
   - External user management (LDAP, SAML, or CA Single Sign-On): Only available in Standalone mode with enterprise license

   > **Note** When you run the Enterprise version on a Tomcat server, Management Console is always in the multi-user mode and you can choose to manage your users either in Management Console (like in the embedded mode) or get the user credentials from your corporate LDAP server.

   Check for login attempts

   By default, the check for number of login attempts made by a user and wait time before the next attempt is disabled. To enable this functionality, edit the following section in the `authentication.xml` file located in: `<Tomcat installation folder>\WebApps \Management Console\WEB-INF\spring`

```
   <bean id="loginAttemptService"

 class="com.kapowtech.scheduler.server.spring.security.LoginAttemptService" lazy-
init="true">
       <constructor-arg type="boolean" value="false"/>
       <constructor-arg type="int" value="3"/>
       <constructor-arg type="int" value="10"/>
   </bean>
```

   Set the first value to **true**. The second and third values are for the number of login attempts (3 in this example) and the wait time in minutes before the next attempt (10 in this example), respectively.

Backups

Use the Backups section to copy all or part of the Management Console configuration into a file to back up or export data. If necessary, you can use the file to perform a restore or import operation.

> **Note** You can restore backups created in Kofax RPA version 9.7.10 or later. For earlier versions, contact Kofax RPA technical support for assistance in migrating your data.

It is important to understand the difference between creating/restoring a backup and exporting/importing a project.

Creating and Restoring a Backup

A backup contains all of the Management Console configuration, including all schedules and all projects in the repository, and it can be restored only in its entirety. It may be used to restore the system after data loss, or when upgrading to a later version of Kofax RPA. In the latter case, you will create a backup of the old instance of the Management Console and restore it into the new instance.

> **Note** After you restore a Management Console backup that has user management off, use the default superuser credentials to log in to the Management Console (user: `admin`, password: `admin`). If you restore a backup from the Management Console with enabled user management, the default `admin` user is replaced with a superuser from the backup. Use credentials specified in the restored Management Console.

Exporting and Importing a Project

Use the Export Project feature to create a file with information pertaining to a single project. This comprises the schedules, robots, types, resources, OAuth, and Robot File Systems within the project. Such a file may be used to copy schedules, robots, and so on, from one Kofax RPA system to another. For example, when moving from a test environment into production. It is possible to import into an existing project on the target system; in this case, items from the file are merged into the project, overriding present items when names match. It is also possible to do a selective import that includes only some items.

Create a Backup

1. To create a backup, in **Admin** > **Backups**, click **Create backup**.
2. Click **Create**.
3. When the backup is ready, it is downloaded to your computer.

Export a Project

1. To export a project, in **Admin** > **Backups**, click **Export project**.
2. Select the project to export.
3. Click **Export**.
4. When the export project is ready, it is downloaded to your computer.

Export a Project by Name

You can also export a project by name using the following URL. For example, it can be useful if you need to back up a project using API.

```
http://user:password@localhost:8080/ManagementConsole/secure/BackupProjectByName?
projectName=MyProject
```

In this URL, replace the following parts with you actual data:

- **user:password**: User credentials to access the Management Console.
- **localhost:8080/ManagementConsole**: URL to the Management Console.
- **MyProject**: Name of the project to export.

Restore a Backup

Note Starting from Kofax RPA 9.5.0, the threshold for cleanup is changed from a number of records to a number of days; thus the number of days is set to 10 in case of an old backup with 50000 or more records (the old default). If the number of records is less, the number of days is scaled back accordingly.

1. To restore a backup, in **Admin** > **Backups**, click **Restore backup**.
2. Click the 📎 paper clip sign to select a backup file to restore, select the file on your computer, and then click **Open**.
3. Select **Merge** or **Reset**.
   - Choose **Merge** to merge the settings of the backup into the current version of the Management Console.

     Those settings that are missing in the backup, that is, settings that did not exist in an earlier version when the backup was created, keep their custom value.
   - Choose **Reset** to reset all the settings before restoring a backup.

     Those settings that are missing in the backup, that is, settings that did not exist in an earlier version when the backup was created, are reset to their default value.
4. Click **Restore**.
5. When the restore is complete, click **Close**.

Import a Project

1. To import a project, in **Admin** > **Backups**, click **Import project**.
2. Select the items to import.

   Available options include:
   - Import schedules
   - Import robots, types and snippets
   - Import resources
   - Import OAuth
   - Import permissions
   - Import Robot File Systems
   - Import trigger mappings
   - Import email triggers
3. Click the 📎 paper clip sign to select a project file to import, select the file on your computer, and then click **Open**.
   - If a project with the same name already exists and you want to replace it with the newly imported project, select **Delete project if exists**. If you do not select this option, the changes are merged into the existing project.
4. Click **Import**.
5. When the import is complete, click **Close**.

License

Use the License section to enter a new license or view the current license. Management Console has room for two license keys: A Production and a Non-Production key. If your production environment is completely separated from your development/testing environment, you need to install two Management Consoles: One containing the Production license key, and the other containing the Non-Production license key. In mixed environments, a single Management Console should contain both keys.

On the Design Studio seats tab, you can see information on the Design Studio seats currently in use. The table lists Design Studio clients who have received a license from this Management Console and the last IP address that the user logged in from.

Here you can also select the table columns to display using the ▥ menu icon and refresh the displayed information by clicking the ↻ refresh icon.

Important A Non-Production license has the following constraints:

- Edition: Enterprise
- High availability: Yes
- Max number of schedules: Unlimited
- Max number of robots: Unlimited
- Max number of users: Unlimited
- Number of production CRE: Unlimited
- Number of non-production CRE: 100
- Number of Design Studio seats: Unlimited
- Kofax Analytics for RPA: Yes
- Desktop Automation: Yes

Note Some items in the list are shown only if you have an appropriate license. For example, if you have a license for Analytics, the **Kofax RPA Analytics: Yes** item appears in the "About the current license" table.

Settings

Use the Settings menu to configure additional Management Console settings.

Additional options that affect how you connect to Management Console (such as port numbers and security settings) are configured with the RoboServer Settings application as described in "Embedded Management Console Configuration" and "RoboServer Parameters" in the *Kofax RPA Administrator's Guide*.

General

In this section, you can configure database settings, proxy server, Robot File System server, and other.

Login

On this tab, you can configure to send email notifications to users to restore lost or forgotten passwords.

Prerequisites: An SMTP server must be pre-configured. See **Settings** > **General** > SMTP server.

RoboServer authentication

On this tab, you configure the credentials (user name and password) that Management Console uses when running robots on the RoboServers belonging to the configured clusters. The Management Console uses the same set of credentials for all RoboServers. These credentials must match the configuration as described in the "Request Authentication" section in the *Kofax RPA Administrator's Guide* for all of the configured RoboServers.

RoboServer purge

On this tab, by selecting the option, you can configure Management Console to automatically remove offline RoboServers from the list of RoboServers in the **Admin** > **RoboServers** section. If a RoboServer goes online, it automatically registers with a Management Console and appears in the list of RoboServers.

RoboServer log database

On this tab, you can configure the logging database used by Management Console to store schedule runs, schedule messages, robot runs, robot messages, robot tags, Desktop Automation Service messages as well as all RoboServers belonging to clusters where database logging has been enabled in the cluster settings.

Note To use a database for logging, you need to prepare your database server by either creating a new database (schema), or simply making sure an existing database is available. You need to obtain a user name and password with rights to create tables, drop tables, create indexes, drop indexes, select, insert, update, and delete in the database.

Both Management Console and RoboServer create the log tables automatically when they are started (if the tables do not already exist). However, you may also create them using the Scripts for creating database tables.

Select **Log to database** and configure the logging database.

Property	Description
Host	Host name of the database server, which can be an IP address, the fully qualified domain name (myhost.kofax.com), or a work computer name (PC1X1XXX).
Port	Port of the database server.
Schema	Name of the database schema (or catalog).
Type	Type of database, such as Oracle. The different types of databases are configured in Management Console and are provided automatically when Design Studio is started.
User name	User name for the database.

Property	Description
Password	Password for the database.

To test the current configuration, click **Test**.

Note This action only tests the connection to the database. It does not test whether you have the proper permissions in the database.

Click **Save** when finished.

The following cleanup thresholds can be configured for the RoboServer log database.

Property	Description
Keep robot and schedule statistics for (day)	Specify the number of days to keep the robot and schedule statistics. According to the cleanup settings you define, the old data is deleted on a daily basis.
Max messages in robot run	Specifies the maximum number of messages in a single robot run. The robot message logging stops for the specific run when the number of robot messages exceeds this threshold. The cleanup deletes the oldest robot runs and the messages for the deleted runs. If you experience performance problems with the log database, you can lower this threshold. To store more historic messages, you can increase this threshold.

Scripts for creating database tables

The SQL scripts for creating and dropping tables in your database are located in the `documentation\sql` directory in your Kofax RPA installation directory. For example, `C:\Program Files (x86)\Kofax RPA 11.1.0 x32\documentation\sql` on the Windows system. The name of the script file includes the name of the database the script is intended for.

Note SQL scripts are installed together with Kofax RPA and if you install Design Studio.

The `sql` directory contains four subdirectories with different scripts as follows:

- `altosoftsession`: Scripts for creating and dropping tables for a single sign on with Altosoft.
- `logdb`: Scripts for creating and dropping logdb tables.
- `mc`: Scripts for creating and dropping Management Console tables.
- `statistics`: Scripts for creating and dropping statistics tables required for producing reports in Kofax Analytics for RPA.

Important The IBM DB2 database must have a table space with a page size of at least 8192 KB to create all tables.

Management Console uses a third party scheduling component called Quartz. Quartz also requires a number of tables that must reside among the other platform tables. These tables are also created automatically when Management Console starts, or may be created manually using the scripts.

The following is a Quartz verification script.

```
select count(*) from QRTZ_SIMPLE_TRIGGERS;
select count(*) from QRTZ_BLOB_TRIGGERS;
select count(*) from QRTZ_CRON_TRIGGERS;
select count(*) from QRTZ_TRIGGER_LISTENERS;
select count(*) from QRTZ_CALENDARS;
select count(*) from QRTZ_FIRED_TRIGGERS;
select count(*) from QRTZ_LOCKS;
select count(*) from QRTZ_PAUSED_TRIGGER_GRPS;
select count(*) from QRTZ_SCHEDULER_STATE;
select count(*) from QRTZ_JOB_LISTENERS;
select count(*) from QRTZ_TRIGGERS;
select count(*) from QRTZ_JOB_DETAILS;
```

Analytics database

On this tab, you can set up your connection to the database used by the Kofax Insight Analytics dashboards.

Select **Log the statistics to Analytics database** and configure the Analytics database in the same way as the RoboServer log database. You can test the configuration. Click Save when finished.

The following cleanup threshold can be configured for the Analytics database.

Property	Description
Keep RoboServer statistics for (day)	Specify the number of days to keep the statistics for. According to the cleanup settings you define, the old data is deleted on a daily basis.

You need to prepare your database server by either creating a new database (schema), or simply making sure an existing database is available. You must obtain a user name and password with rights to create tables, drop tables, create indexes, drop indexes, select, insert, update, and delete in the database. For more information, see RoboServer log database.

Note The availability of the Analytics functionality depends on your license. If you have a license for Analytics, the Analytics database section is present under the Settings menu in Management Console.

SMTP server

On this tab, you can define SMTP server settings to be used by RoboServers to send email notifications, such as password change notifications.

The SMTP server is configured using the following properties.

Property	Description
Host	SMTP server host name.
Port	SMTP server port.
User name	If the SMTP server requires authentication, enter the user name.

Property	Description
Password	If the SMTP server requires authentication, enter the password.
Encryption	• **NONE**: Credentials and email are sent in clear text. • **TLS**: TLS encryption is used. This option only works if the SMTP server has a trusted certificate (if the server has a self-signed certificate, it must be exported and imported into the JVM's truststore using the keytool utility). Uses the STARTTLS SMTP extension. • **SMTPS**: SMTP over SSL. A secure channel of communication is established, over which an email is sent. This is rarely supported by **SMTP servers**, but works even if the server uses a self-signed certificate.
From address	Email address to send notifications from.

Proxy server

On this tab, you can specify the proxy server through which Management Console connects to external servers, such as when generating OAuth security tokens for accessing external APIs.

Property	Description
Use proxy server	Select to use a proxy server. When not selected, a direct connection is used.
Host	Proxy server host name.
Port	Proxy server port.
User name	If the proxy server requires authentication, enter the user name.
Password	If the proxy server requires authentication, enter the password.

Robot File System server

On this tab, you can configure the Robot File System server that handles all mappings to drives from RoboServers and Design Studio instances. Robot File System server is configured using the following properties.

Property	Description
Use Robot File System server	Select to enable the Robot File System server usage.
URL	URL to the Robot File System server. For example, `http://myserver.mydomain:8080/rfs`

Base URL

Use this tab to define the Management Console base URL. Typically, the base URL is configured automatically, and it is not necessary to change it.

To make the documentation available for use in offline mode, select **Use local documentation**. In **Local documentation base URL**, specify the URL to the Tomcat website containing the documentation.

Example: `http://localhost:8080/ManagementConsoleHelp/`

When **Use local documentation** is selected, the Management Console uses the offline version of the documentation by default, even if an active Internet connection exists.

For more information on the offline documentation, see the *Kofax RPA Installation Guide*.

DAS configuration

On this tab, you can set up the interval in milliseconds for the Desktop Automation Service to ping Management Console. At the first ping Management Console passes the information to the Desktop Automation Service on how often it should ping Management Console.

Design Studio

To make mapped databases and/or Desktop Automation Services from a specific cluster available in Design Studio, select this cluster here.

Shared databases are sent to all Design Studio clients being activated by getting the license from the Management Console.

- To make databases from a certain cluster available in Design Studio clients, select the cluster in the drop-down list and make sure a database mapping exists that points to this database. You can check database mappings in the **Repository** > **Database mappings** section.

  Select **All clusters** to make databases from all clusters available in Design Studio clients. If **No cluster** is selected for databases, Design Studio will only receive database types and drivers.

  > **Tip** To define a special set of databases that are available for Design Studio, you can create a special cluster. This approach may be useful if you prefer not to send production databases to Design Studio.

- To make Desktop Automation Services from a certain cluster or all clusters available in Design Studio clients, in the drop-down list, select this cluster or **All clusters**, respectively. If **No cluster** is selected, Desktop Automation Services will not be available in Design Studio clients.

  > **Note** Users with the RoboServer role and users belonging to the RPA Administrators group have access to all Desktop Automation Services regardless of the selected cluster.

The default cluster named Production is automatically created for shared databases and Desktop Automation Services.

For information on database mappings in Design Studio, see Map Databases.

Databases

To simplify the user interface, the databases are configured on a cluster, and a link to the RoboServers section is provided here.

Database types

In this section, you can define one of the following database types: Apache Derby (Development Database), Sybase Adaptive Server, PostgreSQL, Oracle, MySQL, Microsoft SQL Server, and IBM DB2.

To edit an existing database type, select the respective tab. To create a new type, on the **New type** tab, specify the following properties and click **Save**. For an example, see Add database Microsoft SQL Server database type.

Property	Description
Name	Name identifying the database type.
JDBC driver	JDBC driver class of the driver (the driver must be uploaded under Database drivers).
Connection URL template	Template string defining how connection URLs for databases of the given type look. Possible variables to use in the template string are the following: **${ServerName}** Defines the server name (host) of the database server. **${Schema}** Defines the schema (or database/catalog depending on database vendor) of the database. An example of a connection string template is `jdbc:mysql://${ServerName}/${Schema}`. This string defines the connection string for a MySQL database running on the default port (no port is specified), on the server given by ${ServerName} and using the schema given by ${Schema}. The variables are given values when databases of the given type are created (see the cluster databases section).
SQL flavor	Select the database type from the list. For more information on supported databases, see the *Kofax RPA Technical Specifications* document on the Kofax RPA 11.1.0 documentation site.

The database types are also sent to Design Studio clients.

Note Kofax RPA does not support adding other database types than specified in the "SQL flavor" list. However, you can modify the database types to add a type that defines a database server running on a non-standard port or requiring a different authentication method.

Add database Microsoft SQL Server database type

Use this procedure to add a modified database type in Management Console. As an example, see how to add a Microsoft SQL Server that requires Windows Integrated Security.

1. In Management Console, select **Settings** > **Database types** .
2. On the **New type** tab, specify the following values.
   - Name: Microsoft SQL Server (Integrated Security)
   - JDBC driver: com.microsoft.sqlserver.jdbc.SQLServerDriver
   - Connection URL template: jdbc:sqlserver://${ServerName};databaseName=${Schema};integratedSecurity=true
   - SQL flavor: Select **Microsoft SQL Server**.
3. Click **Save** to add the database type.
4. Download the "Microsoft JDBC Driver 4.0 for SQL Server" from the Microsoft website and extract it to a folder on your disk.

5. Copy `sqljdbc_auth.dll` to the `\lib` directory in the Kofax RPA installation folder. The file is located in the extracted folder under `Microsoft JDBC Driver 4.0 for SQL Server \sqljdbc_4.0\enu\auth\x64` or `\86` folder.

   If there are RoboServers running on other computers, the `sqljdbc_auth.dll` must be installed on each one.
6. Upload the `sqljdbc.jar` file to Management Console. Select **Database drivers** under **Settings** and click **Upload driver JAR**. Browse to the corresponding JDBC driver. The .jar file is located in the extracted folder under `Microsoft JDBC Driver 4.0 for SQL Server\sqljdbc_4.0\enu`.
7. Restart Kofax RPA Management Console.

You can now connect to the database that uses the newly created database type in the **Admin** > **RoboServers** section in the Management Console.

Notes on Microsoft SQL Server for Windows Integrated Security

- If you do not copy the required database .dll file, use the 32-bit version of the auth .dll instead of the 64-bit, or if you do not restart Management Console, you may get the following error message: "Error connecting to the database: This driver is not configured for integrated authentication."
- If the server is not part of the domain, you may get the following error: "Error connecting to the database: Login failed. The login is from an untrusted domain and cannot be used with Windows authentication."

See the following topics for more details about using databases in Kofax RPA.

- Database types
- Database drivers
- Databases
- Interact with Databases

Database drivers

In the "Database drivers" section, you can upload new database JDBC drivers and see the list of the other uploaded JDBC drivers. These drivers are required by the various databases types.

> **Note** Only one database driver of each type should be uploaded to the same Management Console. For example, it is not supported to have two different MySQL drivers at the same time in the same Management Console. If you upload several drivers of the same type, only the first uploaded driver is used.

As with database types, uploaded database drivers are also sent to Design Studio clients. Note that if you run your Management Console on, for example, a MySQL database, you need to provide Tomcat with the MySQL driver. This means that MySQL databases work when used in Management Console in the Log view or when testing the connection. However, for the MySQL databases to also work on all RoboServers, it is necessary to upload the MySQL driver here so that it can be sent to the RoboServers and Design Studio.

Note The Derby JDBC driver is not distributed with the Enterprise Management Console. See Apache Derby web site for Derby JDBC driver download information. We recommend using MySQL or another enterprise-class database with your Enterprise Management Console.

Hint

On certain database types, you may need to tweak parameters or settings to store files larger than a certain size. For example, on MySQL databases, you may have to increase the value of the "`max_allowed_packet`" variable, which in many installations is set to 1 MB (meaning that database drivers larger than 1 MB cannot be stored). Consult the documentation for your database if you encounter problems when uploading drivers. To help identify any problems, you receive an error message, and the Management Console log contains further details.

Security Note

By default, only the admin user is allowed to upload JDBC drivers while accessing Management Console from the localhost. Otherwise, the **Upload driver JAR** button is disabled. If you are running Management Console in embedded mode, you can change this behavior in the **RoboServer Settings** application. See the "Embedded Management Console Configuration" section in the *Kofax RPA Administrator's Guide* for more information. If you are running Management Console in Tomcat, see the "Security" section under "Tomcat Management Console" in the *Kofax RPA Administrator's Guide*.

Process Discovery Analyzer

Kofax RPA Process Discovery Analyzer is designed to process the recorded raw data and to generate refined data for the Kofax Analytics for RPA dashboard.

Specify Process Discovery Analyzer database connection settings, runtime parameters, cluster settings, and supply credentials to perform database provisioning on the Process Discovery Analyzer page. Once you finish editing, click **Save** to save the settings.

Option	Description
Schedule settings	Select how to run the Analyzer. • **Execute at specific times**: Set a comma-separated list of time values in `hh:mm` format to run Analyzer in the **Execute at specific times** field. The time can be specified as follows: 1:00, 15:30 • **Execute periodically**: Set the time interval between Analyzer runs in the **Execution interval (hours)** field.

Option	Description
Cluster settings	Specify cluster settings for the Process Discovery Analyzer. See Process Discovery Analyzer cluster for details. **Note** If you change any cluster settings, restart the master node. If you assign another computer as a master node, restart both the current master node and the newly assigned master node. For example, currently your cluster contains three nodes: A, B, and C. "A" is a master node. If you assign "B" as a master node, restart "A" and "B." • **Master address**: IP address (IPv4 only), fully qualified domain name (such as `myhost.company.com`), or a computer name of the computer hosting the master application of the cluster. Specify a domain name as a master address if DHCP is used in your network. • **Network pattern**: This is an optional parameter that helps you bind the cluster computer to a specific subnet, such as `192.168.*.*` or `10.*.*.*`. Enter the pattern if your computer runs several network interfaces with different IP addresses. The following ports must be changed only if standard ports are blocked on your network. • **Master port**: Specify a port for the master application host. • **Master WebUI port**: Specify a port to access the activity monitor web interface of the master application host. • **Worker WebUI port**: Specify a port to access the activity monitor web interface of the worker application host. • **Master memory (GB)**: Specify the maximum amount of memory for the master process. If you encounter an `outofmemory` type of error in the Analyzer log on a master node, increase the amount in this setting and restart the master node. • **Worker memory (GB)**: Specify the amount of memory Apache Spark reserves for the worker process when the process starts. The Apache Spark cluster technology used in Process Discovery implies that you use computers that are similar in processing power and memory amount as worker nodes. If worker nodes have different amount of free RAM, specify the lowest available amount of free memory on a node. If a node does not have enough memory to reserve for a worker process, it is started but it does not participate in data analysis. To check the node, open `http://<worker_ip_address>:<worker_ui_port>` in a browser and make sure that there is a running executor on the node. **Important** Do not decrease below the default settings the amount of memory on master and worker nodes.

Option	Description
Database settings	Specify database connection settings for the Process Discovery Analyzer to store refined data. • **Host**: Database server name or IP address. If you use Docker for deploying Process Discovery, this is the computer where Process Discovery runs in a Docker container. • **Port**: Specify the port to access the database. • **Schema**: Name of schema. • **Type**: Specifies the type of database. This parameter cannot be changed. • **User name**: Account name to access the database by the Analyzer. • **Password**: Account password to access the database by the Analyzer. • **Test** button: Checks the connection to the database.
Advanced database settings	Specify Analyzer run settings. **Important** Do not alter settings in this section without consulting with Kofax Technical Support. • **Connection timeout (seconds)**: Database operations timeout setting. • **Read batch size**: A number of events read from the Agent database in one transaction. • **Write batch size**: A number of events written to the Analyzer database in one transaction.
Database provisioning	Enter credentials to create the database schema and grant access to the user specified in **Database settings**. **Note** Database provisioning buttons are disabled until you save your settings on the **Database settings** tab. • **Database administrator name**: Specify the name of the administrator account to access the database. • **Database administrator password**: Specify a password to secure the administrator account. • **Create schema** button: Creates a schema in the database using the specified credentials. If the schema exists, you are prompted to overwrite the existing schema or cancel. • **Grant user access** button: Grants necessary database access rights to the user specified in **Database settings**. If the user exists, you are prompted to recreate the existing user or cancel. If you recreate the user, the previous user access rights and other user settings are lost.

Process Discovery Groups

This section contains Process Discovery Group settings. Groups specified here logically combine computers with installed Process Discovery Agents, such as `finacial_dept`, `hr_dept`, `sales_dept`, or others. When you deploy the Process Discovery Agents, specify a name of the group created here.

To create a new group, click **Process Discovery Groups** on the left menu, specify necessary parameters on the **Create new group** tab and then click **Save**. A new tab with the specified Process Discovery Group name appears in the **Process Discovery Groups** section and you are automatically navigated to the newly created group tab.

If you are editing settings in the existing group, click **Save** to save the settings after editing.

Option	Description
Group settings	Specify general group parameters. • **Group name**: Specify a new or edit the existing group name. • **Record**: Enables action recording for this group. • **Analyze**: Enables action analysis for this group.
Recording settings	**Application ignore list**: This is a comma-separated case-insensitive list of process names of the programs that Process Discovery ignores when recording and processing data. A program process name can be viewed in the Windows Task Manager. As ignored programs you can specify messengers that usually are not part of the work process. Specify this list to increase the Agent and Analyzer performance and reliability of discovered processes by removing extra "noise" from data.
Agent database settings	Specify database parameters for Agents to access and store data. • **Address**: Database server name or IP address. • **Port**: Specify the port to access the database. • **Schema**: Name of schema. • **Type**: Specifies the type of database. This parameter cannot be changed. • **User name**: Account name to access the database by the Agents in this group. • **Password**: Account password to access the database by the Agents in this group. • **Test** button: Checks the connection to the database.
Agent database provisioning	Enter credentials to create the database schema and grant access to the user specified in **Database settings**. These options are available in the existing group. **Note** Database provisioning buttons are disabled until you save your settings. • **Database administrator name**: Specify the name of the administrator account to access the database. • **Database administrator password**: Specify a password for the administrator account to secure it. • **Create schema** button: Creates the schema in the database using the specified credentials. If the schema exists, you are prompted to overwrite the existing schema or cancel. • **Grant user access** button: Grants necessary database access rights to the user specified in **Database settings**. If the user exists, you are prompted to recreate the existing user or cancel. If you recreate the user, the previous user access rights and other user settings are lost.

