



Pega Robotic System Architect Essentials

Student Guide

© 2019 Pegasystems Inc.



© 2019
Pegasystems Inc., Cambridge, MA
All rights reserved.

Trademarks

For Pegasystems Inc. trademarks and registered trademarks, all rights reserved. All other trademarks or service marks are property of their respective holders.

For information about the third-party software that is delivered with the product, refer to the third-party license file on your installation media that is specific to your release.

Notices

This publication describes and/or represents products and services of Pegasystems Inc. It may contain trade secrets and proprietary information that are protected by various federal, state, and international laws, and distributed under licenses restricting their use, copying, modification, distribution, or transmittal in any form without prior written authorization of Pegasystems Inc.

This publication is current as of the date of publication only. Changes to the publication may be made from time to time at the discretion of Pegasystems Inc. This publication remains the property of Pegasystems Inc. and must be returned to it upon request. This publication does not imply any commitment to offer or deliver the products or services described herein.

This publication may include references to Pegasystems Inc. product features that have not been licensed by you or your company. If you have questions about whether a particular capability is included in your installation, please consult your Pegasystems Inc. services consultant.

Although Pegasystems Inc. strives for accuracy in its publications, any publication may contain inaccuracies or typographical errors, as well as technical inaccuracies. Pegasystems Inc. shall not be liable for technical or editorial errors or omissions contained herein. Pegasystems Inc. may make improvements and/or changes to the publication at any time without notice.

Any references in this publication to non-Pegasystems websites are provided for convenience only and do not serve as an endorsement of these websites. The materials at these websites are not part of the material for Pegasystems products, and use of those websites is at your own risk.

Information concerning non-Pegasystems products was obtained from the suppliers of those products, their publications, or other publicly available sources. Address questions about non-Pegasystems products to the suppliers of those products.

This publication may contain examples used in daily business operations that include the names of people, companies, products, and other third-party publications. Such examples are fictitious and any similarity to the names or other data used by an actual business enterprise or individual is coincidental.

This document is the property of:

Pegasystems Inc.
One Rogers Street
Cambridge, MA 02142-1209
USA
Phone: 617-374-9600
Fax: (617) 374-9620
www.pega.com

Course Objectives

After completing this course, you should be able to:

- Apply programming experience to develop a multi-project solution
- Create a standard Windows form using .NET window controls
- Integrate and interrogate a Windows application within the solution
- Integrate and interrogate a Web application within the solution
- Create a unified solution using project-to-project references
- Employ the use of Interaction Framework within the unified solution to seamlessly maintain and transfer data and activities across the solution

Course Objectives (continued)

- Create a project and solution hierarchy structure to easily maintain and scale the unified solution
- Create automations to streamline the process by integrating the multi-project solution through programming logic
- Add and edit match rules on interrogated objects to create an understanding of the matching process for both Window and Web applications
- Generate build files and deployment packages of the unified solution
- Utilize the standard Visual Studio diagnostic and debugging tools to test the unified solution



MODULE Introduction to Pega Robotic Automation

This module contains the following lessons:

- Introduction to Pega Robotic Automation
- Pega Robot Studio IDE basics
- RDA and RPA architecture
- Common Tool windows
- How to organize the workspace

LESSON

Introduction to Pega Robotic Automation

Pega Robotic Automation™ automates work more quickly while also improving the customer and employee experience.

After this lesson, you should be able to:

- Identify features and benefits of Pega Robotic Automation
- Differentiate between Pega Robotic Desktop Automation™ and Pega Robotic Process Automation™

Overview of Pega Robotic Automation

Pega Robot Studio enables end users to create run-time solutions that improve productivity and efficiency.

For example, you can create solutions that perform the following tasks:

- Automate manual processes such as copying and pasting between applications.
- Provide a single user interface for interaction with multiple applications.
- Invoke a web service.

Overview of Pega Robotic Automation (continued)

- The Pega Robot Studio application provides base adapters for integrating desktop applications without modifying the underlying application software.
 - Using the base application adapters, you can select specific application controls such as text-entry controls and web links.
 - You can then use the properties, methods, and events of the controls to design application automations and event monitoring solutions.