Option	Description
Analysis settings	Specify Analyzer run settings for this group. See the "Analysis settings explained" section below for more information. • **Min count of process instances**: Set the minimum number of similar sequences of user actions to form a process. • **Min action count in process instance**: Set the minimum number of user actions in process instances to discover. • **Max count of sequential noise actions in process instance**: Set the maximum number of consecutive user actions unrelated to a process instance. • **Max inactivity interval in process instance (minutes)**: Set a maximum user inactivity period in discovered process instances. • **Max application count in process instance**: Set a maximum number of different applications in a process instance. • **Blur screenshots**: If this option is selected, Analyzer makes the text on screen shots unrecognizable for the Insight dashboard views. The screen shots in the Agent database remain intact. We strongly recommend selecting this option if the screen shots can contain sensitive data, such as personal information, passwords, credit cards numbers, and so on.
Advanced recording settings	Specify advanced recording settings for this group. • **Non-UIA applications**: Enter a comma-separated list of process names of the programs for which Process Discovery uses ISA (Intelligent Screen Automation) instead of UI Automation API. By default, the field contains names of some communication programs, Java programs, all browsers except Chrome, and others that show more reliable results when using ISA. Because Kofax RPA implements native UI automation support for the Chrome browser, we recommend using Chrome as the default browser in your environment. • **Agent SQLite DB max size (MB)**: Set the maximum size of the data stored locally on the computer where an Agent is running. The Agent stores collected information locally if connection to the database cannot be established. If the stored data exceeds the limit, the Agent starts overwriting the oldest data. The default setting provides storage for two to three days of data collection. Increase the size limit if you often experience long network outage. • **Min UIA tree immutability duration (ms)**: This is a time the Agent waits before recording a program tree after the tree stopped changing. The lower the timeout, the faster the Agent records the tree and the better results you can get. If programs on the user computers become slow in response, increase this number. Do not change the default setting without thorough consideration.

Option	Description
Advanced analysis settings	Specify advanced Analysis output settings for this group. Do not alter the default values of the following settings without thorough consideration. • **Max compute score distance** (0-100): Sets the proximity score for comparing user actions based on the number of actions in groups. • **Min action block score** (0-100): Provides a resemblance score threshold for combining adjacent action blocks. Decreasing the number makes Analyzer more tolerant and combine more blocks. Increasing the number makes Analyzer more precise in combining action blocks. • **Min strong pair score** (0-100): Sets user action resemblance score to combine actions into pairs while detecting processes. If you decrease the value, you can get more discovered processes with more false detections and vice versa. • **Max distance between process instances** (0-100): Sets the minimum resemblance score of the detected process instances to combine them to clusters. The lesser the specified number, the more processes are considered similar and combined, but you can get more unalike processes combined in clusters. If you increase the value, more clusters are created with less process instances in them. • **Actions in blocks for comparison**: Sets the maximum number of user actions in blocks used for comparing and combining those blocks. If you decrease the value, Analyzer becomes more tolerant to "noise" actions and detects more processes, but you get shorter processes and might miss some longer ones. If you increase the number, Analyzer may start combining more blocks with unalike actions. • **Max ISA X position difference (px)**: Sets the maximum element offset in pixels on the x-axis until marking it as a different element when detecting similar actions. This setting is applied to processes detected using ISA technology. If you increase this value, more actions are considered similar, which may lead to more false process detections and vice versa. • **Max ISA Y position difference (px)**: Sets the maximum element offset in pixels on the y-axis until marking it as a different element when detecting similar events. This setting is applied to processes detected using ISA technology. If you increase this value, more actions are considered similar, which may lead to more false process detections and vice versa.

Analysis settings explained

See the Process Discovery glossary for terms used in this section.

- **Min count of process instances**: This number specifies how many times a sequence of similar actions occur in the collected data to mark that sequence as a process. The number must be consistent with the amount of data you collect (number of people, duration of the data collection process, and so on). If the number of users is large and you spent considerable time collecting the data, you can increase the quality of the analysis by specifying a larger number to cut off short repeated actions that are not beneficial to automate, to shrink the size of the report, and to decrease the analysis time. Before you change the number, it is good to have a general idea of the work in the selected department. For data collected during two weeks or more in a department with 20 people or more, you can start with a number 15 and then increase or decrease the number depending on the discovered processes in the report. We recommend analyzing the same data with different values in this and other properties to refine the report.

- **Min action count in process instance**: This property specifies a number of unique user actions (such as a mouse click, entering text in a text field, copying text, and so on) that the Analyzer must discover and mark as a process. If you want to discover processes that have different number of unique actions, such as five and fifteen, you must specify the lower number (5) in this field.
- **Max count of sequential noise actions in process instance**: This is an additional setting that helps the Analyzer to consider some user actions unrelated to the process instance while discovering this instance and cutting a sequence of actions into process instances. This setting allows to discover process instances even if during a usual course of action a person is distracted by erroneous clicks or mistypes, important instant messages, other tasks withing the same application, and the like. Note that a process instance cannot contain more than the specified number of consecutive noise actions in it. For example, if while performing a usual sequence of actions a person started doing something else and exceeds the number of noise actions specified in this field, the Analyzer considers this process instance finished. Increasing the number in this setting may lead to discovering larger process instances that stuck together, while decreasing the number leads to fragmentation of the process instances.
- **Max inactivity interval in process instance (minutes)**: This setting specifies the maximum duration of a person's inactivity at the computer, that is, when a person is away from computer. A process cannot be discovered if user inactivity time exceeds the specified value. The value you specify must be a little over the inactivity periods in the processes you want to discover.
- **Max application count in process instance**: This property must contain the maximum number of applications involved in a process instance. Before changing the default value, consider the number of applications your employees use when performing their usual tasks. The large number of applications may lead to unreliable data.

KTA configurations

Use this section to configure the credentials that the Management Console will use when running robots referencing the Kofax TotalAgility installation.

To add a new Kofax TotalAgility installation, specify the following properties on the **New KTA configuration** tab.

Property	Description
KTA Installation name	Specify a new Kofax TotalAgility installation name.
URL	Specify a URL for the computer running Kofax TotalAgility.
User name	Type a user name to connect to the selected installation. This must be the user name that you use to log in to Kofax TotalAgility.
Password	Type a password to connect to the selected installation. This must be the password that you use to log in to Kofax TotalAgility.

Click **Test** to test the connection. After the test connection has been executed, Management Console shows a message with an execution state (success or failure with a description).

Click **Save** for the changes to take effect.

Email accounts

This section lists email accounts set up for the email trigger functionality. Use this section to add new accounts and delete accounts that you no longer need.

At the top of the "Email accounts" section, in the Projects drop-down list, select the project with email accounts to display. You can change the way the information for each email account is presented as follows:

- Select the table columns to display for an email account using the ▥ menu icon on the right.
- Refresh the displayed information by clicking the ↻ refresh icon on the right.

By default, the following information is displayed for each email trigger.

Column	Description
Name	Name of the email trigger.
Description	Description of the account.
Project name	Project linked to the email account.

Add Email Account

1. To add a new email account, click the plus sign in the upper left corner.
   The "Create new email account" dialog box.
2. Select an account type and specify the following properties.

   Google Gmail and Microsoft 365 account properties

Property	Description
Name	Specify a name for the account.
Description	Specify the account description.
Assign to project	Select a Kofax RPA project for the email account.
User name	Specify the account login name.
Password	Enter the account login password.[5]

   IMAP account

Property	Description
Name	Specify a name for the account.
Description	Specify the account description.
Assign to project	Select a project for the email account.

[5] Depending on the email account provider, you might need to create an App password and specify it in this field to make sure the account is not blocked by the server. See the email provider documentation for details.

Property	Description
SSL/TLS encryption	Select this option if the email account provider requires to use the SSL/TLS protocol.
Host	Specify the IMAP host name.
Port	Enter the port number for the selected IMAP host.
Protocol	Select the protocol to use either IMAPS (IMAP over SSL) or IMAP.
User name	Specify the account login name (usually the email address).
Password	Enter the password for the specified email address.

3. Click **Test** to test the connection. After the test connection has been executed, Management Console shows a message with an execution state (success or failure with a description).
4. Click **Save** for the changes to take effect.

To delete an account from the list, select it and click the 🗑 bin icon in the upper left corner. You are prompted to confirm the deletion.

CyberArk configuration

Use this section to configure credentials for the Management Console to use when accessing the CyberArk password manager.

To upload a CyberArk application, see CyberArk applications. To use CyberArk in Kofax RPA, see CyberArk.

Use the following properties to configure the credentials.

Property	Description
CyberArk URL	Specify the URL for the CyberArk Central Credential Provider host.
CyberArk port	Specify the port number for the CyberArk Central Credentials Provider host.
IIS application name	Specify the CyberArk Central Credentials Provider application name specified in the Internet Information Services (IIS).
Server certificate	Upload the TLS server certificate of the CyberArk Central Credential Provider. It is only required if you use a self-signed certificate for the CyberArk Central Credential Provider web service. • To upload a new certificate, click the **Change certificate** ⭳ button. • To remove a certificate, click the **Remove certificate** × button.

Click **Test** to test the connection. After the test connection is executed, a message is shown containing an execution state.

CyberArk applications

Use this section to upload CyberArk application.

Add New CyberArk application

1. Click the plus sign.

   The "CyberArk application" dialog box appears.
2. Fill in the following fields to configure credentials:
   - Application ID: Enter the ID specified on the CyberArk Applications list.
   - File to upload: Click the 📎 paper clip sign to select a keystore file to upload. This is a keystore file with a certificate-private key pair required to authorize the application in CyberArk.
   - Store password: Enter a keystore password.

     Important Always use strong passwords to ensure maximum security of your data.
   - Private key password: Enter a private key password to the keystore, if present.
3. Click **Save**.

JMX

Management Console offers a small amount of information via the JMX protocol. The information exposed through JMX is experimental, and should not be considered a public API.

Management Console exposes the following MBeans.

Name	Description
DistributionController	Information on executing schedules, and cluster configuration.
FileBackedDataExporter	Information on exports created from the Data View.
RobotLibraryGenerator	Allows you to control the internal caching of Robot Libraries within Management Console.
TaskQueuePerformanceTracker	Allows you to profile the distributed task queue between two clustered Management Console instances.

The MBeans are accessible via JConsole and Java VisualVM (+MBean plugin). Both are included in JDK 1.6+ or any other JMX client.

OAuth

OAuth is an open standard for authorization used with many popular APIs such as Twitter and Facebook. It provides a means for applications and robots in case of Kofax RPA to access data or perform actions on behalf of the user without having direct access to the user login credentials. For example, a robot can use

Twitter API to extract the most recent mentions of a user, such as @Kofax, using access tokens provided by that user but without having explicit access to @Kofax Twitter password.

Management Console is used to generate access tokens that are stored securely in a keystore. The access tokens can be passed on as input to robots in a schedule. As always, the robots that perform the actual API calls and handle the returned data are created in Design Studio.

Supported service providers

In the OAuth terminology, a service provider is a provider of the web service. Kofax RPA supports the following service providers. Documentation on the APIs of each service provider can be found on their respective websites.

- Dropbox

  https://www.dropbox.com/developers
- Facebook

  https://developers.facebook.com
- Google

  https://console.developers.google.com/apis/library
- LinkedIn

  https://developer.linkedin.com
- Microsoft Azure AD 2.0

  https://docs.microsoft.com
- Salesforce

  https://developer.salesforce.com
- Twitter

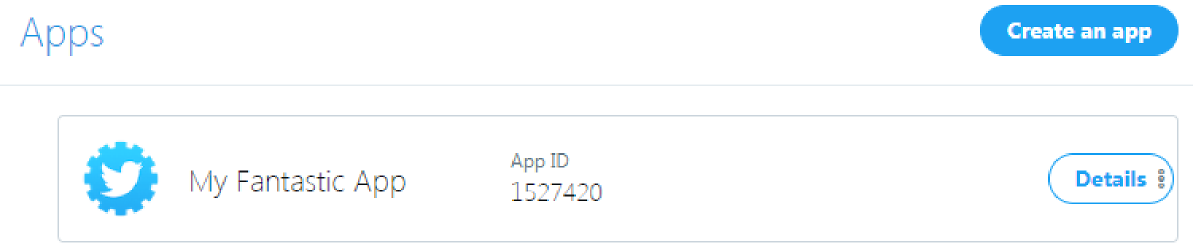
  https://developer.twitter.com

Add applications

To access the API of a service provider, you must create an application with that service provider. Creating an application provides you with a consumer key (also known as an API key or application key) and a consumer secret (also known as an API secret or application secret).

To create an application, log in to the developer community of the service provider, select **Create New Application**, or similar option, and fill out the required information. See Supported service providers for links to service providers documentation for developers. In this topic, we take a look at how this is done with Twitter.

1. First, log in to https://developer.twitter.com/en/apps/, creating a new account if necessary, and click **Create an app** in the top right corner.

2. Fill out the required information, such as the name and description of the application, and read through Twitter terms of service before accepting.

   One of the fields in the form is a **Callback URL**. This is the URL that Twitter redirects a browser to after the user has accepted to let your application interact with the Twitter account on the user's behalf. This field must be set to the path OAuthCallback under the folder in which the Management Console is deployed. For example, if running with an embedded Management Console, it runs at `http://localhost:50080/`. In this case, the callback URL would be specified to `http://localhost:50080/OAuthCallback`. However, note that some service providers do not allow a callback URL containing `localhost`. Twitter is one of those providers, so use `http://127.0.0.1:50080/OAuthCallback` instead.

   Callback URL: http://127.0.0.1:50080/OAuthCallback

   Alternatively (and this is required by some service providers), you need to specify the hostname or non-loopback IP address of the computer on which you are running Management Console. As this page is loaded by the browser of the authenticating user, this cannot be a public hostname or IP address.

   > **Note** All service providers require an HTTPS connection if the IP address is other than 127.0.0.1. In this case, the Management Console must also be configured to use SSL. For more information, see the Tomcat documentation on using SSL.

   After creating the application, a summary of the application appears. You must copy some of these values into Management Console.

3. Open Management Console in a browser. Use the same IP address or hostname that was entered as the callback URL.

   In the example below, we point our browser to http://127.0.0.1:50080/.

4. Navigate to **Repository** > **OAuth** and click the plus sign in the upper left corner.

   The "New application" dialog box appears.

5. Select a project and specify the following information.
   - Name: Specify a name for the application. It does not need to be the same name as was used when you created the application at the service provider.
   - Service provider: Select the service provider. To follow the example above, select Twitter.
   - Consumer key and Consumer secret: Copy these values from the summary page of the application presented by the service provider. For the Twitter example procedure, see Step 2.
   - Callback URL: Specify the callback URL. For the Twitter example procedure, enter `http://127.0.0.1:50080/OAuthCallback`.
   - Scope: Some service providers additionally require that you specify a scope, such as what parts of the API that a user authorizes the application to access. For example, when accessing Google, the scope `https://www.google.com/analytics/feeds/` must be specified if the application should be allowed to access the Google Analytics Data API. Twitter does not use the scope field, so leave this field blank if you are following the example procedure.
   - Commit message: You may add a description for the commit.
6. Click **OK**

   You have now set up an OAuth application in the Management Console.

To edit the application later, click ⋮ > **Edit**. The consumer secret will be displayed as "(encrypted)" for security reasons.

To delete an application, select it in the table and click the click the 🗑 bin icon in the upper left corner. You are prompted to confirm the deletion.

Add users

This topic shows how to add a user for an OAuth application configured in Add applications.

1. Click the plus sign in the upper left corner.

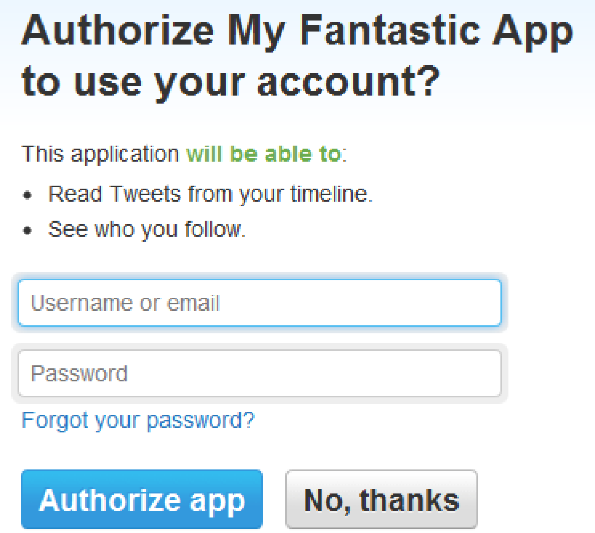
   A dialog box with several tabs appear.
2. On the **Select application and user name** tab, select an application and specify a user name.

   This user name does not need to match the user name used by the service provider. It is only used inside Management Console.
3. Click **Next**.

4. On the **Authorize** tab, click the Authorization Link.

Authorization link:

This opens the website of the service provider. At Twitter, it looks as follows:



Authorize My Fantastic App to use your account?

This application will be able to:

- Read Tweets from your timeline.
- See who you follow.

Username or email

Password

Forgot your password?

5. Enter the user name and password and click **Authorize app**.

   The service provider now forwards you to the callback URL. If the authorization was successful, the Proceed with OAuth Authorization page appears.

6. Close the browser tab and return to the Management Console. In the wizard, click **Next**.

   On the **Retrieve access tokens** tab, you will see the access tokens that can be used for accessing the service provider on behalf of the user. They have been securely stored in the Management Console keystore and can now be used as input to schedules.

   Note You will need sample access tokens as test input for the robot that we will build in a later step. Copy the values into a text editor such as Notepad. For security reasons, you will not be able to retrieve them from the keystore in unencrypted form after clicking Finish.

   At Twitter, we get both an access token and an access token secret. Service providers that use OAuth 2.0 do not use an access token secret, so they will only return an access token. Some service providers will additionally return a refresh token. This is used when the access tokens returned by the service provider are only short-lived. Robots can then use the refresh token to obtain new access tokens without a user having to re-authorize through the Management Console. To create robots against the API of a service provider, you must copy all of the tokens displayed at the final step of the wizard.

7. Click **Finish**.

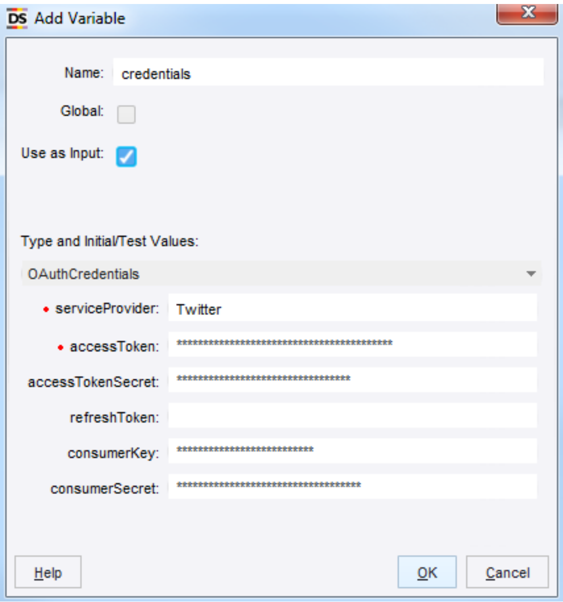
   The user entry appears in the OAuth section.

Write robots

The following procedure shows how to write a robot that accesses a REST API that uses OAuth as its authentication mechanism. As an example, we use the Twitter REST API to obtain the most recent statuses by the authenticating user and the followed users.

1. Start Design Studio and create a new Web Automation robot.

   Do not enter a URL in the wizard. You will not be able to access the REST API before authentication.
2. To the robot, add a new variable of complex type OAuthCredentials and select **Use as Input**.
3. In the **serviceProvider** field, type Twitter.
4. Enter the access token and access token secret that were obtained when you went through the user authorization process in the Management Console. Also enter the consumer key and consumer secret for the Twitter application.



Name: credentials

Global: □

Use as Input: ☑

Type and Initial/Test Values:

OAuthCredentials

- serviceProvider: Twitter
- accessToken: **

accessTokenSecret: ********************************

refreshToken:

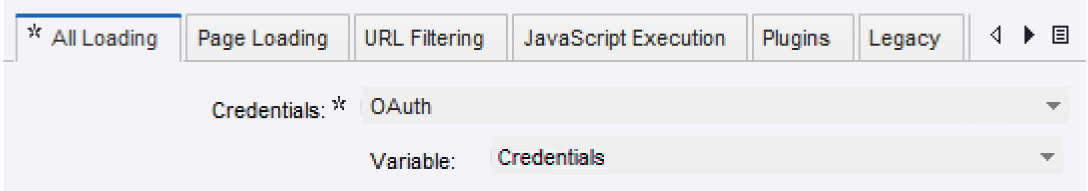
consumerKey: *************************

consumerSecret: **********************************

5. Click **OK**.
6. Click **Configure Robot** .

   The robot configuration window appears.
7. On the **Basic** tab, click **Configure**.
8. On the **All Loading** tab, locate **Credentials** and switch it from **Standard** user name/password authentication to **OAuth**.

9. Select the input variable you just added.



Credentials: * OAuth

Variable: Credentials

You should now see the XML that has been returned, containing the most recent statuses in the user timeline.

10. Click **OK** in both dialogs.

    The robot is now configured to use OAuth and use the specified credentials when running in Design Studio. You can now start accessing the Twitter API. For example, to see a collection of the most recent Tweets posted by the user, you can access the URL `https://api.twitter.com/1.1/statuses/user_timeline.json`.

Schedule robots with credentials

The following procedure shows how to schedule a robot created in Writing Robots.

1. Upload the Web Automation robot created in Writing Robots to Management Console.
2. Open the Management Console in a browser and create a new schedule.
3. Select the **Jobs in schedule** tab.

   Select **A single robot**, add the Web Automation robot to the schedule, and then click **Next** until you reach the **Configure input** step.
4. Select the OAuth user whose credentials you want to use when running the robot in this schedule.

   This action auto-populates the service provider, access tokens, consumer key, and consumer secret fields when the robot is run on RoboServer.
5. Click **Next** and click **Save** to save the schedule.

   The robot is now ready to run.

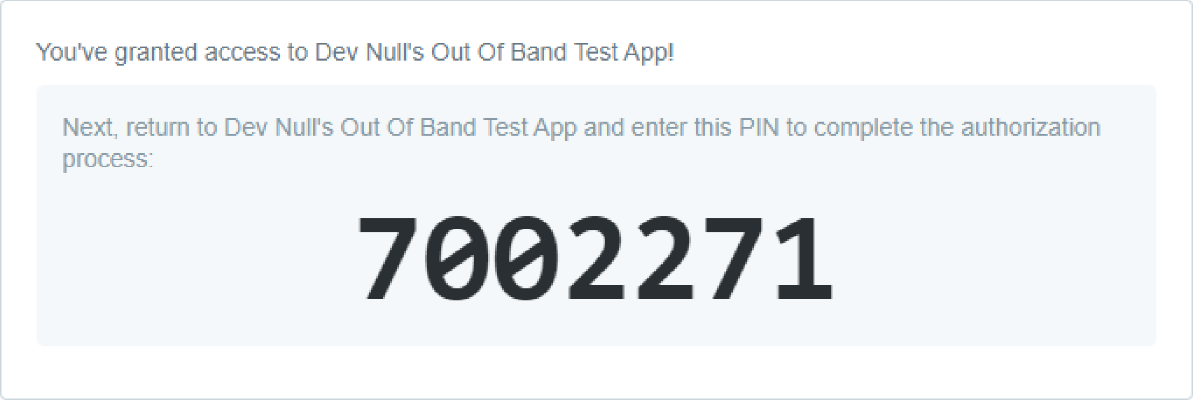
Out of band applications

Some service providers also offer a mechanism to authorize OAuth applications without using a callback. This is known as out-of-band. Management Console also supports out-of-band authorization. The application created at your service provider must be registered as an out-of-band application. For example, at Twitter this is done simply by leaving the callback URL field blank.

1. Leave the callback URL field blank when creating a Twitter application.

2. When registering the application in Management Console, leave the callback URL field blank here.

   When adding users to the application, clicking the authorization link does not redirect back to the Management Console. Instead, the service provider presents a verifier or PIN.

   You've granted access to Dev Null's Out Of Band Test App!

   Next, return to Dev Null's Out Of Band Test App and enter this PIN to complete the authorization process:

   7002271

3. In the **Verifier** field, enter the PIN provided by your service provider.
4. Click **Next**.

   When the user is successfully authorized, the **Retrieve access tokens** tab is opened with the **Access token** and **Access token secret** fields filled out. Click **Finish**.

Filtering

The **Filter** field in most of the sections of the Management Console helps you filter a list of items to view by their names and path. Note that the **Robots** and **Trigger mappings** sections contain filtering text fields that do not use wildcards. For example, to filter the list of robots by their tags, type a tag name or a part of the name in the **Tag** text box. In some sections, such as **Admin** > **RoboServers**, to filter the list, first select the item to view by clicking **Filter** and then type its name.

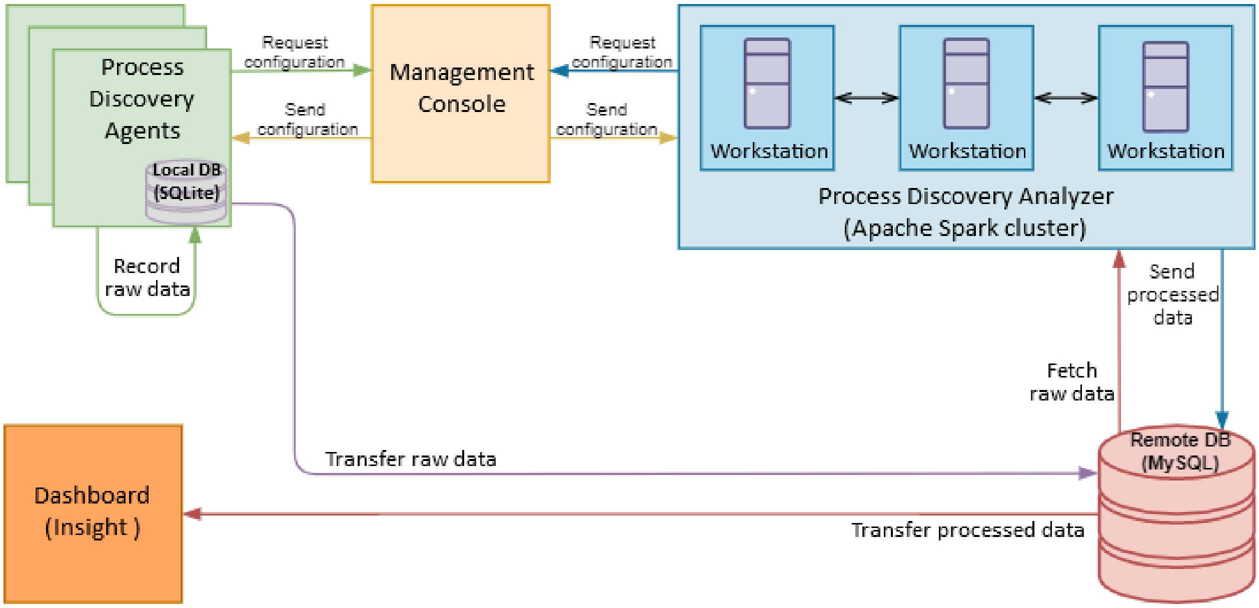
Note the following rules when filtering the list using the **Filter** field:

- ? matches a single character, * matches zero or more characters
- When you type a search string without a forward slash and without any wildcard characters, the search is performed as a substring matching the name of the items. For example, the search string "foo" would match the robots `foo` and `foobar` regardless of the folder they are in.
- When you type a search string without a forward slash but with one or more wildcard characters, the search is performed as pattern matching the full name of the item. For example, "foo*" would match the robots `foo` and `foobar` regardless of the folders they are in, but would not match `xxxfoo`. The search string "foo?ar" would match the robot `foobar`.
- When you type a search string containing a forward slash, the search is performed to match the full path of the items, that is folder plus name. For example, the search string "sub1/foo" would match `sub1/foo`, but not `sub1/foobar`. The search string "sub1/foo*" would match both `sub1/foo` and `sub1/foobar`.

Chapter 7

Process Discovery

Kofax RPA Process Discovery helps you gather the data necessary to monitor user activity and process bottlenecks, and to make informed decisions about using Kofax RPA in your production environment.



Process Discovery Components

Note Process Discovery relies on Windows UI Automation API. Do not run any UI Automation API clients on the same computer simultaneously with Process Discovery Agent.

The Process Discovery views in the Kofax Analytics for RPA dashboard provide reports with information based on data collected by the Process Discovery Agents. The view includes a variety of visual and analytical representations of data using charts and tables.

Your product includes the files you need to successfully install the Kofax RPA Process Discovery components. See the following table.

Components	Required Installation File
Linux • .deb and .rpm packages • Dockerfile for installing and running Analyzer **Windows** Process Discovery Analyzer .msi installer	`Kofax_RPA_ProcessDiscoveryAnalyzer_11.1.0.0_<buildnumber>.zip` Process Discovery Analyzer is designed to process the recorded raw data and to generate refined data for the Kofax Analytics for RPA dashboard views.
Process Discovery Agent	`Kofax_RPA_ProcessDiscoveryAgent_11.1.0.0_<buildnumber>_x32.msi` Process Discovery Agents are installed on computers to gather user activity information in specified applications and store the data in a database.

Kofax Analytics for RPA	• Kofax Insight installation file, such as:
	`KofaxInsightSetup_6.2.1.0.0.<build>_x64.msi`
	• Kofax Analytics for RPA project `zip` file, such as:
	`Kofax_RPA_KAFRPA_11.1.0.0_.zip`

To use Process Discovery, perform the following installation and configuration steps.

Important Before deploying Process Discovery, make sure Java Runtime Environment is installed on the computer running Process Discovery Analyzer and `JAVA_HOME` variable that points to the JRE installation folder is added to system variables.

1. Depending on your environment, perform one of following:
   - Deploy Analyzer using Docker on Linux
   - Deploy Process Discovery Analyzer on Windows
   - Deploy Process Discovery Analyzer on Linux
2. Process Discovery Agents.
3. Analytics.

Important Kofax RPA Process Discovery is designed to use a MySQL database. For supported versions of MySQL, see the *Kofax RPA Technical Specifications* document on the Kofax support portal. We recommend using a dedicated server with fast disk drives, such as SSD for your MySQL database and run no more than 100 agents simultaneously for one Process Discovery group.

MySQL database name length limit specified in Management Console is 104 characters.

Process Discovery glossary

- **Event**: An action recorded by the Agent on a user desktop. Examples are left clicks on a PDF in Acrobat, choosing an option from a context menu in Excel, pressing Enter key, and so on.
- **Sequence**: A set of events performed by a single user, ordered in time.
- **Task**: A sequence that achieves a particular goal. The user might have indicated the start and end of the sequence that makes up the task.

- **Step**: A discrete action executed to accomplish some task. Steps are detected by the Analyzer based on observed events. Those events are associated with that Step.
- **Process**: A set of steps, ordered in a directed graph, that accomplishes some task. Processes are detected by the Analyzer based on observed sequences. A process may or may not coincide with a task.
- **Process instance**: An observed sequence that fits a process.

Process Discovery Agents

Process Discovery Agents are installed on computers to gather user activity information in specified applications and store the data in a database.

> **Note** Include the Agent program into your antivirus software exception list for the uninterrupted data gathering.

In Management Console create at least one group to logically combine computers and assign Agents to, such as `financial_dept`, `hr_dept`, `sales_dept`, or others. When you deploy Process Discovery Agents, specify a name of the group created in the Management Console.

To create a group before installing an Agent, perform the following steps.

1. In the Management Console, expand **Settings** in the left pane and click **Process Discovery Groups**.
2. On the **Create a new group** tab type a group name, specify other settings, and click **Save**. For details, see Process Discovery Groups.

   This action creates a new group and opens a tab with a newly created group settings.
3. Perform database provisioning by clicking **Create schema** and **Grant user access** in the **Agent database provisioning** section.

To manually install an Agent, run `Kofax_RPA_ProcessDiscoveryAgent_11.1.0.0_<buildnumber>_x32.msi` and follow the wizard steps.

> **Note** Any previous version of the Process Discovery Agent is removed when you install a new version of the Agent.

The first time you run the Agent after installation, the Process Discovery Agent configuration window is opened. Fill in parameters, such as Management Console URL, Process Discovery group name, and credentials to log in to the Management Console if necessary. Click **Start recording**.

You can also perform a silent installation of Agents for automating the installation process using a script file.

Configure Process Discovery Agents

To edit the Agent settings, right-click the Process Discovery Agent icon in the notification area and select **Configure**. To open the help topic, right-click the Process Discovery Agent icon in the notification area and click **Help**.

Troubleshooting

The following procedures help you resolve some common problems when Agents run.

If computers in general or several programs on the user computers become slow in response when the Agents are running, perform the following.

1. Open the Management Console and click **Process Discovery Groups** under **Settings** in the left pane.
2. Open the tab with the group the computers belong to.
3. Locate the **Advanced recording settings** and increase the timeout in the **Min UIA tree immutability duration (ms)**. Use 50 ms increments until the computers become responsive.

If a certain program on a user computer becomes slow in response when the Agent is running, you can do one of the following.

- If the program is not a part of your business processes and the Agent can ignore it, in Management Console open the Process Discovery group the computers belong to and type the program executable name in the **Application ignore list** under **Recording settings**.
- If the program is a part of your business processes and the Agent must record it, either include the program name to the **Non-UIA applications** list under **Advanced recording settings** on the respective group tab, or perform the same steps as for computers that are slow in response in general as described above until the program becomes responsive. Note that these changes can change the process analysis results.

If your business process involves using Microsoft Excel, use the latest available version to get better results in discovering processes.

Silent installation of Process Discovery Agents

You can perform a silent installation of agents for automating the installation process using a script file. To perform a silent installation of Kofax RPA Process Discovery Agent to the default location, run the following command with administrative rights:

```
msiexec /qn /I Kofax_RPA_ProcessDiscoveryAgent_11.1.0.0_<buildnumber>_x32.msi
MCURL="mc_url" MCUSER="username" MCPW="password" GROUP="group"
```

Where “`mc_url`”, “`username`”, and “`password`” are a URL, a user name, and a password to connect to the Management Console, and “`group`” is the group name Created in the Management Console in the **Process Discovery Groups** section under **Admin** > **Settings** .

To specify a location to install the agent, run the following command:

```
msiexec /qn /i Kofax_RPA_ProcessDiscoveryAgent_11.1.0.0_<buildnumber>_x32.msi
MCURL="mc_url" MCUSER="username" MCPW="password" GROUP="group"
INSTALLDIR="dir"
```

Where `"dir"` is the path where you want to install the agent.

To log the installation process, specify a log file:

```
msiexec /qn /i
Kofax_RPA_ProcessDiscoveryAgent_11.1.0.0_<buildnumber>_x32.msi /l
RPAmsilog.txt
```

To start the agent after installation, specify the following command:

```
"start /d <installdir> KofaxRPAProcessDiscoveryAgent.exe"
```

Where `<installdir>` is the directory where the program is installed. If the `INSTALLDIR` option was specified during the installation, use its value for `<installdir>`.

Note To edit the agent settings, right-click the Process Discovery Agent icon in the notification area and select **Configure**. To open the help topic, right-click the Process Discovery Agent icon in the notification area and click **Help**

Configure the Process Discovery Agent

The Process Discovery Agent is installed on a computer to gather user activity information in specified applications and store the data in a database.

Note Some assistive technology applications, such as screen readers or anti virus software may detect the Process Discovery Agent and either produce warnings or prevent the Agent from running. To allow the Agent to run, approve the Process Discovery Agent as a safe application or select the "Do not show this dialog box again" option when a warning is shown.

Configuration window

To edit the connection settings, right-click the Process Discovery Agent icon in the notification area and select **Configure**. The opened **Kofax RPA Process Discovery Agent** window contains the Management Console connection settings and shows the agent's status messages. The window also contains the following buttons:

- **Quit**: Stops the Agent and exits the program completely.
- **Help**: Opens the "Configure Process Discovery Agent" topic in help.
- **About**: Opens a window with the agent's version and log file information.
- **Start recording** and **Stop recording**: Starts or stops recording user actions.

Management Console connection settings
Contains connection settings to connect to the Management Console.

- **URL**: URL and port of the Management Console to connect to as in the following examples:
  - Embedded Management Console: `http://localhost:50080`
  - Management Console on Tomcat: `http://192.168.0.101:8080/ManagementConsole`
- **Group**: Process Discovery group name specified in the Management Console.
- **User**: Login username.
- **Password**: Login password.

Status

The following table lists Agent recording and connection statuses.

Component	Status
Agent	**Not started**: The Agent is not running. **Recording is being started**: The Agent is starting and all required connections are currently being established. **Recording to local storage**: The Agent is running, but connection to either the Management Console or the database cannot be established. The Agent is storing collected information on the computer where it is running. The Agent periodically tries to reconnect. **Recording to remote storage**: The Agent is running, all the connections are established successfully, and collected data is stored in the Agent database. **Recording disabled**: The Agent is running, but does not record any user actions. Recording is disabled in the **Recording mode** setting in the Management Console. **Failed**: The Agent could not record user actions during the last try.
Management Console	**Not connected**: The Agent is not connected to the Management Console. Possible reasons are: • Recording is not started. • Management Console URL is not specified. • Process Discovery group is not specified. **Connecting**: The connection to the Management Console is currently being established. **Connected**: The Agent is connected to the Management Console. **Connection failed**: The Agent could not connect to the Management Console during the last try. Possible reasons are: • Unable to connect to the specified host. Make sure the Management Console server is up and running. • The specified port is not open or not reachable. Make sure the Management Console is running on the specified port. • Invalid username or password. Specify correct Management Console username and password. • The Management Console does not support the resource path. Make sure it is correctly specified. The Agent periodically tries to reconnect. **Incorrect settings**: In case this message is displayed, make sure the Process Discovery Group settings in the Management Console are correct. The Agent ties to connect and apply the settings from the Management Console periodically. **Management Console version does not match the Agent version**: The Agent cannot work with the current Management Console. Management Console and Agent versions must match. The Agent periodically tries to reconnect.

Component	Status
Database server	**Not connected**: The Agent is not connected to the database, because the connection to the Management Console is not established. **Connecting**: The connection to the database is currently being established. **Connected**: The Agent is connected to the database. **Connection failed**: The Agent could not connect to the database during the last try. Possible reasons are: • The database address is specified incorrectly in the Process Discovery Groups settings in the Management Console. • The database has undefined schema. Recreate it or change settings in the Management Console. • The database schema is for the previous version Agent. Recreate the schema or change settings in the Management Console. The Agent periodically tries to reconnect to the database.

Start and stop recording

You can start and stop recording by clicking **Start recording** and **Stop recording**, respectively.

If the Management Console settings are configured, click **Start recording** to start the Agent. First, the Agent always tries to connect to the Management Console to get the group settings. Next, the Agent connects to the database. If the connection is established successfully, the Agent starts recording and after several seconds the window is minimized to the system tray. Otherwise, the Agent performs the following:

- If the Management Console settings are not configured, the Agent starts recording and stores all collected data locally.
- If the Management Console settings are defined, but the Agent could not connect to the Management Console, the Agent starts recording and stores all collected data locally trying to reconnect periodically until the connection is established.
- If the Agent cannot connect to the database, it stores all collected data locally trying to reconnect periodically. When a connection to the database is established, the collected data is moved to the database.

If the connection to the Management Console or remote database is lost during the recording process, the Agent tries to reconnect until the connection is established.

Quit the Agent

To stop the Agent and exit the program completely, click **Quit**.

Note Clicking the **Close** button in the configuration window minimizes it to the system tray.

About window

This window lists the Agent's version and log file information.

To open the **About Kofax RPA Process Discovery Agent** window, either right-click the Process Discovery Agent icon in the notification area and select **About**, or click **About** in the configuration window.

Agent version
The Agent version is displayed next to the **Version** label in the **About Kofax RPA Process Discovery Agent** window.

Open log

Opens the Agent's log.

Open crash dump directory
Opens the directory with the memory information file on disk. This dump file can help developers to debug the cause of a crash.

Open documentation
Opens the Agent's documentation in the Kofax RPA help.

Start Task

The **Start task** option on the Agent's shortcut menu denotes the beginning of a task the employee performs, such as filling a form, answering a customer's request, preparing a report, and so on. When the task is finished, the user must click **Stop task**. The task performance can be displayed on the Process Discovery view of the Kofax Analytics for RPA dashboard. Starting and stopping tasks helps to better analyze the actions performed while the work is done. When you use this option, the discovery type of the process is marked **With assistance** in the view.

Cancel Task

If you want to cancel started task, click the **Cancel task** option on the Agent's shortcut menu.

Open help

To open the Agent's documentation, either right-click the Process Discovery Agent icon in the notification area and select **Help**, click **Help** in the Agent's configuration window, or click **Open documentation** in the **About Kofax RPA Process Discovery Agent** window.

Process Discovery Analyzer

Process Discovery Analyzer is designed to process the recorded raw data and to generate refined data for the Kofax Analytics for RPA dashboard views.

The Analyzer is a command line application that outputs its status to the console window and the Analyzer log file.

> **Note** Include the Analyzer program into your antivirus software exception list for the uninterrupted analysis.

Getting settings from the Management Console

During the startup, the Process Discovery Analyzer connects to the Management Console to receive the necessary settings and information about the Process Discovery Groups to start the analysis.

> **Note** Analyzer and Management Console version must be the same. If the version does not match, the Analyzer reports an error.

Set up Management Console URL and authentication settings to connect to the Management Console. The Process Discovery Analyzer options section explains how to set the options depending on the way the Analyzer is deployed.

Analyzing data

Process Discovery Analyzer processes all recorded data collected by the agents within the Process Discovery groups. The analysis is performed for every Process Discovery group separately. Analysis settings are specified in the Process Discovery Groups menu in Management Console.

After the connection to the Management Console is established, the Analyzer checks the connection to the database. If the database connection is established successfully, the Analyzer starts the analysis.

> **Note** The Analyzer does not perform the analysis if it cannot connect to the Management Console or the Analyzer database.

A group is omitted from the analysis if **Analyze** option is not selected for a group in Management Console or if the Analyzer cannot connect to a Process Discovery group database.

Once data analysis for all groups is finished, Analyzer enters the standby mode and waits for the next run according to the schedule specified in the Management Console settings.

> **Important** We strongly recommend starting Process Discovery Analyzer at the same time Process Discovery Agents start to perform initial preprocessing of data. Analysis is a time consuming operation that may take hours and even days depending on the amount of data. To decrease the data processing time, you can increase the number of nodes in the Analyzer cluster. See Process Discovery Analyzer cluster for details.

Log files

The location of the Analyzer and cluster log files depends on the operating system and running mode. The following log files are created:

- `analyzer.log`: General Process Discovery Analyzer log file.
- `spark_worker.log`: Analyzer cluster worker log file.
- `spark_master.log`: Analyzer cluster master log file (created on the Master node only).

Windows application

If Analyzer is installed as a Windows application, the log files are located in the Analyzer Application Data folder. For example:

```
C:\Users\UserName\AppData\Local\KofaxRPAProcessDiscoveryAnalyzer\<version>
\Logs
```

Windows service

If Analyzer is installed as a Windows service, the log files are located in the Program Data folder. For example:

```
C:\ProgramData\KofaxRPAProcessDiscoveryAnalyzer\<version>\Logs
```

Linux application

If Analyzer is installed on Linux, the log files are located as follows:

```
/home/KofaxRPAProcessDiscoveryAnalyzer/<version>/Logs
```

Process Discovery Analyzer options

Process Discovery Analyzer deployed without Docker

To see the list of Analyzer options with descriptions, in the console window run either `KofaxRPAProcessDiscoveryAnalyzer.exe` for Windows platform or `KofaxRPAProcessDiscoveryAnalyzer` for Linux platform with `-h` or `--help` parameter, such as:

```
KofaxRPAProcessDiscoveryAnalyzer.exe --help
```

Process Discovery Analyzer options

Option	Description
`-h, --help`	Shows the help message and exits.
`--version`	Shows the version of the Analyzer.
`--mc-url`	Management Console URL and port to connect to as in the following examples: • Embedded Management Console: `http://localhost:50080` • Management Console on Tomcat: `http://192.168.0.101:8080/ManagementConsole`
`--mc-user`	Management Console username.
`--mc-password`	Management Console password.
`--locale`	Explicitly specifies locale, such as "ja". By default locale is retrieved from system settings.
`--log`	Sets logging level. Possible values are `CRITICAL`, `ERROR`, `WARNING`, `INFO`, `DEBUG`.

Option	Description
`--open-doc`	Opens the Analyzer documentation.
`--dump`	If this parameter is specified, the Analyzer records some raw data to a temporary directory for debugging purposes.

> **Note** Process Discovery Analyzer saves configuration of the last run. Next time you start the Analyzer without options, it uses the configuration saved during the previous run.
>
> Windows application and Windows service configurations are saved separately.

Analyzer default settings

```
--mc-url=localhost:50080
--log=info
--locale=<system settings>
```

Deploy Analyzer using Docker on Linux

This procedure provides basic steps for deploying Process Discovery Analyzer on a standalone server on a Linux-based system using Docker.

1. Start MySQL server.

   MySQL server can also be deployed using Docker. You can find the latest MySQL server image on the official Docker Hub website.
2. Start Management Console and specify Process Discovery Analyzer parameters. Use the default settings where they are available in the following fields.
   a. In the Management Console, expand the **Settings** menu and click **Process Discovery Analyzer**.
   b. Specify Process Discovery Analyzer database connection settings[6], runtime parameters, cluster settings, and supply the database root password.
   c. Perform database provisioning by clicking **Create schema** and **Grant user access** on the **Database provisioning** tab. For details, see the Process Discovery Analyzer.
3. Download `Kofax_RPA_ProcessDiscoveryAnalyzer_11.1.0.0_<buildnumber>.zip` and extract all files from the archive to a folder on your computer.
4. Download Docker from https://www.docker.com and install it on your computer.
5. You can add the current user to the Docker group to avoid being asked to type `sudo` with every Docker command. For example, to add a "kofax" user, replace `docker:x:<n>` with `docker:x:<n>:kofax` in the `/etc/group` file. Log out and log in or reboot to update permissions.
6. Navigate to the folder with the Dockerfile from the archive.
7. Build the Analyzer Docker image by running the following command:

   ```
   docker build --tag process_discovery_analyzer_11.1.0.0 .
   ```

[6] This is an IP address or a domain name of the computer running MySQL server or the Docker container with MySQL server. Note that you cannot specify `localhost` or `127.0.0.1` as the address.

8. Specify environment variables of the container and start the Analyzer. The Analyzer starts with the following default settings:

   ```
   MC_URL=localhost:50080
   LOG=info
   LOCALE=<system settings>
   ```

   You can set the following environment variables:

   - MC_URL: Management Console URL
   - MC_USER: Username
   - MC_PASSWORD: Password
   - LOG: Sets logging level. Possible values are CRITICAL, ERROR, WARNING, INFO, DEBUG.
   - LOCALE: Sets the preferred language for the Analyzer.
   - DUMP: Dumps some intermediate data into a temporary folder.

   To start the Analyzer, use the docker run command and specify environment variables at the container startup as follows:

   ```
   docker run -it [-e <ENV VAR>=<VALUE>] process_discovery_analyzer_11.1.0.0
   ```

   To specify environment variable DUMP, use the following pattern:

   ```
   -e DUMP=True
   ```

   For the list of Analyzer options and default settings, see Process Discovery Analyzer options.

   To get the Analyzer container ID, run the following command:

   ```
   docker ps
   ```

   To stop the Analyzer, run the following command:

   ```
   docker stop <container-id>
   ```

   To apply new settings to the Analyzer, first stop the container and then restart it with new parameters.

Deploy Process Discovery Analyzer on Windows

Use the following information to run Process Discovery Analyzer under Windows without using Docker.

> **Important** Before deploying Process Discovery, make sure Java Runtime Environment is installed on the computer running Process Discovery Analyzer and JAVA_HOME variable that points to the JRE installation folder is added to system variables.

To deploy Process Discovery Analyzer on Windows without Docker, you must have a running MySQL server.

1. Start MySQL server.

2. Start and open Management Console and specify Process Discovery Analyzer parameters. Use the default settings where they are available in the following fields.
   a. In the Management Console, expand the **Settings** menu and click **Process Discovery Analyzer**.
   b. Specify Process Discovery Analyzer database connection settings[7], runtime parameters, cluster settings, and supply the database root password.
   c. Perform database provisioning by clicking **Create schema** and **Grant user access** on the **Database provisioning** tab. For details, see the Process Discovery Analyzer.
3. Download `Kofax_RPA_ProcessDiscoveryAnalyzer_11.1.0.0_<buildnumber>.zip` and extract the archive to a folder on your computer.
4. Navigate to the extracted files and run `Kofax_RPA_ProcessDiscoveryAnalyzer_11.1.0.0_<buildnumber>.msi`.
   - By default, the Process Discovery Analyzer is installed as a Windows application in `C:\Program Files (x86)\Kofax RPA Process Discovery Analyzer 11.1.0.0.<buildnumber> x32\`. You can change the installation path if necessary.

     > **Note** Any previous version of the Process Discovery Analyzer is removed when you install a new version of the Analyzer.

     Once installed, navigate to `KofaxRPAProcessDiscoveryAnalyzer.exe` and run it. To specify options other than default, run Analyzer in the Command Prompt window with specified options. See Process Discovery Analyzer for details.
   - If you want to install Process Discovery Analyzer as a Windows service, select **Register as a Windows Service** on the **Destination folder** step of the installer.

     To start Process Discovery Analyzer at Windows startup, select **Automatic startup**.

     Analyzer starts automatically after installation completes. To specify options other than default for the service, perform the following steps.

     a. Go to **Control Panel** > **Administrative Tools** > **Services**.
     b. Right-click the **Kofax RPA Process Discovery Analyzer** service and select **Stop** to stop the service.
     c. Right-click the **Kofax RPA Process Discovery Analyzer** service and select **Properties**.
     d. Specify parameters in the **Start parameters** field and click **Start** to start the service.

     To start the Process Discovery Analyzer as a Windows application, first stop the **Kofax RPA Process Discovery Analyzer** service, then navigate to `KofaxRPAProcessDiscoveryAnalyzer.exe` and run it as described above.

Analyzer default settings

```
MC_URL=localhost:50080
LOG=INFO
LOCALE=<system settings>
```

For the list of Analyzer options and default settings, see Process Discovery Analyzer options.

[7] This is an IP address or a domain name of the computer running MySQL server. Note that you cannot specify `localhost` or `127.0.0.1` as the address.

Deploy Process Discovery Analyzer on Linux

Use the following information to run Process Discovery Analyzer under Linux without using Docker.