Robotic Desktop Automation versus Robotic Process Automation

Robotic Desktop Automation	Robotic Process Automation
Attended automation	Unattended automation
Runs on agent's desktop and invoked by the user	Runs on a VM and do not require human intervention
Works alongside agents to improve productivity and quality	Can run on a dedicated workstation to execute fully automated processes
Automates 20-90% of work	Automates 100% of work

Benefits of Pega Robotic Automation

Implementing Pega Robotic Automations into the daily workflow helps:

- Reduce risk
- Increase compliance
- Reduce errors and rework
- Reduce redundancy and variability
- Reduce need for training and retraining

How does Pega Robotic Automation Work?

Pega Robotic Automation works through three concepts:

- Event-driven architecture
- Supports virtually any object
- Delivers quick results

LESSON

Pega Robotic Desktop Automation and Pega Robotic Process Automation Architecture

In order to developer robotic solutions that meet project needs developers use either Pega Robotic Process Automation™ or Pega Robotic Desktop Automation™.

After this lesson, you should be able to:

- Differentiate between the architecture structure of Pega Robotic Desktop Automation and Pega Robotic Process Automation

Key Differences In Automations

Pega Robotic Desktop Automation (RDA)	Pega Robotic Process Automation (RPA)
Attended automation	Unattended automation
Executes workflow on an end user's desktop to assist in human work.	Runs on a virtual machine and does not require human intervention
Works alongside agents to improve productivity and quality	Can run on a dedicated workstation to execute fully automated processes
Automates 20-90% of work	Automates 100% of work

Pega Robotic Desktop Automation

Pega Robotic Desktop Automation is triggered by specific events, actions, or commands (an employee engages within a specific workflow) from the end-user desktop.

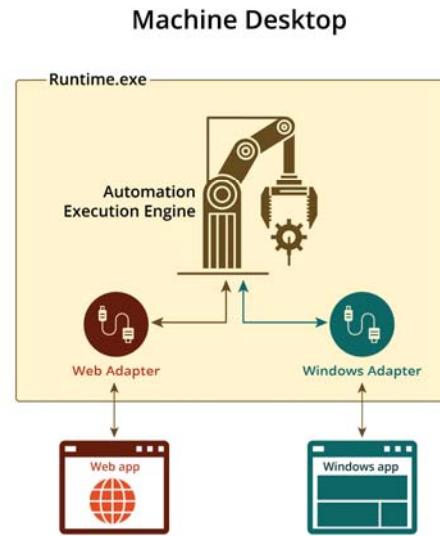
The solution executes workflows on the end-user desktop in order to assist human work.

Pega Robotic Desktop Automation (continued)

Pega Robotic Automation Runtime automation desktop contains Pega Robot Runtime and all the enterprise applications required for the robotic solution.

The Automation Execution Engine of the Pega Robotic Automation Runtime automation desktop connects with enterprise applications using adapters.

The robot executes the workflow on the machine desktop. It uses the adapters to interact with enterprise applications during the execution.



© 2019 Pegasystems Inc.

11

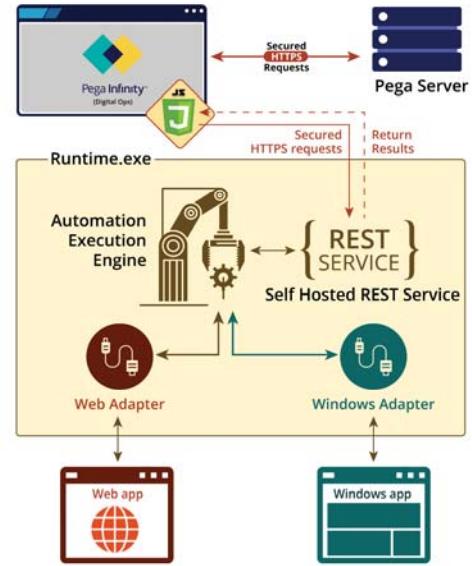
An adapter is a component that lets users integrate with an application built upon a specific platform (Windows, web, and so on).

Adapters have accessible properties, methods, and events that launch, monitor and expose enterprise applications for automation.