Important Before deploying Process Discovery, make sure Java Runtime Environment is installed on the computer running Process Discovery Analyzer and `JAVA_HOME` variable that points to the JRE installation folder is added to system variables.

1. Start MySQL server.
2. Start and open Management Console and specify Process Discovery Analyzer parameters.
   a. In the Management Console, expand the **Settings** menu and click **Process Discovery Analyzer**.
   b. Specify Process Discovery Analyzer database connection settings[8], runtime parameters, cluster settings, and supply the database root password.
   c. Perform database provisioning by clicking **Create schema** and **Grant user access** on the **Database provisioning** tab. For details, see the Process Discovery Analyzer.
3. Download `Kofax_RPA_ProcessDiscoveryAnalyzer_11.1.0.0_<buildnumber>.zip` and extract it to a folder on your disk.

   The extracted folder contains two installation packages:

   - `Kofax_RPA_ProcessDiscoveryAnalyzer_11.1.0.0_<buildnumber>.deb` for Debian-based systems
   - `Kofax_RPA_ProcessDiscoveryAnalyzer_11.1.0.0_<buildnumber>.rpm` for Red Hat-based systems
4. To install the Analyzer on your system, run the appropriate command.
   a. For a Debian-based system, use the following command to install or upgrade the Analyzer:

   ```
   sudo dpkg -i
   Kofax_RPA_ProcessDiscoveryAnalyzer_11.1.0.0_<buildnumber>.deb
   ```

   To uninstall the Analyzer, run: `sudo dpkg -P kofax-rpa-processdiscoveryanalyzer`

   b. For a Red Hat-based system, use the following command to install or upgrade the Analyzer:

   ```
   rpm -Uvh Kofax_RPA_ProcessDiscoveryAnalyzer_11.1.0.0_<buildnumber>.rpm
   ```

   To uninstall the Analyzer, run: `sudo rpm -e kofax-rpa-processdiscoveryanalyzer`

   By default, the Analyzer is installed at `/opt/KofaxRPA/ProcessDiscoveryAnalyzer`. Now you can run it using the command in the following step.
5. To run the Analyzer, navigate to `/opt/KofaxRPA/ProcessDiscoveryAnalyzer` and run the following command.

   ```
   ./KofaxRPAProcessDiscoveryAnalyzer
   ```

Analyzer default settings

```
MC_URL=localhost:50080
LOG=INFO
```

[8] This is an IP address or a domain name of the computer running MySQL server. Note that you cannot specify `localhost` or `127.0.0.1` as the address.

```
LOCALE=<system settings>
```

For the list of Analyzer options and default settings, see Process Discovery Analyzer options.

Process Discovery Analyzer cluster

Automatic process detection using machine learning algorithms and increased data flow in Process Discovery require more processing power. To streamline the need for efficient and effective data processing, Process Discovery 11.1 introduces Analyzer cluster configuration for high performance Big data analysis.

Process Discovery Analyzer cluster is based on Apache Spark technology. A cluster is a distributed computing framework consisting of several computers acting as cluster nodes that run Process Discovery Analyzer application. Spark cluster consists of a master node and one or more worker nodes. The same Analyzer application is installed on a master and each worker node. The master node orchestrates the work of a cluster and takes part in data processing by running the Analyzer worker process. Worker nodes run Analyzer worker processes to analyze data. You can mix Analyzer cluster nodes running on different OS. Master node is assigned by specifying its address in Management Console.

The Apache Spark cluster technology implies that you use computers similar in processing power and memory amount as worker nodes. If you have computers with great difference in RAM and processing power that you want to be cluster nodes, use containers (for example, Docker container) on computers with large amount of RAM to run the Analyzer. This way you can create several clusters with characteristics similar to less powerful nodes.

Important We recommend using the most powerful computer among nodes as a master node to balance the heavier load.

The minimum (default) number of nodes is one node. Depending on the amount of data Process Discovery Agents collect and your requirements for the analysis time, you can add as many cluster nodes as needed. For example, if you want to decrease the Analyzer data processing time, add a node to the cluster and check the processing time again. The processing time is presented in the Status report of the Process Discovery view in Kofax Analytics for RPA views.

Note that adding twice as many nodes does not decrease processing time twice as much, because some time is used for co-ordinating the nodes and transmitting data between nodes. Therefore, reducing network latency increases cluster performance. Also, it is more efficient to add one powerful server with a large amount of RAM as a node instead of adding several less powerful. Because Analyzer uses all available processing power of the system, we recommend using dedicated computers as nodes in the cluster.

Set up Process Discovery Analyzer cluster

1. Under **Process Discovery Analyzer** > **Cluster settings** in Management Console, specify all necessary parameters, such as assign the master node, specify network pattern (optional) and other settings. See Process Discovery Analyzer for details.
2.
   a. Install, configure, and start Process Discovery Analyzer on computers that you want to be Analyzer cluster worker nodes.

   b. After starting all worker nodes, install, configure, and start Process Discovery Analyzer on a computer that you want to be Analyzer cluster master node.

   When starting instances of the Analyzer application on the nodes, specify the Management Console address where Analyzer cluster settings are defined. Specify other parameters if necessary. See Process Discovery Analyzer for details.

   Once all nodes are running, you can add, remove, and configure worker nodes as required. The changes are applied during the next Analyzer run.

If any of the worker nodes fails, the underlying Apache Spark technology preserves the data and distributes the load between working nodes. If you change any Analyzer settings, restart the master node. If you assign another computer as a master node, restart both the current master node and the newly assigned master node. For example, currently your cluster contains three nodes: A, B, and C. "A" is a master node. If you assign "B" as a master node, restart "A" and "B".

If you encounter an `outofmemory` type of error in the Analyzer log on a master node, open **Settings** > **Process Discovery Analyzer** > **Cluster settings** in the Management Console, increase the amount of memory in the **Master memory (GB)** setting, and restart the master node. See "Cluster settings" in the Process Discovery Analyzer for details.

Monitor cluster nodes

Apache Spark cluster includes a tool to monitor node activities. Once you set up a cluster in your environment, open the master node dashboard to make sure all worker nodes are alive. The dashboard contains some basic information about running and completed applications as well as a list of cluster workers. You can see the application details by clicking the application ID in the list. To open the master node dashboard in a browser, type the master address followed by the port number specified in the **Master WebUI port** option in the Management Console. For example:

```
10.10.0.15:8080
```

By clicking a worker ID in the list, you can open the worker dashboard. To open the worker dashboard directly in a browser, type the worker address followed by the port number specified in the **Worker WebUI port** option in the Management Console. For example:

```
10.10.0.11:8081
```

Master and Worker node logs reside at the same location as the general Analyzer log file. To locate the log file in your system, see the **Log files** section in Process Discovery Analyzer.

For more information about Apache Spark cluster, see Apache Spark documentation at https://spark.apache.org/.

Analytics

The Process Discovery views in the Kofax Analytics for RPA dashboard provide reports with information based on data collected by the Process Discovery Agents. The view includes a variety of visual and analytical representations of data using charts and tables. System administrators, business process managers, and other stakeholders use this interface to gain visibility into analytical information.

Important For correct display of data in the Insight dashboard, make sure Java correctly sets the time according to your time zone on RoboServers and computers running instances of Management Console. See the *Timezone Data Versions in the JRE Software* on the Oracle website for the latest updates in time zones. If necessary, use the Oracle *Timezone Updater Tool* to update the time zone information.

Chapter 8

Kofax Analytics for RPA

Kofax Analytics for RPA is an extension of Kofax RPA that produces a graphical business intelligence dashboard based on data collected while Kofax RPA robots and Process Discovery agents are working.

Kofax Analytics for RPA views are based on values from predefined Kofax records and metrics. When using the Dashboard Designer within Insight Studio to add custom views, do not modify the predefined views, records, or metrics that come with the product. You can make a copy of existing views and then customize the settings. For details, see the *Kofax Analytics Project Customizations Application Note* on the Kofax website.

Time zone settings

For correct display of data in the Kofax Analytics for RPA dashboard, make sure Java correctly sets the time according to your time zone on RoboServers and computers running Management Consoles. See the *Timezone Data Versions in the JRE Software* on the Oracle web site for the latest updates in time zones. If necessary, use the Orale *Timezone Updater Tool* to update the time zone information.

Install and Configure Kofax Analytics for RPA

Follow the instructions in this section to perform a new Kofax Analytics for RPA 2.3 installation.

If you are upgrading from the previous version of Kofax Analytics for RPA, see Upgrade from the previous version.

Your Kofax Analytics for RPA product includes `Kofax_RPA_KAFRPA_2.3.zip`, the project bundle file required to successfully import RPA components and built-in views.

SSL

If you initially perform the Kofax Analytics for RPA installation without SSL, you can switch to SSL later. You can switch to SSL by reinstalling Kofax Insight with the "Use Existing Database" option.

Installation and Configuration Checklist

The following checklist provides an overview of steps necessary to set up Kofax Analytics for RPA.

1. Check settings and set up if necessary the following parameters in Kofax RPA. The settings are located in **Settings** > **General** in Kofax RPA Management Console.

   For robot-run analytics

   - RoboServer log database: Select **Log to database** and specify database parameters.
   - Analytics database: Select **Log the statistics to Analytics database** and specify database parameters. The SQL scripts needed to create the Analytics schema are located in the `documentation\sql\statistics` directory in your Kofax RPA installation directory.

   For Process Discovery analytics

   Deploy Process Discovery. See Process Discovery for details.

2. Create and configure Insight databases: admin, metadata, and data. See the "Databases" topic below.
3. Install Kofax Insight.
4. Import and configure your project.

Databases

Use a new or existing database management system instance to create the following databases for use with Kofax Analytics for RPA:

- Metadata: Stores configuration information such as metric definitions and calculation logic. Assign a name such as `rpa_analytics_meta`.
- Data: Stores the processed records and metrics. You can assign any name, such as `rpa_analytics_data`.
- Admin: Stores Insight administrative data related to users, roles, filtering, alerts, logs, and more. Assign a name such as `rpa_analytics_admin`.

We recommend that you use a consistent naming convention for each database, such as rpa_analytics_admin, rpa_analytics_meta, and rpa_analytics_data.

> **Important** Please create and maintain all the Kofax RPA product databases according to the recommendations in the product documentation. If you are considering database modifications or customizations, do not proceed without consulting Kofax; otherwise, the results are unpredictable and the software may become inoperable.

For the Oracle database, all database schemas must already be in place before you proceed with the product installation.

> **Note** If you use an Oracle database for collecting analytics data in Kofax RPA and specify the same database while upgrading to a new version of Kofax RPA, you must manually drop and create tables in the database. See the "RoboServer log database" and "Scripts for Creating Database Tables" topics in Kofax RPA help for information on creating database tables.

Install Kofax Insight

Kofax Insight produces a graphical business intelligence dashboard based on robot-run data and data collected by Kofax RPA Process Discovery Agents.

For more detailed installation instructions, see the *Kofax Insight Installation Guide*. Kofax Analytics for RPA works on Kofax Insight version 6.2 and later. We recommend using Kofax Insight 6.2.1 or later for better results.

Important Before starting the Insight installation process, install MySQL Connector/NET on the computer running Kofax Insight. You can download MySQL Connector/NET from the MySQL website.

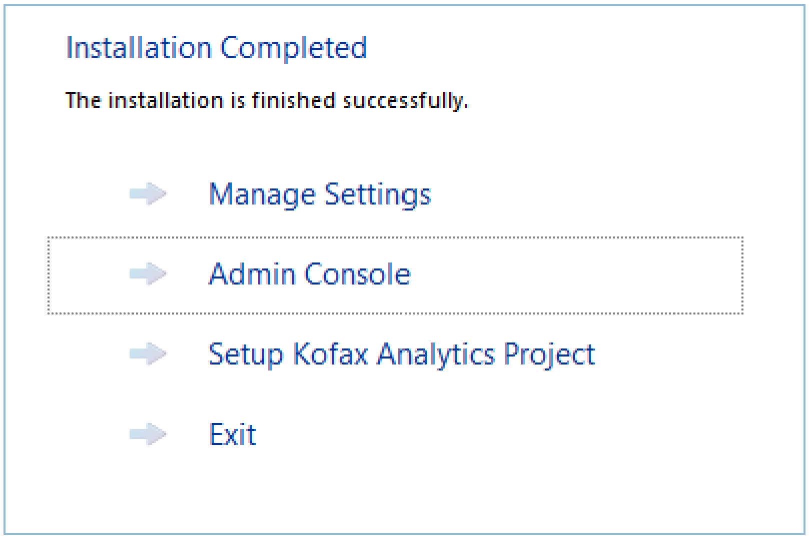
1. Run the Kofax Insight installation file, such as `KofaxInsightSetup_6.2.1.0.0.<build>_x64.msi`. The Insight build number is listed in the "Version information" section of the *Kofax Insight Release Notes*. Install Kofax Insight using the default options.

   Install the latest fix pack either after completing the Kofax Insight Setup Wizard or after performing the entire Kofax Analytics for RPA configuration procedure by running the fix pack installation file.

   When the installation wizard is complete, the **Completed the Kofax Insight Setup Wizard** window appears.
2. Click **Finish**. The Insight Installation Manager is started. If you clear the **Launch Insight Installation Manager** option before clicking **Finish**, you can start the Installation Manager later from the Start menu. Go to **Start** > **Insight 6.2.1** > **Administration** > **Installation Manager**.

   The Installation Manager continues the installation process and helps you set up and configure Kofax Insight, such as the Insight admin database and other parameters. See the *Kofax Insight Installation Guide* for details.

Once the Insight installation is complete, the final screen of the Insight Installation Manager provides several options:



Installation Completed

The installation is finished successfully.

To set up the dashboard project, click **Admin Console** and continue to the next section.

Import and configure your project

Use this procedure to set up your Kofax RPA dashboard project.

Note that by default Insight sets up the **User's logins and passwords are defined within Insight platform. Users are explicitly linked to roles** option for your project. You can change the type of authentication later.

1. Start Insight Admin Console.
   - If you have not closed the Insight Installation Manager, on the Installation Completed window, click **Admin Console**.
   - If you closed the Insight Installation Manager, navigate to **Start** > **All Programs** > **Insight 6.2.1** > **Admin Console**.
2. Enter the login credentials for the Administrator that you specified during the Insight installation.

   If this is the first time you start the Admin Console, you are prompted to enter license information on the **License** tab.
3. In the navigation panel, click **Projects**, and then right-click and select **New Project**.

   The **New Project** window appears.
4. Type `Kofax_RPA` as the project name and click **OK**.

   The **Create New Project [Kofax_RPA]** window appears.
5. Click the **Import from file** tab, and then click **Select file**. Select **File is located on the client computer and will be copied to the server for processing**.

   The **File Upload** window appears.
6. Navigate to `Kofax_RPA_KAFRPA_2.3.zip`, and click **Open**.

   Wait until the file is imported.
7. Set the database connection information for the Meta and Data databases.

   The database name field is automatically pre-filled with the project name and a database ending, such as `Kofax_RPA_meta`. Edit the name to suit your requirements. Note that the database name must not contain spaces.

   > **Tip** If the server details are nearly the same for the Meta and Data databases, select **Same server as the meta database** to copy the entries from the **Meta** section into the **Data** section. After the entries are copied, be sure to update the database name in the **Data** section.
   >
   > Do not click **Connect** during this step, because the databases do not exist yet.
8. Click **OK** to import the project.
9. When prompted to confirm database creation, click **Yes** even if databases already exist. Insight creates the necessary tables in the database.

   An indicator appears on the screen while the project import is in progress.

   The built-in views, records, and metrics for Kofax RPA are added to your installation.

10. Expand the newly created project and verify that you have the following databases under **Data Sources**:
    - **Data DB**: Insight data warehouse
    - **Analytics**: Reporting database that you have configured in Kofax RPA Management Console under **Settings** > **General** > **Analytics database**
    - **Logs**: Log database that you have configured in Kofax RPA
    - **erd_dean**: Process Discovery database

    To expand the project, click the arrow to the right of the project name.
11. Set parameters to connect to each of the databases by specifying the Server name, Database name, User and password. Click **Connect** to test the connection. The circular indicator turns green if the connection is successful.

    > **Note** The analyzer database is set in the Kofax RPA Management Console in **Settings** > **Process Discovery Analyzer Database settings**.

12. Click **Save** to save your configuration.
13. Expand **Users** and set passwords for the built-in users that can access Kofax RPA Viewer directly to view the reports. Click **Save** to save passwords. The following users are available:
    - **KAFK**: This user can access both Process Discovery and RPA dashboards.
    - **RPA**: This user can access only the RPA dashboard.
    - **PD**: This user can only access the Process Discovery dashboard.
    - **KAFK_admin**: This user is preserved for backward compatibility with version 1.0 and has similar rights as the Administrator.
14. Open Insight Studio by clicking **Studio** from the Insight Admin Console, or open a web browser and enter the following URL:

    ```
    http://<server>/Insight/Studio/
    ```
15. Expand **Execution Plans** and click **One minute plan** for the imported Kofax RPA project.

    This plan is scheduled to load data every five minutes.
16. Click **Data Load** and do the following on the **Date range** tab:

    a. Select today's date and tomorrow's date to load and show data gathered today.

    b. Click **Load Data**.

    To load data collected during previous days, load data day by day instead of loading the entire time period. Loading data for more than one day increases the load and may prevent the program from loading data.

    Once configuration is complete, you can start using the dashboard.

Upgrade from the previous version

Kofax Analytics for RPA 2.3 is designed to use Kofax Insight 6.2 with the latest fix pack. We recommend using Kofax Insight 6.2.1 or later for better results.

For Kofax Insight upgrade, see the "Upgrade Insight" chapter in the *Kofax Insight Installation Guide*. You can also consult Kofax support before performing an upgrade.

Windows Authentication

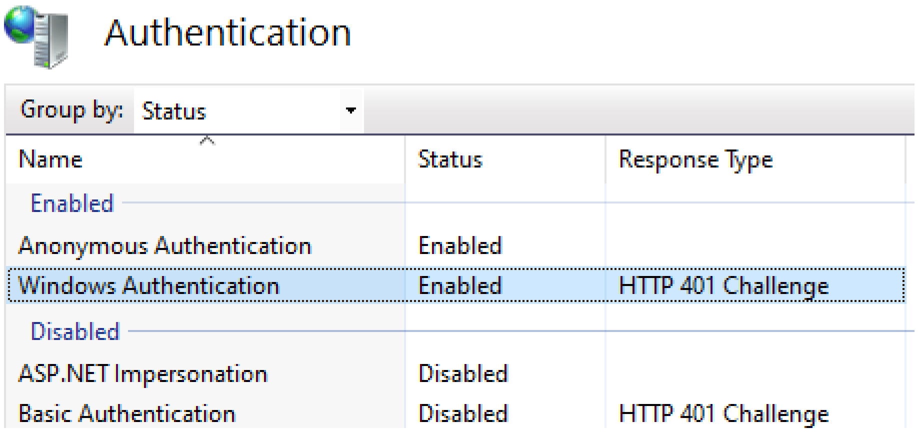
If you are using Windows user authentication, access to all Kofax Analytics for RPA databases must be given to the following:

1. User account used to perform the Kofax Analytics for RPA installation
2. Account/identity for the IIS application pool
3. Account/identity for the Insight Scheduler service

Configure Insight for Windows Authentication

Use the procedure in this section to configure Kofax Insight for Windows Authentication.

Before configuring Insight, select Windows authentication in IIS for the web application (default website).



Authentication

Group by: Status

Name	Status	Response Type
Enabled		
Anonymous Authentication	Enabled	
Windows Authentication	Enabled	HTTP 401 Challenge
Disabled		
ASP.NET Impersonation	Disabled	
Basic Authentication	Disabled	HTTP 401 Challenge

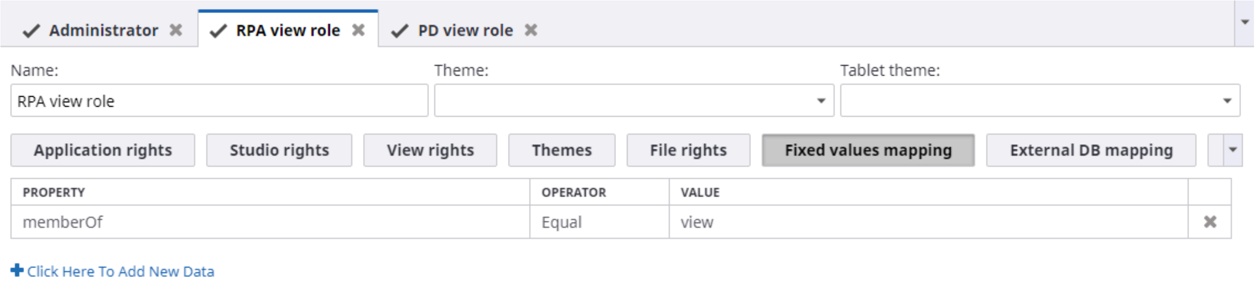
1. Navigate to **Start** > **All Programs** > **Insight 6.2.1** > **Administration** > **Admin Console**.
2. In the navigation panel, click **Authentication**.
3. Click the **Authentication Method** tab and select the following:
    a. **User properties are obtained from the environment: Windows**
    b. **And then user roles and access rights are determined by comparing these values to**: **Fixed values**
4. **User identifier**: Specify a way to get the user's ID. The ID should be constant for a specific user's login. Usually, it is a session property (Identity) that looks to the Active Directory domain and user name.
    a. In the navigation panel, click **User mapping**.
    b. On the **User Mapping** tab for User Identifier (UID), set the "Session property" to **Identity**.

5. Set session properties for **User Name** and **Email**.
   a. On the **User Mapping** tab for **User Name**, set the "Session property" to **FullName**.
      User Name is the display name of a user account. Usually, it is one of the Active Directory properties, such as *Identity*, *name*, *FullName*, *displayName* or another convenient property. Your Domain Admin can provide all available Active Directory properties.
   b. On the User Mapping tab for Email, set the "Session property" to **EmailAddress**.
      Email is the email address of the user account. It is used for self-subscriptions only. Usually, it is the Active Directory property *mail* or *EmailAddress*. Your Domain Admin can provide you with all available Active Directory properties.

Mapping Roles

Roles define a set of predefined Admin settings such as the theme, date format, etc. Also, roles define specific access rights to projects and dashboard views. It's necessary to describe mapping rules for each role. Usually, the Active Directory property *memberOf* is used. In the sample illustration, users with the Active Directory property *memberOf*, including the *admin*, are assigned to the KAFK admin role. Your Domain Admin can provide you with all available Active Directory properties.

Each row in the mapping grid uses the **AND** operand. If multiple roles on the Roles list match conditions for a user account, the access rights are merged from all matching roles, while other settings (such as the theme or date format) are assigned by the top matching role on the list.



Name: RPA view role

Theme:

Tablet theme:

PROPERTY	OPERATOR	VALUE
memberOf	Equal	view

Use Kofax Analytics for RPA

With Kofax Analytics for RPA, you can view the data retrieved from your Log and Analytics databases in an interactive dashboard through the Viewer.

The Kofax Analytics for RPA dashboard consists of several built-in views. Each view has graphical elements and other components to analyze data in your Kofax RPA database.

Viewer

Use the Viewer to display the dashboard views included in your Kofax Analytics for RPA project. These views include a variety of visual and analytical representations of data using charts and tables. System administrators, business process managers, and other stakeholders use this interface to gain visibility into analytical information.

Note that Viewer data is updated every five minutes.

Open the Viewer

Depending on the authentication method, the procedure for opening the Viewer may vary. Follow the procedure in this section, depending on the authentication type.

Insight Users Authentication

Do one of the following to open the Viewer:

- Navigate to **Start** > **All Programs** > **Insight 6.2.1** > **Viewer**.
- Open a web browser and enter the following URL:

  `http://<server>/Insight/View/`

  where `<server>` is the name or IP address of your computer running Insight.

  A login window appears. Enter one of the user names (KAFK, RPA, or PD) and a password set for this user in the Insight Admin Console. Click **Login**. Note that only one of the above-mentioned users can access the Viewer directly. See Import and configure your project for user configuration details.

  > **Note** Be sure to verify that the website's binding host name is set to blank or localhost in your IIS settings. Otherwise, a login error may occur.

To ensure proper viewing, verify the proper Authentication Method setting in Insight Admin Console. The "No authentication" setting is not supported and may produce errors or unexpected results in the browser window. For details, see Import and configure your project.

Also, if Main is not selected as the default view for the role associated with the user who is logging in, the following error may appear:

```
You can specify view name parameter in Admin Tool
```

Resolve the issue by setting Main as the default view.

Windows Users Authentication

Do one of the following to open the Viewer:

- Navigate to **Start** > **All Programs** > **Insight** > **Viewer**.
- Open a web browser and enter the following URL:

  `http://<server>/Insight/View/`

  where `<server>` is the name of your server

  Be sure to verify that the website's binding host name is set to blank or localhost in your IIS settings. Otherwise, a login error may occur.

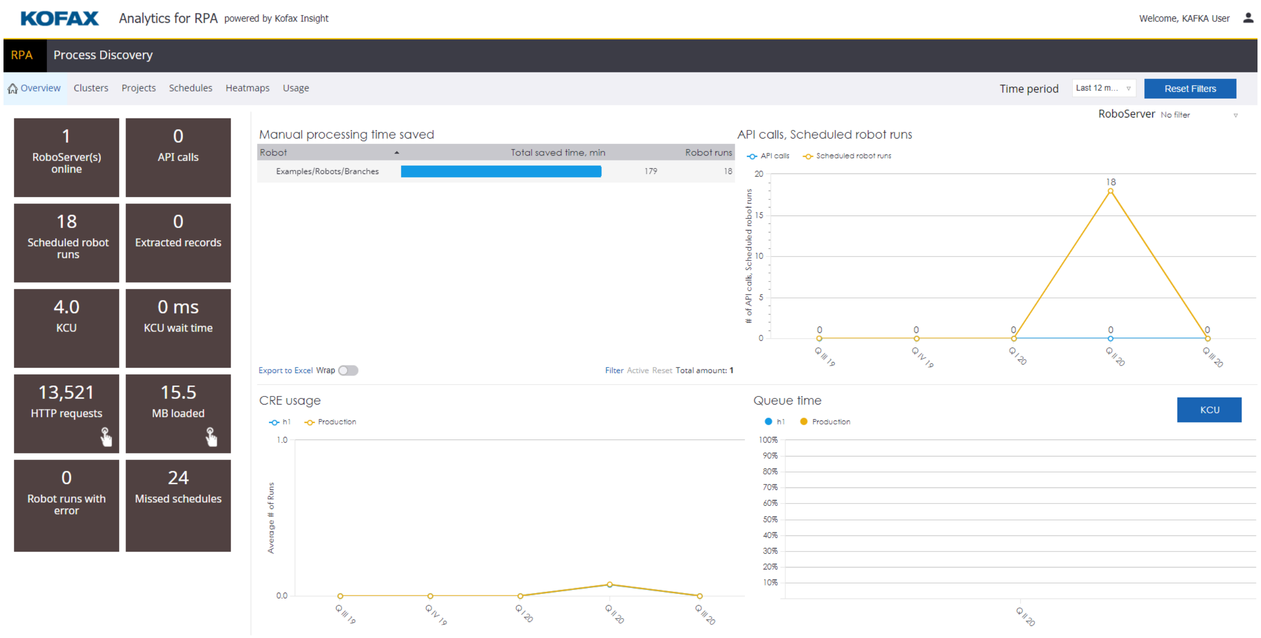
Viewer screen layout and navigation

Toolbar Menu Use the menu to access different sections, such as **RPA** and **Process Discovery** as well as different views in the selected section. Use additional user-level items to bookmark frequently used views and log out.

Context Menus and Chart Options

Right-click a chart or grid to access more options such as Reset, Pivot, Zoom in, View, and so on. The options vary, based on the chart type or area where you right-click.

RPA views



KOFAX Analytics for RPA powered by Kofax Insight

Welcome, KAFKA User

RPA | Process Discovery

Overview | Clusters | Projects | Schedules | Heatmaps | Usage

Time period: Last 12 m...

RoboServer: No filter

1 RoboServer(s) online	0 API calls
18 Scheduled robot runs	0 Extracted records
4.0 KCU	0 ms KCU wait time
13,521 HTTP requests	15.5 MB loaded
0 Robot runs with error	24 Missed schedules

Manual processing time saved

Robot	Total saved time, min	Robot runs
Examples/Robots/Branches	179	18

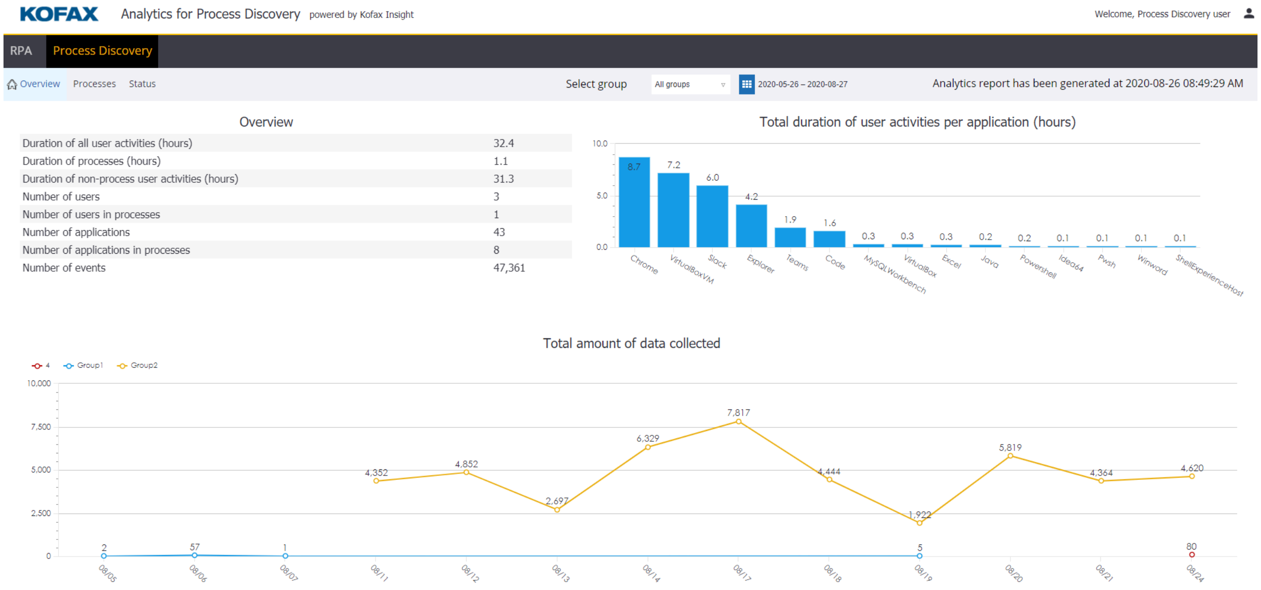
Export to Excel Wrap

Filter Active Reset Total amount: 1

RPA view filters

You can filter the reports by the time in the **Time period** list and by RoboServers where robots run in the **RoboServer** list.

Process Discovery views


KOFAX
Analytics for Process Discovery
powered by Kofax Insight
Welcome, Process Discovery user
RPA
Process Discovery
Overview
Processes
Status
Select group
All groups
2020-05-26 – 2020-08-27
Analytics report has been generated at 2020-08-26 08:49:29 AM

Overview

Duration of all user activities (hours)	32.4
Duration of processes (hours)	1.1
Duration of non-process user activities (hours)	31.3
Number of users	3
Number of users in processes	1
Number of applications	43
Number of applications in processes	8
Number of events	47,361

Process Discovery filter input

Filter the results by the Process Discovery group name and date.

Select a group name in the **Process Discovery Group** list. If you select **No filter**, information for all groups is displayed in the view.

Use the Viewer

While working with the Viewer, you can do the following:

- Click any Project, Robot, or Schedule name in the list to drill down to activity details.
- Manipulate and interact with any component on the dashboard.
- Hover over a point in a graph to view details.
- Double-click any graph name to expand it to the full browser window.
- Click and drag to the side any graph name to switch to the list view and back.
- Select the period for the statistics in the **Time period** list.
- To restore default values in filters, click **Reset Filters**.

Export to Excel

On any grid with Export to Excel functionality, you can set preferences for exporting the content of the grid to Microsoft Excel.

1. On a grid with **Export to Excel** enabled, click **Export to Excel**.

   The **Export to Excel** window appears.

2. Select an **Export content** option:
   - **Current page and top drill down**
   - **All pages of a grid. top drill down level. Could take some time.**
   - **All pages of a grid, all drill down levels. Could take much longer.**
3. Select an **Export format** option:
   - **Unformatted XML file, readable by Excel**
   - **Formatted Excel file**
   - **Unformatted CSV file**
   - **Unformatted TSV file**
4. Optionally select **Enabled** to use a flat export, which retains granular details for drill down or expandable data.
5. Click **OK** and when prompted, save or open the .xml file.

Kofax Analytics for RPA views

This section gives you information about each view provided in your Kofax Analytics for RPA project. You can interact with and manipulate different objects on the view to get detailed information. Kofax Analytics for RPA provides RPA views and Process Discovery views.

RPA views

RPA views provide Kofax RPA analytics information.

- Overview
- Clusters
- Projects
- Schedules
- Heatmaps
- Usage

Overview

The Overview provides a graphical representation of Kofax RPA system performance parameters, so that you can quickly see capacity bottlenecks or system utilization issues. You can filter the data by selecting a period in the **Time period** list and selecting a RoboServer in the **RoboServer** list.

The percentage on tiles in the left pane shows a difference between the selected period and the same period preceding the selected one. For example, if you select "Last hour" as the time period, the percentage shows the difference between the last hour and an hour preceding it. Note the triangle next to the value points down if the value decreased and points up if the value increased.

- For API Calls, Scheduled robot runs, Extracted records, HTTP requests, and Mb loaded: The percentage is green if the value is more than zero, indicating the value has increased.
- For KCU, KCU wait time, Robot runs with error, and Missed schedules: The percentage is green if the value is less than zero, indicating the value has decreased.

Click a tile with this symbol 👆 to view network details.

Manual processing time saved

This graph provides information on the difference between the value specified in the **Human Processing Time** option in the Robot Configuration dialog box in Design Studio and the actual duration of the robot run calculated for all robot runs during the specified period. In the **Human Processing Time** option you can specify the time required for a person to perform the same task that the selected robot performs during its run.

API calls, Scheduled robot runs
Number of robot runs started via API or manually.

CRE usage

An average number of concurrently executed robots using CRE license. For details about CRE, see "Concurrent Robot Execution License" in the *Kofax RPA Installation Guide*

> **Note** By default, the Overview report shows the CRE usage information. To see the KCU usage, click the **KCU** button in the right pane of the view.

Queue time
Ratio between the total execution time and queue time in percent.

KCU usage

Average use of KCUs by cluster.

Wait time
Ratio between the total execution time and KCU wait time in percent.

KCU Health

KCU (Kofax RPA Compute Units) Health is calculated based on two metrics: one for API calls and one for Schedules. For details about KCU, see "Kofax RPA Compute Units" in the *Kofax RPA Installation Guide*.

The KCU health for API calls is calculated based on KCU wait time from robot runs.

- The KCU health becomes Yellow if the maximum KCU wait time for a robot run is more than one second.
- The KCU health becomes Red if the average KCU wait time for a robot run is more than one second.

The KCU health for schedules is drawn from the statistics as follows: zero base is 2 seconds per minute, and the peak limit is 10 seconds per minute.

- The KCU health becomes Yellow if 10% of the data points have a KCU wait that exceeds the peak limit, or if 50% of the data points have a KCU wait that exceeds the zero base.
- The KCU health becomes Red if 50% of data points have KCU wait that exceeds the peak limit, or if 90% of data points have a KCU wait that exceeds the zero base.

Clusters

This view provides a graphical representation of Kofax RPA system performance parameters on a cluster level. The CPU and Memory graphs show average usage for all RoboServers in a cluster.

RoboServer names change color depending on the last activity as follows.

- Green: If the value is greater than 60 seconds.
- Orange: If the value is greater than or equal to 60 seconds, and less than 5 minutes.

- Red: If the value is greater than or equal to 5 minutes, and less than 1 day.
- Gray: If the value is greater than or equal to 1 day.

You can filter the results by selecting a RoboServer in the list. When you select a RoboServer, the CPU and Memory graphs show minimum, maximum and average usage.

Projects

This view provides an overview of robot health in your projects by displaying a graphical presentation of key robot-run metrics. You can identify and fix robots that are not performing as expected. The following columns and rows are available in the view.

Field name	Description
Projects	
Project	Displays a project name.
Scheduled	A number of scheduled robot runs in the project during the selected time period.
API calls	A number of API calls during the selected time period.
Health	Color circle that represents the overall health of the project. See Robot Health and Color for details.
Robots	
Robot	Robot name and path in the selected project.
Execution time	The average execution time of one robot run.
Total saved time, min	An estimated difference between the manual and automatic (by robots) task execution during the selected time period in minutes.
Extracted records	The average number of extracted records during one robot run.
KCU	The average number of KCU points used during one robot run.
Errors	The average number of errors in one robot run.
TAG	
Tag	Names of tags in robots. Clicking a tag opens a list of all robots that contain the selected tag.

Robot Health and Color

Robot health is calculated based on three values: errors count, execution time, and extracted records. For each robot run for a selected period, a mean and deviation is calculated. If all values fall into mean plus/minus deviation, the robot run is Green. Otherwise, the robot run becomes Red.

If all robot runs of a robot are Green, the robot health is Green. If one or more, but not all robot runs are Red, the robot health is Yellow. If all robot runs are Red, the robot health is shown as Red.

Click a row in the list of robots to see the selected robot information, such as execution time, number of extracted records, errors and warnings, and robot run details.

Schedules

This view provides a graphical presentation of the number of concurrent schedule runs, which offers an overview of the load distribution on your system.

To see the details of a schedule run, select a schedule and click **Details** in the **Schedules** pane. You should see the **Schedule details** window. Click a schedule run in the **Schedule run** list to see its details under **Robot runs**.

Schedule Health and Color

Schedule health is calculated based on three values: error count, execution time, and extracted records. For each schedule run for a selected period, a mean and deviation are calculated. If all values fall into mean plus/minus the deviation, the schedule run is Green. Otherwise the schedule run becomes Red.

If all schedule runs of a schedule are Green, the schedule health is Green. If one or more, but not all schedule runs are Red, the schedule health is Yellow. If all schedule runs are Red, the schedule health is shown as Red.

Heatmaps

The RPA dashboard provides color-coded heatmaps for robot runs and schedule runs. The threshold value for a heatmap is specified in the **Enter critical status threshold** option above the report. When you hover your mouse over the report, the number of runs appears in cells. The following colors are available.

Color	Value
Red	Above the specified threshold
Orange	Above three-quarters and up to the specified threshold
Yellow	Above half and up to three-quarters of the specified threshold
Green	Up to the half of the specified threshold

Usage

Provides license usage and robot execution information organized by cluster.

KCU usage

KCU usage Avg
Average use of KCUs.

KCU Usage Peak
Displays the peaks of KCU usage for the selected period.

KCU Wait Time Peak (s)
Displays the peaks of KCU wait time for the selected period.

KCU Wait Time Accumulated
Total KCU wait time for the selected period.

CRE usage

CRE usage Avg
An average number of concurrent robot runs.

CRE Usage Peak
Displays the peaks of concurrent robot executions for the selected period by cluster.

Queue Time
Average time spent by robots in queues for the available license.

Queued Robots
The maximum number of queued robots.

Robot executions

Robot Executions Accumulated
Total number of robot runs.

Robot Execution Time Accumulated
Total robot execution time.

Robot Errors
Total number of robot errors.

Queue Time (s)
Total time spent by robots in queues.

Process Discovery views

Process Discovery views are reports with information based on data collected by the Process Discovery agents and processed by the Analyzer.

Filters and status

- **Select group**: Use this filter to see data for a selected Process Discovery group. To view data for all groups, select **All groups**.
- **Date**: Specify a date range. By default the range is the last 30 days.
- **General status**: Shows the current status of the report. If "Analytics report has been generated at <date time>" is shown, Analyzer successfully finished the analysis and the data is ready for evaluation. Any other status denotes that the report is incomplete. Open the **Status** view for details.

The following charts and graphs are available in the Process Discovery views.

Overview

Provides a summary of all data processed by the Analyzer. The following information is available in this view.

Overview

- **Duration of all user activities (hours)**: Total duration of detected user activities.
- **Duration of processes (hours)**: Duration of detected processes.
- **Duration of non-process user activities (hours)**: Duration of user activities that do not form processes.
- **Number of users**: Total number of analyzed users.
- **Number of users in processes**: Number of users who performed actions that formed processes.
- **Number of applications**: Total number of detected applications.
- **Number of applications in process**: Number of applications used in processes.
- **Number of events**: Number of processed events.

Total duration of user activities per application (hours)
This bar chart helps you detect the time spent working in each application.

Total amount of data collected

This graph provides information on the total number of analyzed events.

Processes

This view provides information about detected processes. You can export the list of discovered processes, the list of process instances, and the list of steps in the process instances to a spreadsheet by clicking **Export to Excel** in the lower left corner of the window. Also, you can export the list of process instances, and the list of steps with screenshots to a PDF file by clicking **Export to PDF** in the **Process details** and **Process instance details** window.

Overview

Provides a summary of the detected processes.

- **Total duration of processes (hours)**: Duration of all detected processes.
- **Number of processes**:Number of detected processes.
- **Number of applications in process**: Number of applications used in processes.
- **Number of users in processes**: Number of users who performed actions that formed processes.

Duration of user activities in processes per application (hours)

The chart shows the time spent working in each application within processes.

Average time per process (min)

The chart shows the average duration of a process instance in the process. Click the process name to the right of the chart to show or hide the process in the chart.

Duration of processes (hours)

The chart shows the total time of the detected processes. Click the process name to the right of the chart to show or hide the process in the chart.

Discovered processes
Lists discovered processes. The following information is displayed for each process. To view process details, click a process in the list.

- **Title**: Name of the process. Process names are created automatically and are comprised of the word "Process" and an ordinal number, such as "Process 22".
- **Most used application**: Name of the mostly used application in the process.
- **Amount of runs**: Number of detected process instances.
- **Total time (hours)**: Total duration of all instances of the process.
- **Similarity (%)**: An average similarity of the process instances.
- **Discovery type**: Shows how the process was discovered. It can be either **Automatic** or **With assistance**. If at least one of the process instance was discovered using user assistance, the Discovery type is shown as **With assistance**. **With assistance** type of the process appears if users use **Start task** and **Stop task** options on the Agent's menu to denote the beginning and the end of the task.
- **Average time (min)**: An average duration of the process instance.
- **Steps**: An average number of steps in the process instance.
- **Users**: A number of users performed the selected process.

Process details

When you click a process in the **Discovered processes** list, the **Process details** window appears. The window shows detailed information about the selected process and a list of process instances. The following information is displayed for each process instance. Click the column title to sort the list of process instances.

- **Instance**: Instance number.
- **Similarity**: Each instance similarity.
- **Discovery type**: The discovery type of the process instance.
- **Steps**: Number of steps in the detected process instance.
- **Duration (min)**: Duration of the process instance.

Process instance details

When you click a process instance in the **Process details** window, the **Process instance details** window appears. The window shows detailed information about the selected process instance and a list of steps with screenshots if available.

Status

This report shows a list of Analyzer events and agents status. Use this report to trace any errors occurred in the Analyzer if its general status is different from "Analytics report has been generated at <date time>".

Using the **Agents recording status** list, you can track agents' activity. The agents are sorted by date with the most recent activities on top. The list helps you detect agents that are turned off or incorrectly set up.

Chapter 9

Kapplets

Kofax RPA Kapplets expose a friendly user interface to robots and help facilitate your work with them. You can interact with robots through the provided Kapplet even if you do not have the knowledge about robots and how they are built.

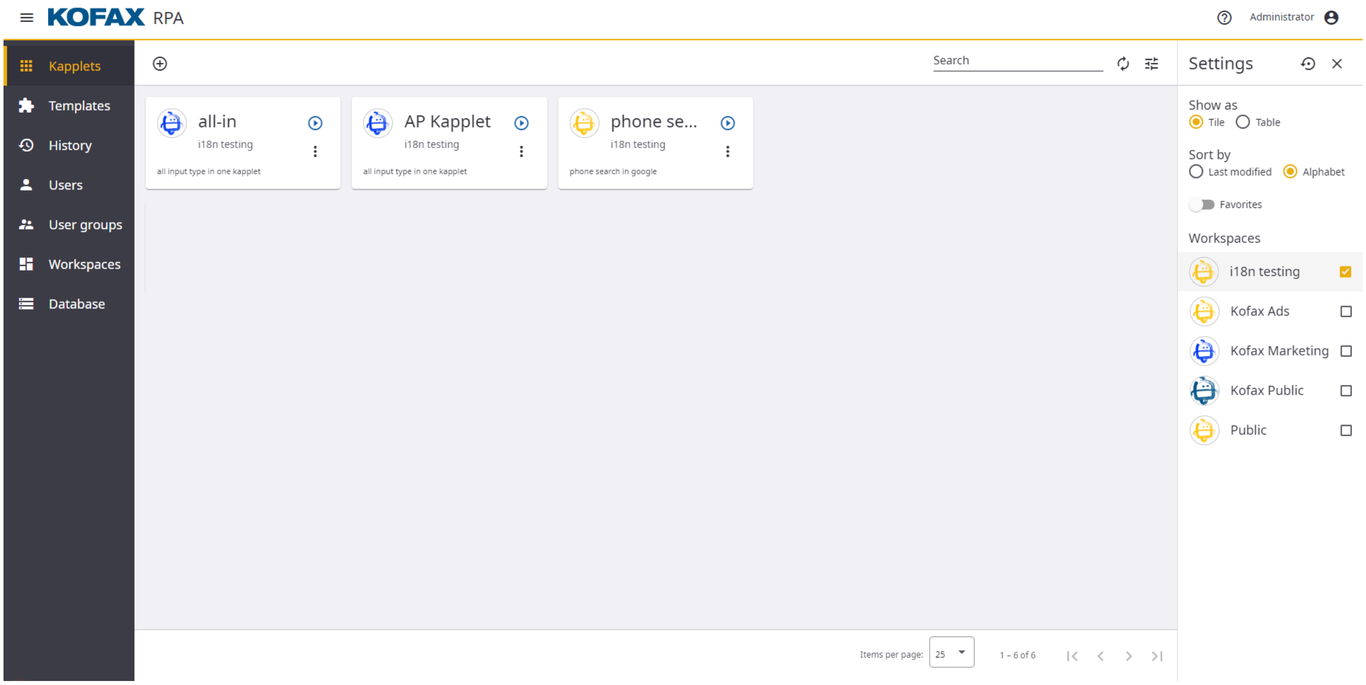
Kapplets User Interface

This topic introduces you to the Kofax RPA Kapplets user interface and describes its elements.

To view the Kofax RPA Kapplets main window, you must have a valid, activated license. See the *Kofax RPA Installation Guide* for details on licensing.

Main User Interface Elements

The Kofax RPA Kapplets has a web-based user interface. It gives you the ability to work with Kapplets not only from the computer, but also from a mobile device such as a smartphone or a tablet.



all-in
i18n testing
all input type in one kapplet

AP Kapplet
i18n testing
all input type in one kapplet

phone se...
i18n testing
phone search in google

Show as
● Tile ○ Table

Sort by
○ Last modified ● Alphabet

Workspaces
i18n testing ☑
Kofax Ads □
Kofax Marketing □
Kofax Public □
Public □

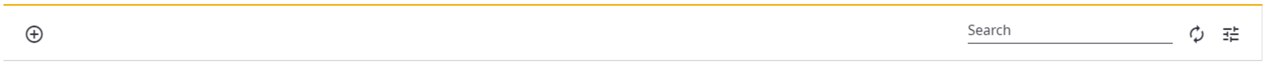
Items per page: 25

The main window consists of the following user interface elements:

- Toolbar
- Side Menu
- User Menu

Toolbar

The toolbar is located at the top of the display area of each tab. Depending on the selected tab, a set of elements on the toolbar may vary.



Use the ⊕ **Add** button to create a new item on the selected tab.

To refresh the items on the tab, click the ⟳ **Reload data** button.

Use the search box to search through the items on the selected tab.

Most tabs of Kofax RPA Kapplets display items of a certain type in a table. When many items are available for display, they are divided into pages. This is managed with the **Items per page** setting at the bottom of the page.

The tables of the displayed items support filtering. Click the ☷ **Settings** pane button to open the list of filtering options. Depending on the selected tab, the set of filtering options may vary. For example, on the **Kapplets** and on the **Templates** tabs, it also shows a list of available workspaces to filter Kapplets and Templates by.

Side Menu

The side menu is located on the left of the Kofax RPA Kapplets main window.

To either minimize the side menu to icons or to restore it down, click the ☰ side menu button in the upper left corner of the window.

Use the side menu tabs to navigate through the Kofax RPA Kapplets.

Kapplets

This tab is available for all user groups.

Use this tab to create, edit, and run Kapplets. Note that before creating a Kapplet, you need to create a Template first.

Templates

This tab is available only for developers, administrators, and global administrators and developers.

Use this tab to create and edit Templates. Kapplets are fully based on Templates.

History

This tab is available for all user groups.

Use this tab to view the execution information about Kapplets. Click a Kapplet name in the table to open the execution information.

Filter the execution information by the execution state on the **Settings** pane.

Users

This tab is available only for global administrators.

Use this tab to manage Kofax RPA Kapplets users:

- Edit the information about users
- Assign global privileges to users
- Change user group assignment
- Enable, disable, or remove users

User groups

This tab is available only for global administrators.

Use this tab to create and manage user groups:

- Create new user groups
- Add new users to user groups
- Assign global privileges to user groups and local privileges in the workspaces

Workspaces

This tab is available only for administrators and global administrators.

- Use this tab to create, manage, and remove workspaces.
- Assign local privileges to user groups in a workspace

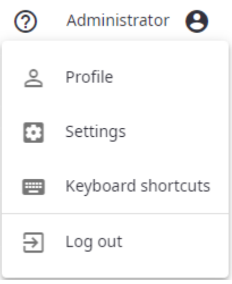
Database

This tab is available only for global administrators.

Use this tab to import the Kofax RPA Management Console backup files. Click **Select a file** to select and upload a file.

User Menu

The user menu button is located in the upper right corner of the main window. Click the button to open a drop-down menu.



Profile

Click **Profile** to see general information about the user.

Settings

Click **Settings** to open the **Settings** tab. Use this tab to change the user interface language (by default, the user interface language is set to English). Note that local user interface language settings have a higher priority than those assigned by the administrator.

To reset all settings to default ones, click **Reset settings** on the **Settings** tab.

Keyboard shortcuts

Click **Keyboard shortcuts** to see the keyboard shortcuts that can be used in the Kofax RPA Kapplets.

Kapplets User Roles

When a Kapplet is ready for use, a Kapplet user or a group of users can obtain access privileges to the Kapplet through a specialized workspace. A group of users can work only with a set of Kapplets defined by the administrator. Kapplet users keep a list of the installed Kapplets in their workspace.