Executing RDA From Pega Server

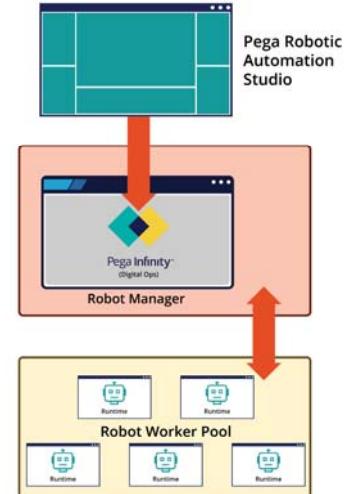
Pega Robot Runtime uses a self-hosted REST service with a Pega application. The process is as follows:

1. Requests from the Pega server send to self hosted REST service through Application JavaScript in the form of secured HTTPS requests.
2. The self-hosted REST service enables the automation engine to perform the associated automation.
3. Results of the automation pass to the Pega browser application in the form of secured HTTPS requests through self-hosted REST service.



RPA Architecture

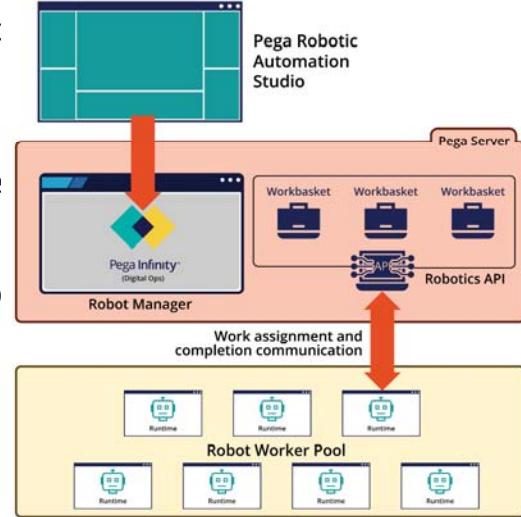
- The solution is deployed through Pega Robot Manager to execute on a runtime virtual machine from the Robot worker pool.
- The robot worker pool is a collection of individual runtime machines connected to the Pega Robot Manager.
- Each runtime machine executes the robotic solution using its Automation Execution Engine.



Pega Robotic Process Automation

Robotic Process Automation with a Pega server uses the Pega case management workflow and workbasket to execute robotic solutions. The process is below:

1. Robotics API manages the task from a workbasket and assigns it to a runtime available in the robot worker pool.
2. The Robotics API communicates the status of the automation to the Pega Server.
3. Pega Robot Manager orchestrates the deployment of robotic solutions and monitors the robots in the Robot worker pool.



Pega Robot Studio™ (PRS) provides a unique, intuitive, and rapid visual development environment to create robotic automation solutions and projects efficiently.

After this lesson, you should be able to:

- Identify the integrated development environment (IDE) interface components by their use and purpose.

Integrated Development Environment

- The environment of Pega Robot Studio is similar to any other integrated development environment (IDE) that provides comprehensive facilities for software development.
- Pega Robot Studio IDE works within the Microsoft Visual Studio.
- Both standalone or plug-in versions of Pega Robot Studio provide streamlined access to all the tools required for project design, debugging, and deployment packaging.

Interface Elements

Studio consists of five interface elements:

- Menu toolbar
- Standard toolbar
- Automation toolbar
- Tool windows
- Designer windows



Menu toolbar



Standard toolbar



Automation toolbar



Tool windows



Designer windows

© 2019 Pegasystems Inc.

17

Menu toolbar – provides access to the windows menu categories and functions

Standard toolbar – contains standard functions (File, Open, Save, Cut, Copy, and Paste) as well as Studio-specific functions. The toolbar also changes to provide different options based on the active designer window or the invoked function.

Automation toolbar - contains buttons for performing automation view and layout actions such as grouping automation lines, error handling, and modifying the viewing size of an automation.

Tool windows - provide access to automation component (solution, project, and project item) related functions that help configure various items in the automation.

Designer windows - are created when you add or open a project item, which results in change of Studio toolbars and tool windows to provide access to the development components of project item.



DEMO

Tour of Pega Robot Studio

© 2019 Pegasystems Inc.

18

The Integrated Development Environment (IDE) in Pega Robot Studio provides Tool windows and document windows.

After this lesson, you should be able to:

- Identify each Tool window by purpose and feature

Tool Window Overview

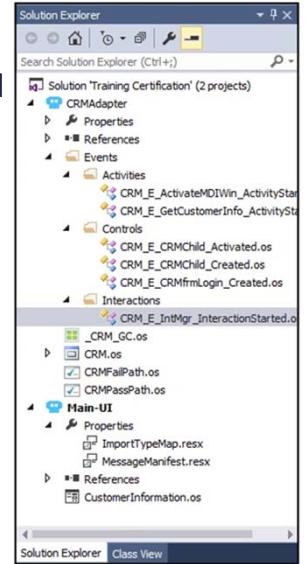
The four most commonly used windows for development are:

- Solution Explorer
- Object Explorer
- Properties
- Toolbox

Solution Explorer

The Solution Explorer mimics the Windows Explorer in that the solution, projects, and project items are organized alphabetically in a parent/child relationship. You will learn about the following project items in later lessons:

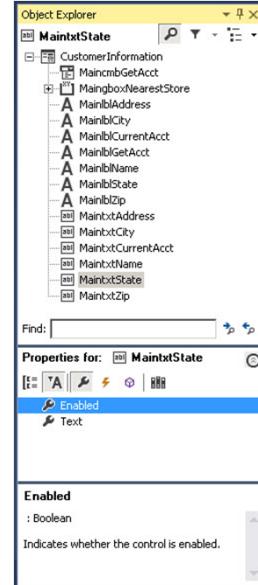
- Adapters
- Automations
- Windows forms
- Folders
- Global containers
- Configuration project items



Object Explorer

The Object Explorer lists all adapters, forms, interrogated controls, automations, and components contained in a project. The Object Explorer provides access to the properties, events, and methods of these items.

The Object Explorer is separated into two parts: Object Hierarchy and the Object Inspector.



© 2019 Pegasystems Inc.

22

The Object Hierarchy displays changes based on which project item is active in the designer window area.

The Object Hierarchy displays Project Items including:

Adapters

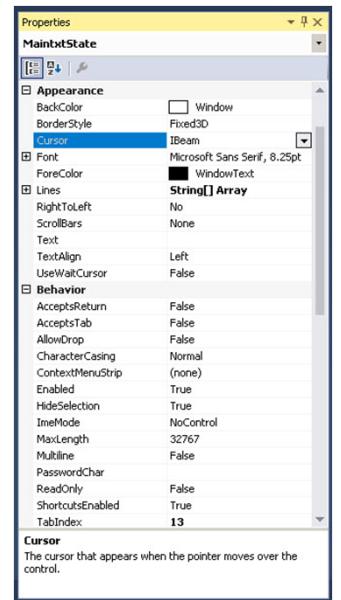
Interrogated controls

Automations in a parent/child relationship

The Object Inspector provides access to an objects library of properties, events, and methods for use in developing automations; it sorts the items by category or alphabetically. The lower box displays a usage definition when selecting an item.

Properties

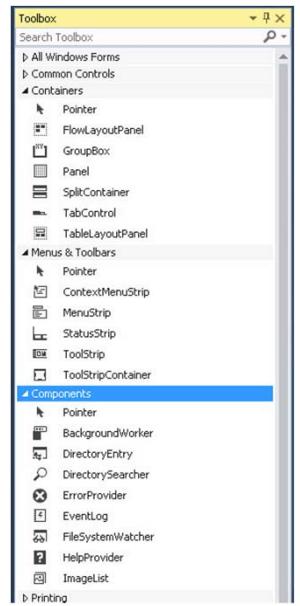
The Property window provides access for adjusting a project item or object properties. The window displays different properties based on the project item or object selected in the Solution Explorer or the Object Explorer.



Toolbox

The Toolbox window contains .NET components and Studio components available for use in solutions. Standard .NET components are useful when creating a Windows form and application bars in solutions.

The toolbox divides the components into specific categories, such as Windows Forms, Databases, Variables, Expressions, and Pega application integration.





DEMO

How to Organize the Workspace



MODULE Project Documentation

This module contains the following lessons:

- Solution Design Document
- Solution Architecture Document

The Solution Design Document outlines the general role in how a robotic solution fits within business requirements.

After this lesson, you should be able to:

- Recognize the benefits of the Solution Design Document
- Identify the importance of various sections of the Solution Design Document

Solution Design Document Overview

The Solution Design Document helps the developer envision the processes and provides a roadmap to understanding how to deploy and use the robotic solution.