Kapplets are built and maintained by users depending on their roles and privileges, which are assigned and changed by the global administrator on the Kapplets side.

- **User**: Runs Kapplets.
- **Developer**: Creates Kapplets and Templates, runs Kapplets.
- **Administrator**: Creates Kapplets and Templates; runs Kapplets and assigns privileges.
- **Global Administrator**: Creates Kapplets, Templates, and Workspaces; runs Kapplets and assigns privileges.

Equivalents of the Management Console roles in Kapplets

Important This transfer occurs only when a Management Console backup is imported to the Kapplets.

Management Console group role	Kapplets group privilege
Kapplet User	User

Management Console group role	Kapplets group privilege
Developer	Developer
Kapplet Administrator	Developer
Project Administrator	Administrator
RPA Administrator	Global Administrator

Note All other Management Console roles do not correlate with the Kofax RPA Kapplets privileges.

These privileges can be of two types: local privileges and global privileges:

- Global privileges are managed at the system-wide level and can be assigned to a group of users, where each user inherits the privilege, or to a certain user.
- Local privileges are assigned only to the user groups of a certain workspace, users inherit these privileges from the user group .

Transfer of Kapplets roles

All Management Console roles related to Kapplets that were assigned to users in a previous product release can be transferred to the new Kapplets by importing a Management Console backup to the new Kapplets. The roles from the backup are converted into their equivalents in the new Kapplets.

Note If you have a new instance of Management Console, you need to first import the backup to theManagement Console and then to the new Kapplets.

Also, as Kapplets is a standalone application, if a user is assigned a role related to Kapplets in the Management Console, it is not automatically transferred when the user logs in to the Kapplets. The new role or any role change must be explicitly confirmed by the global administrator, that is, assigned to the user on the Kofax RPA side.

Restore Kapplets from Management Console backup

Apart from creating new Kapplets, Classic Kapplets can also be restored with the Management Console backup with the option **Merge**.

The users are also moved to Kofax RPA Kapplets within their user groups. These user groups and users get no privileges, until these privileges are assigned by the global administrator.

Objects from Management Console backup are moved to the Kofax RPA Kapplets the following way:

Management Console backup objects	Kofax RPA Kapplets objects
Project	Workspace
Master Kapplet	Template
Installed Kapplet	Kapplet marked as "favorite"

Build and Maintain Kapplets

This section provides information about Creating Workspaces, Creating Templates, Creating Kapplets, and Running Kapplets.

Create and Edit Workspaces

A Workspace is a logical group that combines Kapplets and Templates in one place by meaning and by department needs.

Create Workspaces

Workspaces can be created by administrators and global administrators.

To create a workspace, navigate to the **Workspaces** tab on the side menu and perform the following steps:

1. Click the ⊕ **Create new workspace** button on the toolbar.
2. In the **Create new workspace** window, configure the following.
   - **Name**: Type the workspace name.
   - **Description**: Type a short description to add more information about the workspace.
   - **Icon**: Add an icon to make the workspace unique and easy to distinguish. Either select an image from the **Gallery** or upload a new one. Tag the uploaded image to make it easy to find among other images.

     Important The image **Gallery** supports only `.png` and `.jpeg` files.
3. Click **Save** in the lower right corner of the window for the changes to take effect.

A user can see the list of the available workspaces on the **Settings** pane of the visible tabs. For example, users with local privileges can see their list of workspaces on the **Settings** pane of the **Kapplets** tab.

Note that Templates and Kapplets can be added to a workspace only on the **Templates** and **Kapplets** tabs.

Edit Workspaces

To edit a workspace, navigate to the **Workspaces** tab on the side menu and click on the workspace name in the table.

Create and Edit Templates

Every Kapplet is based on a template, which functions as a main base object for the Kapplet. A Kapplet can be considered as a "copy" of its template. When a Template is changed, the Kapplet remains unchanged.

Create Templates

Templates can be created by developers, administrators, and global administrators and developers.

To create a template, navigate to the **Templates** tab on the side menu and perform the following:

1. Click the ⊕ **Add** button on the toolbar to open the **General** step window.
2. On the **General** step window, configure the following information.
   - **Name**: Type the Template name.
   - **Workspace**: Select the workspace from the drop-down list of available workspaces.
   - **Description**: Type a short description to add more information about the template.
   - **Icon**: Add an icon to make the template unique and easy to distinguish. Either select an image from the **Gallery** or upload a new one. Tag the uploaded image to make it easy to find among the other images.

     Important The image **Gallery** supports only `.png` and `.jpeg` files.
3. Click **Next** in the lower right corner of the window to proceed to the next step.
4. On the **Robots** step window, add robots to the template.

   The selected robots are shown on the **Selected robots** pane on the left of the window. To open the pane, click the **Selected robots** pane button on the toolbar. The number of selected robots is reflected on the button. On the **Selected robots** pane, use the **Run sequentially** button to define the robots execution order.

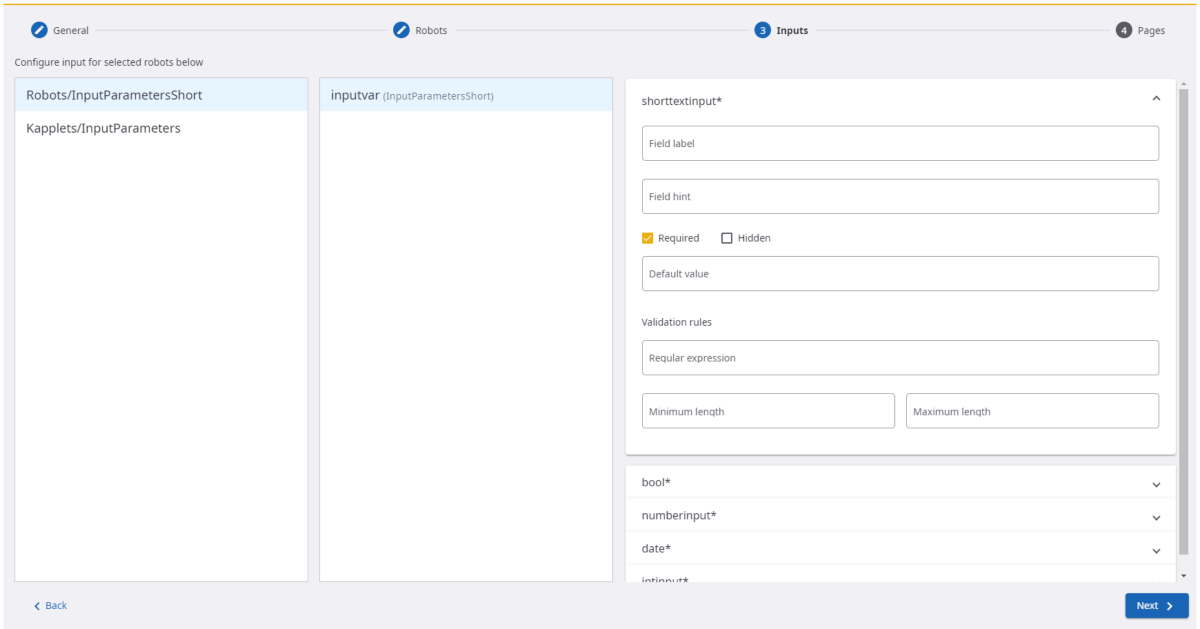
   You can also filter robots by a Management Console project on the **Settings** pane.

   Click **Next** to proceed to the next step.

5. On the **Input** step window, configure input parameters for the selected robots.

   > **Note** The **Input** step is added to the procedure if one of the selected robots contains input parameters.

   - Click a robot to see its variables (variable types).
   - On the window with robot variables, click a variable to configure its attributes.
   - All of the attributes are listed on the window next to the variables window.



☑ Required ☐ Hidden

   On the attribute pane, fill in the **Field label** and **Field hint** for display on the **Start Page**.

   Use the **Required** and **Hidden** check boxes to configure how the fields are reflected on the **Start Page**.

   Configure attribute individual parameters. The set of parameters depends on each individual attribute and may vary.

   Click **Next** to proceed to the next step.

6. The **Pages** step window includes the following configuration.
   - On the **Start Page**, edit the page title and type a description for the page in the **Page text** field. Drag an item on the **Fields order** list to change its order.
   - On the **Status page**, edit the page title.
   - To see the execution result in a table, add a **Table page** by clicking the corresponding icon on the toolbar. On the **Table page**, edit the page title and description. Select a robot on the drop-down list

   to configure its parameters. All of the parameters are listed below the robot. Click a parameter to define its fields for the table.

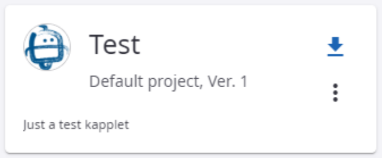
   - To see the execution result in a chart, add a **Chart page** by clicking the corresponding icon on the toolbar. On the **Chart page**, edit the page title and description. Select a robot on the drop-down list to configure the parameters to display on the chart. Select the following parameters:
     - Chart type
     - Color scheme
     - Label axis
     - Data axis

     Additionally, configure other chart parameters.

     Preview the chart image by clicking the ◉ **Preview** button.

7. Click **Save** to save the Template.

A template entity looks similar to the following.



Test

Default project, Ver. 1

Just a test kapplet

Edit Templates

To edit or remove a template, click the ⋮ **Open context menu** button.

Create and Edit Kapplets

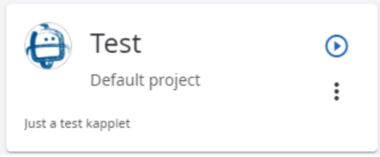
Create Kapplets

Kapplets can be created by developers, administrators, and global administrators.

To create a Kapplet from a Template, perform the following steps:

1. Click the ⬇ **Create new Kapplet** button on the Template.
2. The fields on the **Create new Kapplet** window inherit their values from the Template. Either edit them or proceed with saving the Kapplet. To save the Kapplet, click **Save** in the lower right corner of the window.

A Kapplet entity looks similar to the following.

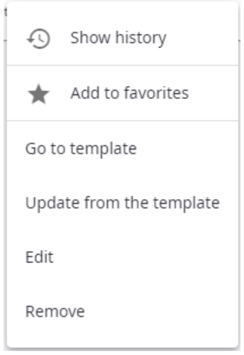


Test

Default project

Just a test kapplet

To perform more actions on the Kapplet, click the ⋮ **Open context menu** button.



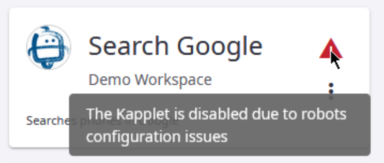
Alternatively, when one or more Templates already exist, you can create a Kapplet on the **Kapplets** tab.

To create a new Kapplet on the **Kapplets** tab, perform the following steps:

1. Click the ⊕ **Add** button on the toolbar to open the **Create new Kapplet** window.
2. In the **Create new Kapplet** window, configure the following information.
   - **Name**: Type the Kapplet name.
   - **Template name**: Select the Template from the drop-down list of available Templates.
   - **Workspace**: A workspace of the selected Template.
   - **Description**: Type a short description to add more information about the Kapplet.
   - **Icon**: Add an icon to make the Kapplet unique and easy to distinguish. Either select an image from the **Gallery** or upload a new one. Tag the uploaded image to make it easy to find among other images.

     Important The image **Gallery** supports only `.png` and `.jpeg` files.
3. Click **Save** in the lower right corner of the window for the changes to take effect.

Important A Kapplet becomes disabled if a robot in this Kapplet is removed or changed. The following notification is thrown:



The Kapplet is disabled due to robots configuration issues

To enable the Kapplet, update the Template with a changed or removed robot and then update the Kapplet.

Edit Kapplets

To edit or remove a Kapplet, click the ⋮ **Open context menu** button.

Run Kapplets

To run a Kapplet, perform the following steps:

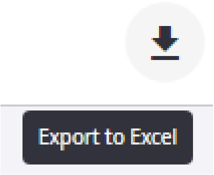
- Click the ⊙ **Run Kapplet** button on the Kapplet to open the **Start Page**. On the **Start Page**, configure the input parameters (if present) and click **Run** in the lower right corner of the window.
- After the Kapplet is executed, the **History** tab opens. See the execution information on the **History** tab. Here you can also see the information on the **Status page**, **Table page**, and **Chart page**.

Export Kapplets to Excel

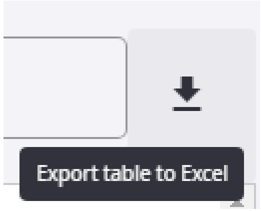
1. To export the result of a Kapplet execution to an Excel file, on the **History** tab, from the list of executions, click a successfully executed Kapplet.

   You can download the execution result only if it is successful and has a table data to be exported.

   - To export the entire execution result, click the download button on the right.

     If you have multiple tables in your Kapplet execution result, they are shown as separate tabs in the resulting Excel file.
   - If you have multiple tables and you want to download a particular table, on the respective **Table page** tab, click the lower download button.

2. Observe the result in your Excel file.

Install Kapplets

To install the Kofax RPA Kapplets, perform the following steps. For details, see the *Kofax RPA Installation Guide*.

1. Deploy the Kofax RPA Kapplets on a web server.
2. For the global administrators, Kofax RPA Kapplets are typically accessed directly from the web browser. Kapplets can be accessed directly at `http://<ip_address>:<port>/kapplets`. Use the Management Console credentials to authorize in the Kofax RPA Kapplets.

See the Kofax RPA Kapplets limitations in the Kofax RPA Limitations section.

Chapter 10

Reference

Contents:

Web Automation

Web automation is a part of Kofax RPA application for creating robots that work with web sites. This section provides additional information about web automation features and parameters.

For an in-depth description of the Web automation, see Web Automation and Web Automation & Desktop Automation.

Protocols

A Protocol defines a mechanism for contacting a RoboServer. Currently Kofax RPA comes with three different protocols, each with its own set of advantages and disadvantages.

Protocol	Description
Socket	Contacts a RoboServer using a TCP socket. This is a simple, low-level protocol, using XML or Java binary representation depending on the platform. See description of properties below.
Random Distribution	Given a list of protocols, each time the client makes a request, one of the protocols will be chosen based on whether it is currently marked as available or not. This provides simple fail-over if at least one of the RoboServers specified in the list is available. While the random distribution protocol does not explicitly provide load balancing, it can be used for that purpose. See description of properties below.

Properties

The protocols are configured using the following properties:

Socket

Host Name

The name of the host machine a RoboServer is located on.

Port Number

The port number a RoboServer is listening on. The default port number is 50000.

Random Distribution Protocol

Retry If Connection Lost

Enable this option to support transparent fail-over. If the connection to a RoboServer is lost while handling a request, the protocol can re-submit the request to another RoboServer from the list. In order for this to work correctly, the robot in question must be idempotent, meaning that repeated invocations of the robot have the same effect as one. Typically this is the case with robots that do not cause permanent changes in the sites they access.

Protocols

The list of protocols to distribute requests to. The random distribution policy select a protocol at random for handling each individual request.

Depending on what plugins are installed into Kofax RPA, other protocols may be available.

Robot Libraries

A robot library is a collection of Kofax RPA robots and types. When a RoboServer receives a request to execute a robot it will search a robot library for the robot and associated types.

Note Robot libraries are available only when running robots through the RoboServer API.

Robot Library	Description
Default Robot Library	A RoboServer uses its current project robot library for looking up robots and types. This is very convenient during development as every time a fix is made to a robot, the change is immediately available.
Robot Library File at URL	A RoboServer loads the library from the URL once and then caches the library for a period of time. The cache timeout can be controlled by the cache-timeout attribute in the deployment descriptor. See description of properties below.
Robot Library Embedded in Request	This option embeds the robot library in all requests sent to a RoboServer. The RoboServer will then use this library to extract the robot and types needed to fulfill the request.
Robot Library Folder at URL	Instructs a RoboServer to look up Robots relative to a specified URL. See description of properties below.

Properties

The libraries are configured using the following properties:

Robot Library File at URL

URL

Location of the packaged robot library.

Allow Caching

Enable this option to allow a RoboServer to cache the robot library. If a RoboServer is not allowed to cache the content of the URL, it retrieves the contents of the URL each time it receives a request. This is very useful if you have a big robot library as it reduces the time it takes to start the robot.

Cache Timeout

The number of seconds a RoboServer is allowed to cache the robot library. If the Allow Caching property is not enabled, this setting has no effect.

Robot Library Folder at URL

URL

The URL to look up robots relative to.

Plugin Simulation from JSON Variable

You can construct your own plugins using a JSON variable. This is an example of how the JSON structure can look like:

```
[
  {
    "name" : "Shockwave Flash",
    "description" : "Shockwave Flash 18.0 r0",
    "filename" : "NPSWF32_18_0_0_232.dll",
    "mimeTypes" : [
                    {
                      "type" : "application/x-shockwave-flash",
                      "description" : "Adobe Flash movie",
                     "suffixes" : "swf"
                    },
                    {
                      "type" : "application/futuresplash",
                      "description" : "FutureSplash movie",
                      "suffixes" : "spl"
                    }
                  ]
  },
  {
    "name" : "Silverlight Plug-In",
    "description" : "Silverlight Plug-In 5.1.40416.0",
    "filename" : "npctrl.dll",
    "mimeTypes" : [
                    {
                      "type" : "application/x-silverlight",
                      "description" : "npctrl",
                      "suffixes" : "scr"
                    },
                    {
                      "type" : "application/x-silverlight-2",
                      "description" : "",
                      "suffixes" : ""
                    }
                  ]
  }
]
```

You can use the following code to generate a JSON variable to be used in the plugin simulation. Just save the code to an HTML file and open it in a browser. The generated text can then be copied directly to a JSON variable on the **Plugins** tab of the **Options** window in Design Studio.

```
<!doctype html>
<html>
<body>
<div id="plugins"></div>
</body>
<script>
var plugins=[];
for(var n=0; n<navigator.plugins.length; n++) {
          plugins.push({});
          plugins[n].name=navigator.plugins[n].name;
          plugins[n].description=navigator.plugins[n].description;
          plugins[n].filename=navigator.plugins[n].filename;
          plugins[n].mimeTypes=[];
          for(var m=0; m<navigator.plugins[n].length; m++) {
                   plugins[n].mimeTypes.push({});
                   plugins[n].mimeTypes[m].type=navigator.plugins[n][m].type;
                    plugins[n].mimeTypes[m].description=navigator.plugins[n]
[m].description;
                    plugins[n].mimeTypes[m].suffixes=navigator.plugins[n][m].suffixes;
          }
}
var json = document.getElementById("plugins");
json.innerHTML= JSON.stringify(plugins);
</script>
</html>
```

Named Tags, Ranges, and JSON

Named tags, ranges, and JSON are markers that can be used for finding other tags, ranges, and some JSON text respectively. Steps use Finders to find the elements they work on (either HTML/XML tags, Excel ranges, or named JSON depending on the kind of content the step works with). A finder may be based on what has been found by previous steps by referring to named tags, ranges, or JSON.

All named tags/ranges/JSON belong to a window. Each window can have any number of named tags/ranges, but only of the appropriate type: Named tags for windows with HTML/XML content, named ranges for windows with spreadsheet content, and named JSON for windows with JSON.

Named tags/ranges are set by many steps. For example, loop steps typically use a named tag/range as a marker for the current iteration of the loop. You can also use the Set Named Tag, Set Named JSON, or Set Named Range actions to explicitly name a tag, range, or JSON. This can be useful when you want to simplify the finders in subsequent steps.

In Design Studio, the named tags or ranges in a window are shown using blue boxes. When you right-click in the current window to perform an action on a tag or range, the Tag or Range Finders of the new step will automatically be configured to search using the named tags or ranges of the window whenever possible.

Tag, Range, and JSON Finders

A Finder is used to find a tag on an HTML/XML page, a range of cells in a spreadsheet document, or an element in a JSON structure. Finders are used in steps, where they identify what part of the page the step should work on. The list of Finders of the current step is located in the "Finders" tab in the Step View.

Finders look very different depending on the kind of page they work on, and the finders for each kind are described separately. See Tag Finders for details on the kind of finders used with HTML/XML pages, Range Finders for details on the kind of finders used with spreadsheet content, and JSON Finders for details on JSON Finders.

Tag Finders

A Tag Finder is used to find a tag on an HTML/XML page. Tag Finders are used in steps, where they define how to find the tag(s) to which the step should be applied. The list of Tag Finders of the current step is located in the "Finders" tab in the Step View. Steps that work on spreadsheet content use Range Finders rather than Tag Finders.

Understanding Tag Paths

In understanding how to use Tag Finders, the concept of a *tag path* is important. A tag path is a compact text representation of where some tag is located on a page. Consider this tag path:

This tag path refers to an <a>-tag inside a <div>-tag inside a <body>-tag inside an <html>-tag.

html.body.div.a

A tag path can match more than one tag on the same page. For example, the above tag path will match all of the <a>-tags on this page, except the third one:

```
<html>
  <body>
    <div>
      <a href="url...">Link 1</a>
      <a href="url...">Link 2</a>
    </div>
    <p>
      <a href="url...">Link 3</a>
    </p>
    <div>
      <a href="url...">Link 4</a>
      <a href="url...">Link 5</a>
      <a href="url...">Link 6</a>
    </div>
  </body>
</html>
```

You can use indexes to refer to specific tags among tags of the same type at that level. Consider this tag path:

html.body.div[1].a[0]

This tag path refers to the first <a>-tag in the second <div>-tag in a <body>-tag inside an <html>-tag. So, on the page above, this tag path would only match the "Link 4" <a>-tag. Note that indexes start from 0. If no index is specified for a given tag on a tag path, the path matches any tag of that type at that level, as we saw in the first tag path above. If the index is negative, the matching tags are counted backwards, i.e. starting with the last matching tag which corresponds to index -1. Consider this tag path:

html.body.div[-1].a[-2]

This tag path refers to the second-to-last <a>-tag in the last <div>-tag in a <body>-tag inside an <html>-tag. So, on the page above, this tag path would only match the "Link 5" <a>-tag.

You can use an asterisk ('*') to mean any number of tags of any type. For example, the tag path

html.*.p|div|td.a

This tag path refers to an <a> tag inside a <p>-, <div>-, or <td>-tag located anywhere inside an <html> tag.

In a tag path, text on a page is referred to just as any other tag, using the keyword "text". Although text is not technically a tag, it is treated and viewed as such in a tag path. For example, consider this HTML:

```
<html>
  <body>
    <a href="url...">Link 1</a>
    <a href="url...">Link 2</a>
  </body>
</html>
```

The tag path "html.body.a[1].text" would refer to the text "Link 2".

Tag Finder Properties

A Tag Finder can be configured using the following properties.

Find Where

In this property, you can specify where to find the tag relative to a named tag. The default value is "Anywhere in Page", meaning that named tags are not used to find the tag.

Tag Path

In this property, you can specify the tag path as described in the previous section. The tag path can be specified in several ways using the Value Selector.

Attribute Name

In this property, you can specify that the tag must have a specific attribute, for example "align".

Attribute Value

In this property, you can specify that the tag must have an attribute with a specific value. If the Attribute Name property is set, the attribute value is bound to that specific attribute name.

These values are case-sensitive.

- "Equals Text" specifies that the attribute value must match a specified text. Note that the text must match the entire attribute value.
- "Contains Text" specifies that the attribute value must contain a specified text.
- "Starts With Text" specifies that the attribute value must start with a specified text.
- "Ends With Text" specifies that the attribute value must end with a specified text.
- "Matches Pattern" specifies that the attribute value must match a specified pattern. Note that the pattern must match the entire attribute value.
- "Does Not Equal Text" specifies that the attribute value must not be equal to a specified text.
- "Does Not Contain Text" specifies that the attribute value must not contain a specified text.
- "Does Not Start With Text" specifies that the attribute value must not start with a specified text.
- "Does Not End With Text" specifies that the attribute value must not end with a specified text.
- "Does Not Match Pattern" specifies that the attribute value must not match a specified pattern.

Tag Pattern

In this property, you can specify a pattern that the tag must match (including all tags inside it), for example ".***.*Stock Quotes.***.*". Some caution should be observed in using this property, as it can have considerable impact on the performance of you robot. This is because the "Tag Pattern" may be applied many times throughout a page just to find the one tag that it matches. One way to try and avoid this is to choose "Text Only" for the "Match Against" property.

Match Against

In this property, you can specify that the "Tag Pattern" should match only the text or the entire HTML of the tag. The default is to match only the text because this is normally much faster.

Tag Depth

This property determines which tag to use if matching tags are contained inside each other. The default value is "Any Depth" which accepts all matching tags. If you select "Outermost Tag", only the outermost tags are accepted, and similarly, if you select "Innermost Tag", only the innermost tags are accepted.

Tag Number

This property determines which tag to use if more than one tag matches the tag path and the other criteria. You specify the number of the tag to use, either counting forwards from the first tag or counting backwards from the last tag that matches.

Example

As an example, if you set the Tag Path property to "table", the Attribute Name property to "align", the Attribute Value property to Fixed Text where the text must be "center", and the Tag Pattern property to ".*Business News.*", then the Tag Finder would locate the first <table>-tag that is center aligned and that contains the text "Business News".

Range Finders

A Range Finder is used to find a cell or a range of cells in a spreadsheet. Range Finders are used in steps, where they define how to find the cell(s) to which the step should be applied. The list of Range Finders of the current step is located in the "Finders" tab in the Step View. Steps that work on HTML or XML pages use Tag Finders rather than Range Finders.

You can select between different starting points when you configure a range finder:

Find Specified Range

Specify (in Range) a cell or range of cells using almost ordinary Excel syntax. Keep in mind that (in contrast to Excel) the sheet name must be given.