The Solution Design Document:

- Provides the business requirements and functional specifications of the robotic solution
- Documents the business need or gap (for example, reduce compliance errors on data entry) and how the robotic solution attains or removes the gap
- Includes information about system requirements, integrations, and UI elements



Solution Design Document Benefits

Solution Design Document provides:

- The overall goals and objectives of the robotic solution
- The outline of automation processes needed to complete the solution
- The context of how project pieces fit together while working on a single area of the solution
- The division of work in a multi-developer project to integrate later into a completed solution

Solution Design Document Components

Overview

- Provides a high-level description of the solution
- Presents the business driver and the success metrics that the deployed solution must meet
- Explains the reasoning behind the solution, its requirements, and priority.

Solution Design Document Components (continued)

Story Board

- Maps the solution through the different solution processes
- Outlines the overall flow of the solution and the interaction points between projects and applications

Story Board Features

The Story Board describes the workflow and specific tasks of the solution through three features.

- **UI feature** – Describes the look and feel of a user interface and what processes it may kick off in the Rules/Process features of the Story Board
- **Data feature** outlines:
 - Logins
 - Required data items
 - Development configuration considerations
 - Security elements
 - Interaction points between projects and applications
- **Rules/Process feature** – Provides steps to automate by process across each project and application

Additional Information

- Typically robotic projects are divided by adapters, UI, and other integration points based on the Solution Design Document and the Architecture document.
- The Solution Design Document provides a central location of all specifications to assist multiple developers in integrating processes into a completed solution.
- The nomenclature and format of this document might be different, but the essence of the document remains same.

Solution Architecture Document goes deeper into the technical structure and project requirements of a robotic solution.

After this lesson, you should be able to:

- Recognize the benefits of the Solution Architecture Document
- Identify the importance of various sections of Solution Architecture Document

Solution Architecture Document Overview

Solution Design Document:

- Provides design specifications required to configure the solution and its projects
- Outlines the solution's development methodology and technical aspects
- Includes information for specific Pega Robotic configurations of components and variables for use in the projects and solution



Solution Architecture Document Benefits

- Provides a detailed view of configuration, design, and implementation of the robotic solution
- Explains in detail, features of the Story Board sections in the Solution Design Document to guide the developer on where to start and what to implement
- Ensures all the modules of the project follow the same naming conventions and standards to minimize changes to the projects when merged to create the full robotic solution

Solution Architecture Document Components

The components of the Solution Architecture Document depend on the scope of the solution. The most commonly used sections are:

- Overview
- Special Limitations
- Solution Configuration
- Solution Design
- Project Implementation Details

Best Practices

Before developing a robotic solution, consider the following guidelines.

- Try the application
- Create a testing solution
- Repeat steps with variation

-Run through the application before interrogation and development of the project to capture any missing application events, navigation steps, and user interactions not found in the documentation.

-Create a solution to interrogate and review the application's controls. This allows a clean space for modifications and adjustments. Once the application review is complete, the project can be imported into the development solution.

-In the testing solution, repeat the documented process by entering or selecting different values and options to ensure that interrogated controls work in as many variations as possible. If necessary, modify any control's match rules to ensure consistent availability for use in the project.



MODULE Solutions and Projects

This module contains the following lessons:

- Robotic solutions and projects
- How to create a solution and project

LESSON

Robotic Solutions and Projects

Starting a robotic project requires recognizing the types of projects and naming conventions. Understanding how each pertains to the robotic solution ensures unified solution development.

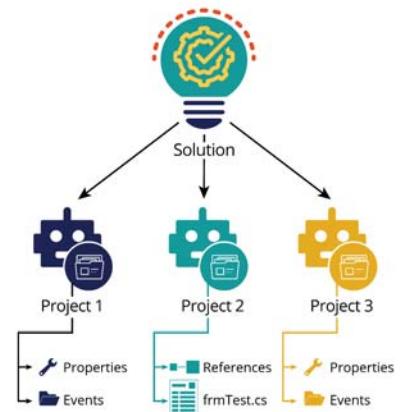
After this lesson, you should be able to:

- Recognize the solution structure based on project documentation
- Define types of projects created for a robotic solution
- Recognize proper naming conventions of solutions and projects

Overview

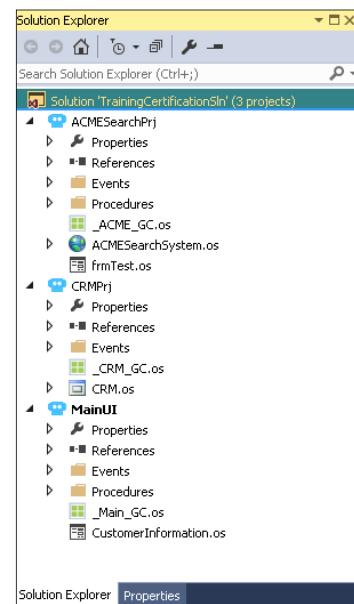
Before development begins, an understanding of the relationship between solutions, projects and project items ensures unified solution development with other developers.