The range can be specified in several ways using the Value Selector.

Find at Named Range

Specify in (Range) a previously defined named range as the starting point. It may have been defined by for example a Set Named Range step or a Loop in Excel step.

Once a range has been selected as the starting point it may be adjusted in several ways as specified by the Use property, which can make it both smaller or larger. See below for details.

Finally, the Use Upper Left Cell in Merged Cells property determines how to handle merged cells in the spreadsheet. Remember that in Excel, adjacent cells can be "merged" visually to form a larger cell with a single value. Excel considers the larger "merged cell" to have the same cell address as the uppermost and leftmost sub-cell, and the value of the "merged cell" is found at this cell address (only). This is mimicked accurately by Kofax RPA but it is not always convenient when doing automated extraction, especially as part of an iteration. Thus if you enable Use Upper Left Cell in Merged Cells and the range refers to a single sub-cell within a "merged cell", then it is modified to refer to the uppermost and leftmost sub-cell of the "merged cell" to make it easier to get at the contents.

Cell Ranges

When configuring a Range Finder to Find Specified Range you write a piece of text that refers to a cell (or a range of cells). Such references are also displayed (and can be entered) in the Cell Range View at the bottom of the Spreadsheet View. The basic form is known from Excel formulas, but Kofax RPA provides some extensions.

The following examples show the variants:

Sheet1!B3

The core elements and how they are put together: The sheet name ("Sheet1"), a separating exclamation mark, a column name ("B") and a row number ("3"). This example refers to a single cell of the named sheet.

B3

When the cell range reference is entered in the Cell Range View at the bottom of the Spreadsheet View, the sheet name can be omitted if you want to refer to the currently displayed sheet. It will, however, be added to the reference as soon as you press Enter. In a Range Finder you need to enter the sheet name every time, as there is no notion of "currently displayed sheet" during execution of the robot. For this reason the remaining examples will all include the sheet name.

Sheet3!B3:F14

How to refer to a range of cells from a single sheet: After the sheet name, you state the upper left and lower right corners, separated by a colon. (It is not possible to have a range that extends over several sheets.)

Sales!C

All of column "C" of the stated sheet. Open ended ranges like this one are not permitted in Excel, but are very useful in robots that must be able to adapt to different sizes of Excel documents. When the robot works on a specific document, it will automatically limit itself to the area of the document that actually contains data. Open ended ranges will often be used in the Range Finder of a Loop in Excel step.

Sales!C:H

All of the columns C to H of the stated sheet and is an open ended range as explained above.

Suppliers!14

Open ended ranges can also be rows. This example refers to all of row 14 of the stated sheet.

Suppliers!14:29

A fixed range that refers to all of rows 14 up to and including 29 of the stated sheet.

'Total Sales'!B3

If the sheet name contains white space or certain special characters, it must be enclosed in single quotes. If the sheet name contains a single quote, it must be entered as two single quote characters. The rules are just the same as when sheet names are included in cell references in Excel formulas.

!B3

The Excel "document properties" (for example, the name of the author and the creation date of the document) are made available in Kofax RPA in the form of a special sheet whose name is empty. Thus this example refers to the value of one of the document properties (the name of the property will be available in the neighboring cell "!A3"). This is an extension over Excel.

JSON Finders

A JSON Finder is used to find necessary data in a JSON text. The list of Finders of the current step is located in the Step View, Finders tab.

For more information about JSON and its terminology see Working with JSON.

JSON Finder Properties

A JSON Finder can be configured using the following properties.

Find Where
In this property, you can specify where to find a JSON element. The default value is "Anywhere in JSON", meaning that named JSONs are not used in a search.

In this named JSON
This property is used when you select **In Named JSON** in the **Find Where** list. In this property, you can specify whether to search in the selected Named JSON or you can specify a name of the Named JSON to use.

Path

In this property, you can specify the path to the JSON element. The tag path can be specified in several ways using the Value Selector.

JSON path expressions always refer to a JSON structure in the same way as XPath expression are used in combination with an XML document. JSON path expressions are very similar to the JavaScript and use the dot-notation, for example `personnel.person[0].name`. `@top:` element is required and tells the finder to search from the top of the JSON.

Examples

JSON Path

The following is a simple JSON structure and a table with path examples and possible results.

```
{
  "personnel" : {
                  "person" : [
                               {
                                 "ID" : 0,
                                 "name" : "Bob",
                                 "age" : 26,
                                 "isMale" : true
                               },
                               {
                                 "ID" : 1,
                                 "name" : "Ted",
                                 "age" : 25,
                                 "isMale" : true
                               },
                               {
                                 "ID" : 2,
                                 "name" : "Jill",
                                 "age" : 47,
                                 "exam" : "553213-3",
                                 "isMale" : true
                               },
                               {
                                 "ID" : 3,
                                 "name" : "Rick",
                                 "age" : 50,
                                 "exam" : "553225-3",
                                 "isMale" : true
                               }
                             ]
                }
}
```

XPath	JSON Path	Result
/personnel/person[2]/name	@top:.personnel.person[1].name	Ted
/personnel	@top:.personnel	Extracts all information from "personnel"

If you want to extract a set of information from a JSON element, you can create an XML page from JSON and extract necessary information using a text expression. For example, if you create an XML page from the JSON above, select `item[1]` in the XML, and run an expression like `".*<name>"+TheInput+"</name>.*"`, as a result you should get something similar to `1Ted25true`.

Finding a Named JSON

In the following example Named JSON is a part of a JSON text that can be used in a JSON Finder to find "a":

In the following JSON text:

{ "a" : [{ "b" : [1,2,3] }], "c" :42 }

we can have the Named JSON mark

"b" : [1,2,3]

and thus we can have a JSON Finder perform a search with the following properties:

Find Where: In Named JSON

In this Named JSON: 1

Path: [1]

This finder will then find the number 2 in the list.

Patterns

A pattern is a way of describing a text. For example, the text "32" can be described as a text containing two digits. However, other texts also contain two digits, e.g. "12" and "00". We say that these texts match the pattern. (Design Studio patterns follow the Perl5 syntax.)

A pattern is composed of normal characters and special symbols. Each special symbol carries its own special meaning. For example, the special symbol "." (dot) means any single character and matches all single characters, e.g. "a", "b", "1", "2", ...

Special Symbols

You can use the following special symbols within a pattern.

Special Symbol	Description
.	any single character, e.g. "a", "1", "/", "?", "." etc.
\d	Any decimal digit, e.g. "0", "1", ..., "9".
\D	Any non-digit, e.g. same as "." excluding "0", "1", ..., "9".
\s	Any whitespace character, e.g. " ", tab, and return
\S	Any non-whitespace character, e.g. same as "." excluding " ", tab, and return
\w	Any word (alphanumeric) character, e.g. "a", ..., "z", "A", ..., "Z", "0", ..., "9".
\W	Any non-word (alphanumeric) character, e.g. same as "." excluding "a", ..., "z", "A", ..., "Z", "0", ..., "9".
\n	A line break character.
\r	A carriage return character.
\t	A tab character.
[abc]	Any character in the set a, b or c.

Special Symbol	Description
[^abc]	Any character not in the set a, b or c.
[a-z]	Any character in the range a to z, inclusive.
a|b	Matches whatever the subpattern a would match, or whatever the subpattern b would match.

If you want a special character, such as "." or "\", to act as a normal character, you can escape it by adding a "\" (backslash) in front of it. So, if you wish to match exactly the "." character, instead of any single character, you should write "\.".

You can organize a pattern into subpatterns by the use of parentheses: "(" and ")". The pattern "abc" can be organized as "(a)(bc)". Subpatterns are useful when applying pattern operators.

Example: Simple Example Patterns

Here are some examples of patterns and what they match:

Pattern	Matches
.an	All texts of length three ending with "an", e.g. "can" and "man" but not "mcan".
\d\d\s\d\d	All texts of length five starting with two digits followed by a whitespace and ending with two digits, e.g. "01 23" and "72 13" but not "01 2s"
m\.n\\o	The text "m.n\o"
(good)|(bye)	"good" and "bye" but not "goodbye"

Repeating Operators

These operator symbols will repeat the previous character, symbol, or subpattern.

Special Symbol	Description
{m}	Matches exactly m repetitions of the preceding subpattern.
{m,n}	Matches between m and n repetitions (inclusive) of the preceding subpattern. It will match as many subpatterns as possible.
{m,n}?	Matches between m and n repetitions (inclusive) of the preceding subpattern. It will match as few subpatterns as possible
{m,}	Matches m or more repetitions of the preceding subpattern. It will match as many subpatterns as possible.
{m,}?	Matches m or more repetitions of the preceding subpattern. It will match as few subpatterns as possible.
?	The preceding subpattern, or the empty text. Shorthand for {0,1}
*	Matches any number of repetitions of the preceding subpattern, or the empty text. Shorthand for {0,}. It will match as many subpatterns as possible.
*?	Matches any number of repetitions of the preceding subpattern, or the empty text. Shorthand for {0,}?. It will match as few subpatterns as possible.
+	Matches one or more repetitions of the preceding subpattern. Shorthand for {1,}. It will match as many subpatterns as possible.

Special Symbol	Description
+?	Matches one or more repetitions of the preceding subpattern. Shorthand for {1,}?. It will match as few subpatterns as possible.

These operators repeat the previous character, symbol, or subpattern.

Example: Examples Using Repeating Operators

Here are some examples of patterns that use repeating operators, and what they match.

Pattern	Matches
.*	Any text, e.g. "hello", "1213" and "" (the empty text)
(abc)*	Matches any number of repetitions of the text "abc", e.g. "", "abc", "abcabc", and "abcabcabc", but not "abca"
(.*)(.*)	Will match "abc" - the first subpattern will match "abc" and the second subpattern will match "" (the empty text)
(.*?)(.*)	Will match "abc" - the first subpattern will match "" (the empty text) and the second subpattern will match "abc"
(.+?)(.*)	Will match "abc" - the first subpattern will match "a" and the second subpattern will match "bc"
\w*\d	Will match "abc1abc1" - \w* matches "abc1abc" and \d matches "1"
\w*?\d	Will match "abc1" but not "abc1abc1" - because the "\w*?" will only match "abc" and the rest cannot be matched by \d
(\d\d){1,2}	Matches either two or four digits, e.g. "12" and "67", but not "123".
(good)?bye	"goodbye" and "bye".

More About Grouping

We have seen that "(" and ")" can be used for grouping subpatterns into a new subpattern. But this actually serves another purpose: you can use these groups in expressions. To make a grouping that cannot be used in expressions you can use "(?:".

You can refer to other groups in your pattern using *n*, where *n* is the group number.

Example: Grouping Examples

Here are some examples of patterns that use grouping, and what they match.

Pattern	Matches
a(bc)	Matches "abc=abc", but does not match "abc=ab"
a(?:bc)	Matches "abc", but you can not refer to "bc" in your expressions.
(.*)=\1	You can refer to a group as $1 in your expressions.

POST Requests as URLs

This information is applicable only to Web Automation robots when using the Classic engine browser.

A standard URL can represent only a GET request, but not an HTTP POST request. To represent a POST request as a URL, Kofax RPA uses its own special URL format with the following syntax:

```
post://<url excluding the http protocol>??<encoded POST parameters>
```

or, for the HTTPS protocol:

```
posts://<url excluding the https protocol>??<encoded POST parameters>
```

or, for a relative URL without a protocol:

```
postx://<url without protocol>??<encoded POST parameters>
```

Example

A POST request to the URL

http://www.abc.com

with the encoded POST parameters

`x=y&v=z`

is represented as the following URL:

`post://www.abc.com??x=y&v=z`

Library Protocol

You can use the Kofax RPA non-standard *library* protocol to refer to a file that a Web Automation robot belongs to in the robot library.

For example, if the file MyPage.html is located in the folder MyFolder in the robot library folder, you can refer to it using this URL:

library:/MyFolder/MyPage.html

This works whether the robot library is represented as a folder or is packed into a robot library file.

Value Selector

Use the Value Selector to specify a value in different ways, depending on your need.

Use the drop-down menu on the right side of the Value Selector to select one of the following ways to specify the value (not all of these ways are available everywhere):

Value

Enter or select a fixed value.

Variable

Select the value of a variable.

Expression

Enter an expression.

Converters

Select a list of data converters, whose output is used as the value. (The input text to the data converters is empty.)

Supported Features in Excel

In this topic, you can find both supported and unsupported features in Excel for Web Automation robots.

Formula Support

Search the https://poi.apache.org for details on supported formula.

Features

Supported

- References: single cell & area, 2D & 3D, relative & absolute
- Literals: Number, text, boolean, error and array
- Operators: arithmetic and logical, some region operators
- Built-in functions: over 350 recognized, 280 evaluatable
- Add-in functions: 3 from Analysis Toolpack
- Font color using the format string such as `[red]`
- Conditional font color, such as in the example where negative numbers are red: `#.##0;[Red]-#.##0`
- Date formatting

Unsupported

- Manipulating array/table formulas (In Excel, formulas that look like "{=...}" as opposed to "=...")
- Region operators: union, intersection
- Parsing of previously uncalled add-in functions
- Preservation of whitespace in formulas (when POI manipulates them)
- Font changes, for example bold, size, etc.
- Background color in cell
- External file references from formulas
- Excluding hidden values
- Converting strings to a date with the FLOOR, HOUR, and MINUTE functions. Only decimal values are supported.

Function in POI

Supported Functions

ABS, ACOS, ACOSH, ADDRESS, AND, AREAS, ASIN, ASINH, ATAN, ATAN2, ATANH, AVEDEV, AVERAGE, BIN2DEC, CEILING, CHAR, CHOOSE, CLEAN, CODE, COLUMN, COLUMNS, COMBIN, COMPLEX, CONCATENATE, COS, COSH, COUNT, COUNTA, COUNTBLANK, COUNTIF, COUNTIFS, DATE, DAY, DAYS360, DEC2BIN, DEC2HEX, DEGREES, DELTA, DEVSQ, DGET, DMAX, DMIN, DOLLAR, DSUM, EOMONTH, EDATE, ERROR.TYPE, EVEN, EXACT, EXP, FACT, FACTDOUBLE, FALSE, FIND, FIXED, FLOOR, FREQUENCY, FV, HEX2DEC, HLOOKUP, HOUR, HYPERLINK, IF, IFERROR, IMAGINARY, IMREAL, INDEX, INDIRECT, INT, INTERCEPT, IPMT, IRR, ISBLANK, ISERR, ISERROR, ISEVEN, ISLOGICAL, ISNA, ISNONTEXT, ISNUMBER, ISODD, ISREF, ISTEXT, LARGE, LEFT, LEN, LN, LOG, LOG10, LOOKUP, LOWER, MATCH, MAX, MAXA, MEDIAN, MDETERM, MID, MIN, MINA, MINUTE, MINVERSE, MIRR, MOD, MODE, MONTH, MROUND, MMULT, NA, NETWORKDAYS, NOT, NOW, NPER, NPV, OCT2DEC, ODD, OFFSET, OR, PERCENTILE, PI, PMT, POISSON, POWER, PPMT, PRODUCT, PROPER, PV, QUOTIENT, RADIANS, RAND, RANDBETWEEN, RANK, RATE, REPLACE, REPT, RIGHT, ROMAN, ROUND, ROUNDDOWN, ROUNDUP, ROW, ROWS, SEARCH, SECOND, SIGN, SIN, SINH, SLOPE, SMALL, SQRT, STDEV, SUBSTITUTE, SUBTOTAL, SUM, SUMIF, SUMIFS, SUMPRODUCT, SUMSQ, SUMX2MY2, SUMX2PY2, SUMXMY2, T, TAN, TANH, TEXT, TIME, TODAY, TRANSPOSE, TREND, TRIM, TRUE, TRUNC, UPPER, VALUE, VAR, VARP, VLOOKUP, WEEKDAY, WEEKNUM, WORKDAY, YEAR, YEARFRAC

Unsupported Functions

ACCRINT, ACCRINTM, AMORDEGRC, AMORLINC, ASC, AVERAGEA, AVERAGEIF, AVERAGEIFS, BAHTTEXT, BESSELI, BESSELJ, BESSELK, BESSELY, BETADIST, BETAINV, BIN2HEX, BIN2OCT, BINOMDIST, CELL, CHIDIST, CHIINV, CHITEST, CONFIDENCE, CONVERT, CORREL, COUPDAYBS, COUPDAYS, COUPDAYSNC, COUPNCD, COUPNUM, COUPPCD, COVAR, CRITBINOM, CUBEKPIMEMBER, CUBEMEMBER, CUBEMEMBERPROPERTY, CUBERANKEDMEMBER, CUBESET, CUBESETCOUNT, CUBEVALUE, CUMIPMT, CUMPRINC, DATEDIF, DATESTRING, DATEVALUE, DAVERAGE, DB, DBCS, DCOUNT, DCOUNTA, DDB, DEC2OCT, DISC, DOLLARDE, DOLLARFR, DPRODUCT, DSTDEV, DSTDEVP, DURATION, DVAR, DVARP, EFFECT, ERF, ERFC, EXPONDIST, FDIST, FINDB, FINV, FISHER, FISHERINV, FORECAST, FORMULATEXT, FTEST, FVSCHEDULE, GAMMADIST, GAMMAINV, GAMMALN, GCD, GEOMEAN, GESTEP, GETPIVOTDATA, GROWTH, HARMEAN, HEX2BIN, HEX2OCT, HYPGEOMDIST, IFNA, IMABS, IMARGUMENT, IMCONJUGATE, IMCOS, IMDIV, IMEXP, IMLN, IMLOG10, IMLOG2, IMPOWER, IMPRODUCT, IMSIN, IMSQRT, IMSUB, IMSUM, INFO, INTRATE, ISPMT, JIS, KURT, LCM, LEFTB, LENB, LINEST, LOGEST, LOGINV, LOGNORMDIST, MDURATION, MIDB, MMULT, MULTINOMIAL, N, NEGBINOMDIST, NOMINAL, NORMDIST, NORMINV, NORMSDIST, NORMSINV, NUMBERSTRING, OCT2BIN, OCT2HEX, ODDFPRICE, ODDFYIELD, ODDLPRICE, ODDLYIELD, PEARSON, PERCENTRANK, PERMUT, PHONETIC, PRICE, PRICEDISC, PRICEMAT, PROB, QUARTILE, RECEIVED, REPLACEB, RIGHTB, RSQ, RTD, SEARCHB, SEQUENCE, SERIESSUM, SKEW, SLN, SQRTPI, STANDARDIZE, STDEVA, STDEVP, STDEVPA, STEYX, SYD, TBILLEQ, TBILLPRICE, TBILLYIELD, TDIST, TIMEVALUE, TINV, TRIMMEAN, TTEST, TYPE, USDOLLAR, VARA, VARPA, VDB, WEIBULL, XIRR, XNPV, YIELD, YIELDDISC, YIELDMAT, ZTEST

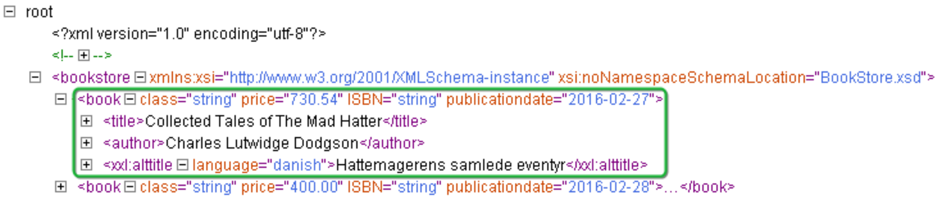
Also, it is not supported to create spilled formulas in Excel in Design Studio. However, Design Studio supports handling of Excel documents with spilled formulas; you can open, edit, and save such documents.

XML Data Mapper

XML Data Mapper allows for convenient mapping of the data records specified by an XML document into variables of suitable structure.

Create a Data Mapping

1. Select the relevant XML element in the source view:



```
⊟ root
    <?xml version="1.0" encoding="utf-8"?>
    <!-- ⊞ -->
  ⊟ <bookstore ⊟ xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="BookStore.xsd">
    ⊟ <book ⊟ class="string" price="730.54" ISBN="string" publicationdate="2016-02-27">
      ⊞ <title>Collected Tales of The Mad Hatter</title>
      ⊞ <author>Charles Lutwidge Dodgson</author>
      ⊞ <xxl:alttitle ⊟ language="danish">Hattemagerens samlede eventyr</xxl:alttitle>
    ⊞ <book ⊟ class="string" price="400.00" ISBN="string" publicationdate="2016-02-28">...</book>
```

For details about issues related to selection, see Selecting Data.

2. Configure the mapping using "Extract with XML Data Mapper" on the context menu of the selected XML element.
3. On the Extract list, select **With XML Data Manager**.

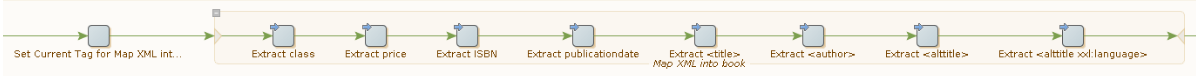
   The XML Data Manager window appears.

   a. Map a source from the available list.
   b. In the Target Variable area, select a target variable and type.
   c. The bottom right area of the window is reserved for the configuration of details for individual entity mappings.

      For more information, see Configuring.

   The step sequence that performs the mapping at runtime is automatically generated.
4. Click **OK**.

   The steps are automatically created.



For more information, see Auto-generating Steps.

Selecting Data

While the selection of data is easy, it is important to know how the selection and mapper interact.

In the Source column, select an XML element to map entities into variable attributes. Use the mapper to map the following entities to variable attributes:

- The attributes of the selected element itself. In the Sources column such attributes are listed as plain text names.
- The contents of sub-elements of the selected element that do not themselves contain sub-elements. In the Source column, such elements appear in an XML-like format where names are enclosed by < and >.
- The attributes of sub-elements of the selected element that do not themselves contain sub-elements. In the Source column, such attributes also appear in an XML-like format where both the enclosing element names and the attribute names are enclosed by < and >.

Note To activate the mapper for a section, at least one mappable element must be associated with the selected element.

Configuring

The configuration step is where most of the user interaction takes place. The configuration step is intended to identify an appropriate target variable attribute for each source entity that you wish to map. You are not required to map all available sources.

1. In the Target Variable section, select **Create a new Variable**.
2. Enter a name for the variable.
3. Select **Create a new Type**.
4. Enter a type name.

   Attribute names matching the source entities are automatically generated and a mapping is suggested.

   - To create a new variable from an existing type, select **Use Existing Type**. The system attempts to map source entities where the type prescribes attribute names that coincide with the names suggested by the sources.
   - You can use an existing variable and associate it with a new type. Attribute names matching the source entities are automatically generated and mapping is suggested.
   - You can use an existing variable and an existing type. The system will try to map sources automatically based on the attributes in the type.

   Note When a new mapping is initiated, the system automatically attempts to make appropriate variable and type choices. Suggestions are made based on the XML element tag name . If a variable has the same name, the variable and type are suggested. If a matching variable exists but a type is named like the tag name, the system suggests creating a new variable using this type.

5. To map Entities, in the Source list, select an entity.
6. In the Target list, select an attribute.

   A line connecting the source and the target shows the connection.

   You can map a single source to more than one target.
7. If the source entity is required for the overall mapping to be sound, select the check box next to the Source entity. When a source entity is marked as required, if that field does not exist during extraction, the robot fails.

   Associations can be dissolved as easily as they are created. To clear an association, hover the line close to the target and click the red cross that appears.
8. If no target attribute is suitable for the source entity is available, select the source element and click add +.

   The Add Attribute window appears.
9. Enter a name and type for the attribute.

   To remove an attribute, select the attribute and click remove −.

   Note This only works with attributes added in the current invocation of the mapper, such as attributes that have not yet been persisted in the corresponding type file.

10. To configure the target attribute, select the attribute.

    Configuration attributes appear.

    You can associate data converters with the mapping, such that data may be brought into acceptable form.

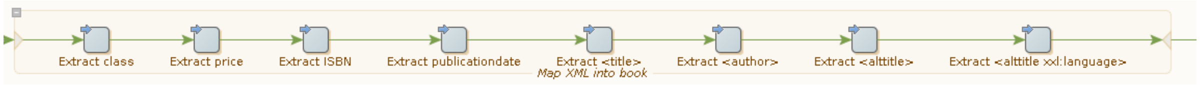
    You can select Trim Spaces if the source entity is an XML attribute.

    - You can change the target attribute name and type.
    - Associate data converters with the mapping such that data is brought into an acceptable form.
    - Select to clear Trim Spaces for XML attributes.

Auto-generating Steps

1. When the configuration of a data mapping is done, click **OK**.

   The system auto-generates the actual steps required to perform the implied extractions and inserts the steps into the robot. The procedure always generates exactly one extract or extract tag attribute step for each association created in the mapper.



2. Anchor the data mapping by a named tag identifying the original XML element you selected.

   If such a named tag does not exist, a Set Named Tag step is generated and inserted into the robot just before the actual data mapping.

   > **Note** Sometimes such a Set Named Tag is not necessary because a suitable named tag is already in place. This often occurs when the mapping is part of a loop.

Edit a Data Mapping

Once a data mapping is configured, you can use the data mapper to open and edit it. Since data mappings are associated with group steps, the ability to reopen the mapper is also connected to group steps.

1. To open a data mapping, right-click a group step and select **Open XML Data Mapper**.

   This will only work if the group step is the current step, that is, green. If it is not green, there may not be XML information to support the mapper.

   Alternatively, you can open the mapper for a group step that is current on the associated step view and click **Open XML Data Mapper**
2. Once open, the mapper will behave exactly as described in Create a Data Mapping.

3. Edit the data mapping steps manually if needed.

Note that only certain changes are compatible with the XML data mapper. Opening a data mapping is subject to a number of conditions, essentially expressing that the group step MUST be a real data mapping.

- The group step may not contain control structure in the form of branches, loops, or similar.
- The group step must contain at least one *extract* or *extract tag attribute* step.
- The extractions must all refer back to the same named tag.
- The extractions must all have target attributes of the same variable.
- For any given attribute of this variable, there may only be one extraction.
- No extraction may use a tag finder where an asterisk '*' occurs.
- All targeted attributes must exist in the underlying type.

URL Blocking

In the Configure URL Blocking Pattern window, you can specify a URL blocking pattern and add this pattern to the robot's list of Blocked URL Patterns. Once you specify the pattern and click OK, the robot is re-executed. To see the list of blocked URLs, go to **File** > **Configure Robot**, click **Configure** on the **Basic** tab, and select **URL Filtering**.

You can use special symbols and edit the pattern in the Pattern Editor. See Patterns for more information.

Web Authentication

Web Automation robots in Kofax RPA can use different authentication over a network. The authentication setting is specified under **Credentials** on the **All Loading** tab of the Default Options window. You can use either Standard or OAuth credentials. See OAuth for more information.

With the Standard Credentials option, Kofax RPA supports Basic, Digest, NTLM, and Negotiate protocols. For Windows systems, Kofax RPA uses the Security Support Provider Interface (SSPI) to provide security services to calling applications. For Unix, Kofax RPA uses the Generic Security Service API (GSS-API) libraries for Negotiate protocol and developed proprietary NTLM support implementation.

Note Your Linux installation must include Generic Security Service API (GSS-API) libraries to use cross-platform authentication. See "Dependencies and Prerequisites" section in the *Kofax RPA Installation Guide* for more information.

In case a user needs to have access to remote network resources, delegation must be set up to access those resources. For more information about setting up authentication and delegation rules, see Microsoft documentation at msdn.microsoft.com and support.microsoft.com.

Kofax RPA automatically detects the type of authentication during the login process and in most situations provides authentication parameters in the required format. For NTLM protocol, SPN (Service Principal Name) string is always as follows: `HTTP/HostName:port`. For Negotiate protocol, SPN may be with or without the port number.

In some cases you may need to explicitly provide authentication parameters for Negotiate protocol. You can do it either in the Authentication Method option on the All Loading tab of the Default Options in Basic Robot Configuration dialog box or using `spn.txt` file.

Specify Negotiate protocol parameters in the Default Options dialog box

1. Open the Default Options dialog box from the Basic tab of the Robot Configuration window.
2. Select Negotiate in the Authentication Method list and click Add (+). Negotiate Authentication Configuration dialog box opens.
   - In the URL Host field, enter the address of the website you want to connect to in the form `HTTP://<host name>:<port>/<page>`, for example, `http://localhost:123/index.html`. Kofax RPA extracts the host name from the entered address, such as `http://localhost:123`. Note that `<port>` is an optional TCP port number you can use to specify a non-standard port number to differentiate between multiple instances of the same service on a single host computer. Ports 80 and 433 are omitted.
   - In the Server field, specify the name of the server/service in the form of a fully qualified domain name (FQDN). For example, `localhost:123` or `computer.global.companyname.com:1433`.

   When Kofax RPA loads a website in the WebKit browser and a server initiates Negotiate protocol usage, WebKit tries to match the host name with parameters specified by the user in the URL Host field. If a match is found, WebKit constructs an SPN string in the following form: `HTTP/Server`. Where `Server` is the FQDN with an optional port parameter specified in the Server field. For example, `HTTP/computer.global.companyname.com:1433`
3. Select Can Delegate if you want to turn on delegation usage for the specified account.
4. Click Save.

Use spn.txt to set Negotiate protocol parameters

Create `spn.txt` in the `bin` directory of the Kofax RPA installation directory, for example `C:\Program Files\Kofax RPA 11.1.0 x64\bin`. The file includes three parameters that can be specified independently.

spn.txt file format

Parameter	Description	Example
<host>:<port>::allow_port=[true|false]	Specifies whether to include port number in SPN. `false` (default): Do not include port number `true`: Include port number	`localhost::allow_port=true`
<host>:<port>::delegate=[true|false]	Turns on delegation usage for the specified account. `false` (default): Do not use delegation `true`: Use delegation	`localhost::delegate=true`

Parameter	Description	Example
<host>:<port>=<FQDN>	Enter host name in the form of a fully qualified domain name (FQDN) of the server. This parameter overrides the `allow_port` parameter and Kofax RPA uses the exact string specified here.	`localhost=COMPUTERNAME.companyname.com`

When Negotiate protocol is used in the environment with multiple websites running on the same hostname with different port numbers and using different application pool identities, you can set `allow_port` to `true` and specify a non-standard port for the SPN, for example:

```
localhost:8888::allow_port=true
```

Also, it is possible to use port as a part of the mask to assign SPN server, such as `localhost:8888=server:555`.

Logging

If you encounter errors during web authentication, you can turn on WebKit logging in the `log4j2.properties` file as follows:

```
logger.webkit.name = webkit
logger.webkit.level = TRACE
```

The log file should contain information about used authentication properties and SPN string. Look for the following lines in the log file

```
Setting SPN to : 'Service Principal Name (SPN)'
Delegate : [ON|OFF]
Non-standard port : [ON|OFF]
Setting NTLM SPN to : 'SPN string'
```

See Logging for details.

Predefined JavaScript Polyfills

The following is a list of predefined JavaScript polyfills. For more information see JavaScript Polyfills section under JavaScript Execution Tab in the Default Options in Basic Robot Configuration topic.

Object or API	Description	Notes
Array.prototype.values Array.prototype.keys Array.prototype.includes Array.prototype.findIndex Array.prototype.find Array.prototype.fill Array.prototype.entries Array.prototype.copyWithin Array.prototype.contains Array.prototype.@@iterator Array.of Array.from	The JavaScript Array is a global object that is used in the construction of arrays; which are high-level, list-like objects.	
Element.prototype.replaceWith Element.prototype.prepend Element.prototype.matches Element.prototype.closest Element.prototype.before Element.prototype.append Element.prototype.after	The Element is a part of a webpage.	
DOMTokenList.prototype.@@iterator	The DOMTokenList interface represents a set of space-separated tokens. It is indexed beginning with 0 as with JavaScript Array objects. It is always case-sensitive.	
IntersectionObserverEntry IntersectionObserver	An interface of the Intersection Observer API that provides a way to asynchronously observe changes in the intersection of a target element with an ancestor element or with a top-level document's viewport. The ancestor element or viewport is referred to as the root.	
Intl	A namespace for the ECMAScript Internationalization API, which provides language sensitive string comparison, number formatting, and date and time formatting. The INTL object provides access to several constructors as well as functionality common to the internationalization constructors and other language sensitive functions.	Supports the entire set of objects and the following languages: `da, de , en, ja, ru`
Map	An object that holds key-value pairs and remembers the original insertion order of the keys.	Supports the entire set of objects

Object or API	Description	Notes
Math.trunc Math.tanh Math.sinh Math.sign Math.log2 Math.log1p Math.log10 Math.imul Math.hypot Math.fround Math.cosh Math.clz32 Math.cbrt Math.atanh Math.asinh Math.acosh	The Math is a built-in object that has properties and methods for mathematical constants and functions. Not a function object.	
NodeList.prototype.forEach NodeList.prototype.@@iterator	NodeList objects are collections of nodes, usually returned by properties such as Node.childNodes and methods such as document.querySelectorAll().	
Number.parseInt Number.parseFloat Number.isSafeInteger Number.isInteger Number.MIN_SAFE_INTEGER Number.MAX_SAFE_INTEGER Number.Epsilon	The Number is a numeric data type in the double-precision 64-bit floating point format (IEEE 754). In other programming languages different numeric types can exist, for example: Integers, Floats, Doubles, or Bignums.	
Object.values Object.setPrototypeOf Object.keys Object.entries Object.assign	A constructor that creates an object wrapper.	
Performance	The Performance interface provides access to performance-related information for the current page. It is part of the High Resolution Time API, but is enhanced by the Performance Timeline API, the Navigation Timing API, the User Timing API, and the Resource Timing API. An object of this type can be obtained by calling the `window.performance` read-only attribute.	Supports the `PerformanceEntry` object. The `PerformanceEntry` object encapsulates a single performance metric that is part of the *performance timeline*.

Object or API	Description	Notes
Promise Promise.prototype.finally	The Promise is an object representing the eventual completion (or failure) of an asynchronous operation, and its resulting value.	
RegExp.prototype.flags	A property that returns a string consisting of the flags of the current regular expression object.	
Set	An object that allows to store unique values of any type, whether primitive values or object references.	Supports the entire set of objects
String.prototype.contains String.prototype.codePointAt String.prototype.endsWith String.prototype.@@iterator String.prototype.includes String.prototype.padEnd String.prototype.padStart String.prototype.repeat String.prototype.startsWith String.prototype.trim	All String instances inherit from String.prototype. Changes to the String prototype object are propagated to all String instances.	
Symbol Symbol.hasInstance Symbol.isConcatSpreadable Symbol.iterator Symbol.match Symbol.search Symbol.species Symbol.split Symbol.toPrimitive Symbol.toStringTag Symbol.unscopables	The Symbol is a primitive value, which is created by invoking the function Symbol, which dynamically produces an anonymous, unique value, and may be used as an object property.	
Url	An interface, which represents an object providing static methods used for creating object URLs.	Supports the entire set of objects
WeakSet	An object that is used to store weakly held objects in a collection.	
WeakMap	A collection of key/value pairs in which the keys are weakly referenced. The keys must be objects and the values can be arbitrary values.	

RoboServer

RoboServer is the Kofax RPA application for running robots as a service to clients. Once started, a RoboServer accepts requests from clients, such as robot execution requests, and sends back responses, such as the objects that a robot extracted during its execution.

For an in-depth description of the RoboServer, please see *Kofax RPA Administrator's Guide*.

Start RoboServer

You can start a RoboServer using the following actions.

- Click the RoboServer program icon (or the Start Management Console program icon which starts both the Management Console and RoboServer).
- Start the RoboServer from the command line.
- Run the server as a service. For more information about running RoboServer as a service, see Starting Servers Automatically.

To invoke a RoboServer from the command line, open a Command Prompt window, navigate to the `bin` folder in the `Kofax RPA` installation folder and type:

```
RoboServer
```

If all necessary parameters are specified in the roboserver.settings configuration file, the RoboServer starts.

If any of the necessary parameters is missing, the RoboServer specifies an error and displays the usage help and available parameters.

RoboServer Parameters

The command line for starting a RoboServer may include the following parameters:

```
RoboServer [-cl <arg>] [-cpuThreads <arg>] [-h]
       [-maxConcurrentRobots <arg>] [-maxQueuedRobots <arg>] [-MC] [-mcUrl
       <arg>] [-p <arg>] [-pauseAfterStartupError] [-s <arg>] [-sslPort
       <arg>] [-v] [-V]
```

Regardless of how you start RoboServer, it accepts the parameters in the following table. Note that you can edit all the parameters in the RoboServer Settings utility. See RoboServer Configuration for more details.

Parameter	Description
`-cpuThreads <arg>`	Specifies the number of CPU threads to use.
`-h` `--help`	Displays help.
`-maxConcurrentRobots <arg>`	Specifies the maximum number of concurrently executed robots.
`-maxQueuedRobots <arg>`	Specifies the maximum number of robots in the queue.

Parameter	Description
`-mcUrl <arg>`	Specify which Management Console to register to in the following format: `http[s]://<username>:<password>@<hostname>:<port number>` Example: `-mcUrl http://admin:password@localhost:8080/ManagementConsole`
`-pauseAfterStartupError`	

`-v |`
`-verbose`
	This optional parameter causes RoboServer to output status and runtime events.

`-V |`
`-version`
	This optional parameter causes RoboServer to output the version number, and then exit.

`-MC` `--scheduler`	This optional parameter triggers the Management Console to be started as part of RoboServer. The Management Console runs on an embedded web server configured through the Settings application.

`-cl`
`--cluster <arg>`
	This optional parameter automatically registers a RoboServer with the specified cluster on the Management Console. In the following example the RoboServer registers itself with the *Production* cluster.
	Example: `-cl Production`
	Example: `-mcUrl http://admin:password@localhost:8080/ManagementConsole -cl Production`

`-s <service-name:service-parameter> |`
`-service <service-name:service-parameter>`
	This parameter specifies a RQL or JMX service that RoboServer should start. This parameter must be specified at least once, and may be specified multiple times to start multiple services in the same RoboServer. The available services depend on your installation.
	Example: `-service socket:50000`
	Example: `-service jmx:50100`

`-p <port-number> |`
`-port <port-number>`
	This is shorthand for calling `-s socket:<port-number>`
	Example: `-port 50000`

`-sslPort <arg>`	This is a shorthand for writing `-s ssl:<port number>`
Available services	
`-service socket:<portNumber>`	`<portNumber>`: The port number for the socket-service to listen on.
`-service ssl:<portNumber>`	`<portNumber>`: The port number for the socket-service to listen on.

Parameter	Description
`-service jmx:<jmx_port_Number>,<jmx_rmi_url>`	`<jmx_port_Number>`: The port number for the JMX service to listen on. `<jmx_rmi_url>`: Optional RMI host and port for the JMX service. Use if you need to connect through a firewall. Example: `example.com:51001`

Note that to set passwords, you must either use the RoboServer Settings utility or the ConfigureRS tool. For more information, see RoboServer Configuration and RoboServer Configuration - Headless Mode.

Starting Servers Automatically

If your installation includes any server functionality, you can configure it to start the servers automatically.

When referring to "server functionality", we mean RoboServers and the Management Console (license server). In fact these two components are provided by RoboServer, depending on the arguments supplied to it when it starts.

The RoboServer Parameters topic contains a detailed description of the command-line arguments for a RoboServer program. To enable the RoboServer program to execute robots, specify the `-service` argument. Similarly, the `-MC` argument enables the Management Console functionality (see Management Console (License Server) in the Installation Guide).

The following topics details how to make RoboServer start automatically on Windows and Linux.

Shutting Down RoboServer

RoboServer can be shut down using the following command line tool. Run `ShutDownRoboServer` without arguments to see the various options for how to shut down the server, particularly how to handle any robots currently running on the server.

RoboServer Configuration

You can configure RoboServer through the RoboServer Settings application. RoboServer Settings can be started from the Windows Start menu. See RoboServer Configuration in the *Administrator's Guide* for the RoboServer configuration details.

Using this application, you can configure the following:

- General: Socket service options, enable and configure JMS Service options, connection options to connect to a Management Console, and the Verbose option.
- Security: Security settings such as authentication and permissions.
- Certificates: The use of certificates.
- Project: The location of the default v project.
- JMX Server: The JMX server.
- Management Console: Configuration of the embedded Management Console.

After changing any of the settings, click OK to store the new settings, and then restart any RoboServers that are running, to make the changes take effect.

Important When using embedded Derby database, robots can create and edit files on computers even when the **Allow File System and Command Line Access** option on the **Security** tab is not selected. We recommend using MySQL or another enterprise-class database in your network environment.

Starting from Kofax RPA version 10, all RoboServers must auto register to the Management Console. Therefore, the URL and credentials for the Management Console along with the cluster and service name must be specified when starting RoboServers (either at the command line or on the **General** tab of the RoboServer Settings application).

Kofax RPA contains several command-line tools to help you modify the settings in batch mode. For example, you can create several users with specified permissions. See Configuring RoboServer in Headless Mode.

If you need to change the maximum amount of RAM that RoboServer can use, see Change the RAM Allocation in the *Administrator's Guide*.

RoboServer Configuration - Headless Mode

Kofax RPA ships with several utilities to configure your RoboServer from a command line. The utilities are located in the `bin` subfolder of the Kofax RPA installation folder. Note that the configuration files are user-dependent and stored in the user folder. For more information, see the Important Folders topic in the Kofax RPA Installation Guide.

- ConfigureRS: Sets the JMX password and the Management Console password in the RoboServer settings file (`roboserver.settings`).
- ConfigureMC: Sets protocols and ports usage, certificate passwords, and upload JDBC jar files permission in the `mc.settings` file.
- ConfigureRSUser: Adds and removes users and updates user credentials in the rsusers.xml file. Information in this file is used to authenticate API requests.

For help on usage, run utilities with an -h option.

To set a connection to the Management Console that the RoboServer will register to, type the following command:

```
ConfigureRS -mcUrl http://admin:password@localhost:8080/ManagementConsole
```

Note The default `admin` superuser credentials are: user name - `admin`, password - `admin`.

To create a user user1 with Password1 password and all permissions type the next command:

```
ConfigureRSUser user1 Password1 -a
```

To enable authentication of API requests, you must open rsusers.xml and change the enabled attribute to true, as shown in the following example.

Sample rsusers.xml configuration file

```
<?xml version="1.0" encoding="UTF-8"?>

 <userConfiguration enabled="true">
   <users>
```

```
     <user username="user1"
 password_hash="20c7628c31534b8718a1da00435505e4262e3f4dc305">
       <startRobot/>
       <stopRobot/>
       <shutdownRoboServer/>
     </user>
   </users>
 </userConfiguration>
```

Sample roboserver.settings configuration file

```
# Settings file for RoboServers. Some configurations contains encrypted passwords and
 should not be edited by hand,
           these should be modified using dedicated commandline tools.

# The directory for use on RoboServers when the API used the DefaultRoboLibrary. On
 Windows \ must be escaped in the following way:
c:\\\\users\\AppData\\Local\\Kofax RPA\\...
defaultProject = /home/TestUser/Kofax RPA/trunk

# Should RoboServers be allowed to access the fileSystem, or call commands/scripts.
 Values: true/false
sec_allow_file_system_access = false

# Will RoboServers accept JDBC drivers sent from the Management Console. Values: true/
false
sec_accept_jdbc_drivers = true

# Should RoboServers log all loaded URLs to the log4j2 audit log. Values true/false
sec_log_http_traffic = false

# If enabled RoboServers will check credentials for API requests. Values: true/false
sec_authenticate_api_requests = false

# If enabled RoboServers generate an error when accessing a https site without a valid
 certificate. Values: true/false
cert_verify_https_certificates = false

# If enabled, RoboServers will only allow SSL connections from trusted client. Values
 true/false
cert_verify_api_certificates = false

# Configures if the the JMX service should be enabled
enable_jmx = false

# The port number for the JMX service to listen on.
jmx_port_Number = 50100

# If enabled, input for robots is exposed through JMX. Values: true/false
jmx_show_inputs = true

# Heartbeat notification interval, in seconds
jmx_heartbeat_interval = 0

# Configure if JMX should use RMI
enable_jmx_rmi = false

# Optional RMI host and port for the JMX service. Use if you need to connect through a
 firewall. Example: example.com:51001
jmx_rmi_url =

# Enables authentication for JMX requests. Values: true/false
jmx_enable_authentication = true
```

```
# The user-name used for JMX authentication
jmx_username =

# The password used for JMX authentication. This should be created using the
 ConfigureRS command line tool.
jmx_password =

# Configures if the socket service should be enabled
enable_socket_service = false

# Configures which port the RoboServers should be listening on
port = 50000

# Configures if the ssl socket service should be enabled
enable_ssl_socket_service = false

# Configures which ssl port the RoboServers should be listening on
ssl_port = 50001

# Configures if the JMS service should be enabled
enable_jms_service = false

# Configures which id the RoboServers should have when running JMS
jms_id = 1

# Configures the URL of the message broker when running JMS
broker_url =

# Specify which Management Console to register to formatted as: http[s]://
<hostname>:<port number>
mc_URL =

# The user name to use for authentication to the Management Console
mc_username =

# The password to use for authentication to the Management Console
mc_password =

# Specifies which cluster the RoboServers should be registered to
cluster =

# Causes RoboServers to output status and runtime events
verbose = false
```

Sample mc.settings configuration file

```
# Settings file for Management Console. Passwords should not be edited by hand, but
 using the 'ConfigureMC' command line utility.


 # Should the MC web-server start a HTTP listener. Values true/false
 mc_http = true

 # Configures the port of the http listener.
 mc_http_port = 50080

 # Should the MC web-server start a HTTPs listener. Values true/false
 mc_https = false

 # Configures the port of the HTTPS listener.
 mc_https_port = 50443

 # Password for the certificate used by the HTTPs listener. This should be created
 using the ConfigureMC command line tool.
```

```
mc_https_cert_password = 3W2MTrL/b2k=

# Configures which hosts are allowed to upload JDBC jar files to MC. Values: NONE,
LOCALHOST, ANY_HOST
mc_allow_jdbc_upload = LOCALHOST
```

Use Proxy Services

Some IP proxy providers are beginning to offer built-in IP rotation services. Although this is convenient, it is not always a useful service for those who need IP proxies.

Kofax RPA does not recommend using an IP-rotation model where the proxy provider rotates the IP at random or pre-set times, because not all web sites are able to or even allow maintaining a browser session across an IP address change.

On the open Internet, IP rotation can work just fine, because the typical browser socket connection has a very short life and the web site does not check the source IP address. Many web sites and shops implement the most basic session management. With the increase in cyber-threats and focus on security, many web sites are increasing their security level and adding IP address monitoring. It is a good practice for web servers to detect and prevent in-session IP address changes for protection against Man-in-the-Middle attacks. This is why all web banking sites and many other commercial and financial services sites with user login are implementing protection against in-session IP address changes.

As changing the IP address is likely to break the session in progress with the web-server, you cannot arbitrarily rotate IP addresses while running robots. The best way to use proxies effectively is to control the proxy from within the robot, for instance by using the Change Proxy step, or by using web services provided by the proxy service.

If the rotation is done in the robot, with consideration of the remote website, in a way the transactions are made within an IP address session, the robot should work fine.

See also:

- Change Proxy step
- Proxy Servers in Design Studio settings
- Configuring Proxy servers in Management Console

Kofax RPA Limitations

The following section describes some limitations of the Kofax RPA product.

Browser

- Kofax RPA uses a browser to interact with web sites or web applications. In fact, Kofax RPA currently comes with two different browsers, each optimized for different purposes - the Classic engine for legacy web applications and the Default engine for modern web applications. However, these browsers - like any other browsers in the market - are not compatible with every internal application or Internet web site.

- The web pages downloaded using Make Snapshot are a representation of the content and styles downloaded from the internal browser in Kofax RPA. When viewing the downloaded pages in a desktop browser, they may render differently from viewing the original web page in that browser.

Excel

- When looping over a global Excel variable, certain step actions are not allowed inside the loop, that is after the loop step. This is dynamically enforced, in other words the error does not occur until the steps are executed. The following step actions will always refuse to work on a global variable that the robot loops over inside the loop: Insert Rows, Insert Columns, Remove Rows, Remove Columns and Set Sheet Name.
- Excel modification is memory intensive and might not work on large Excel documents. The limit may depend on many factors, for example, available memory on a design platform or a server platform, how many modifications the robot does, etc. Therefore it is not possible to give precise criteria for when an Excel document is too large for Kofax RPA to handle.
- Formula support

  Search the https://poi.apache.org site for details on formula support in Excel.
- Unsupported Features
  - Manipulating array/table formulas (In Excel, formulas that look like "{=...}" as opposed to "=...")
  - Region operators: union, intersection
  - Parsing of previously uncalled add-in functions
  - Preservation of whitespace in formulas (when POI manipulates them)
  - Font changes, for example bold, size, etc.
  - Background color in cell
  - External file references from formulas
  - Excluding hidden values
  - Converting strings to a date with the FLOOR, HOUR, and MINUTE functions. Only decimal values are supported.
- Sometimes, when your robot performs some actions on a very large Excel document, the processing may become slow after several hundred iterations. To speed up the process, use the following settings in your robot for Excel processing:
  - Use Global variable
  - Do not use the Ignore and Continue error handling
  - Do not run your robot in Design mode

  > **Note** With all the above-mentioned conditions, if an error occurs when working with Excel, the variable value is set to empty. You will need to inspect your robot to make sure you use the supported Excel capabilities and correct the error to have a valid value in your variable.

See Supported Features in Excel for more information.

Password Length

All passwords used in the Kofax RPA, for example, passwords for Cluster database, Logdb, Analytics, proxy, and SMTP, must not exceed 117 bytes.

Database

Note the following limitations for IBM DB2 database:

- The database must have a "table space" with a page size of at least 8192 KB to create all tables.
- Calling a stored procedure on a DB2 database is not supported.

Execution Mode

The following list contains steps that are not available in the Smart Re-execution mode.

- Call REST Web Service (old version)
- Call Web Service (old version)
- Clear Web Storage
- Divide Tag (old version)
- Divide Text
- Divide Text (old version)
- Execute SQL (old version)
- Extract from Excel (old version)
- Extract from PDF (old version)
- Extract Image
- For Each File (old version)
- Hide Tag
- Insert Tag
- Load Data (old version)
- Make Snapshot
- Normalize Table
- Query Database (old version)
- Remove Table Rows
- Remove Tag Range
- Remove Tags
- Remove Tags (old version)
- Replace Tag
- Restore Tag
- Rewrite Page
- Rewrite Style Sheet
- Select File
- Set Top Tag (old version)
- Submit Form
- Transpose Table
- Unhide Tag

Kofax RPA Kapplets

The following list contains features that are not available in the Kofax RPA Kapplets.

- Import and export Kofax RPA Kapplet backups
- WebSphere application server support
- DB2 database support
- Scheduling for Kapplets
- Adding OAuth robots to Kapplets

Using Kapplets with robots containing big volume data output is not recommended.

General

When collecting a large number of data elements, we recommend you to structure your robot for a divide-and-conquer approach, so that each run of the robot collects only a portion of the data elements.