Organizing solutions and projects in a hierarchy gives developers flexibility to maintain, enhance, and expand project requirements.



Solution

- A solution is a container for the projects, project items, additional files, and resource files used to build a runtime project.
- Each solution must contain at least one project.



Solution: Naming Conventions

Solution names identify the overall functionality of the solution.

Each solution name describes the purpose of the solution and includes a **Sln** suffix. Each term has a capital letter and there are no-spaces between terms.

For example:

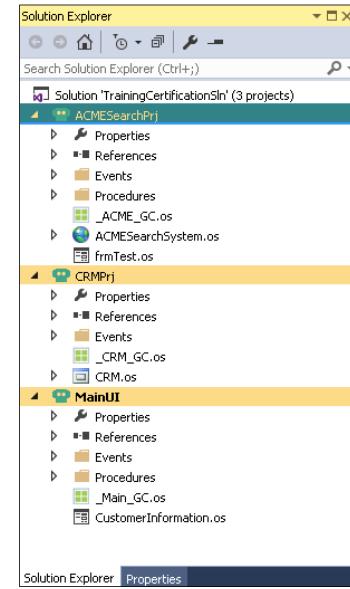
Type	Purpose	Example Name
Solution	Training Certification	TrainingCertificationSln

Projects

A project is a logical container for all items needed to build your application.

Each project contains items such as automations, adapters, user interfaces or Windows forms, global containers, and folders for organization.

Each project is an individual module used to build an overall solution.



Types of Projects

Projects can be divided into two types:

- **Controllers**
 - Functions as the main project of a solution
 - Contains the logic required to connect the individual project applications together
- **Adapters**
 - Automates the functions to be performed against an external application
 - Uses a single application for each adapter project

Project: Naming Convention

Project names identify either the application or business process performed

Controller projects do not end with the **Prj** suffix

Projects containing business logic or application interaction (adapter) include the descriptive name with a **Prj** suffix

For example:

Type	Purpose	Example Name
Controller	Loan Support	LoanSupport
Business Logic	Validate Customer	ValidateCustomerPrj
Application	Automate Siebel	SiebelPrj



DEMO

How to Create a Solution and Project



MODULE System Integration: Adapters

This module contains the following lessons:

- Overview of adapters
- Windows adapters
- Web adapters
- Universal web adapters

LESSON Overview of Adapters

Pega Robot Studio™ uses adapters to integrate and interact with systems. Communication between applications is difficult when applications are separate. Companies can improve workflow by using adapters to integrate applications to interact with one another.

After this lesson, you should be able to:

- Recognize the Pega Robotic Automation™ adapter types
- Recognize the most common adapters used in Pega Robotic Automation

Adapters

An **adapter** is a class that encapsulates the data in an application and permits automation to monitor and modify how the application works. Adapters have accessible properties, methods, and events that launch, monitor, and expose enterprise applications for automation. Pega Robot Studio does not require the target application to be recompiled or modified.

Adapter Type	Applications
Windows	Windows applications, including Java applications and SAP GUI
Web	Web-based applications that run on Internet Explorer
Universal Web	Web-based applications that use Google Chrome and Mozilla Firefox
Text Adapter	Host-based or mainframe applications accessed through an emulator

LESSON

Windows Adapters

When a developer is ready to add another application to a solution, the developer must first consider the type of adapter that is necessary based on the type of application.

After this lesson, you should be able to:

- Recognize the common adapter properties and values for Windows adapters

Windows Adapter

A Windows adapter is added to a project for a Windows application. The following adapter properties apply to a Windows application:

- Path
- TargetPath
- HookChildProcesses
- StartMethod
- StartOnProjectStart
- WorkingDirectory
- Arguments
- HideApplicationAtRuntime

Path Property

Enter the full path in the **Path** property and the executable file name when the application runs from the same installation folder of the program. The adapter requires the Path property completed. Here is an example:

- C:\Windows\System32\notepad.exe

If the application is in the system path, enter the application file name without the full path. Here is an example:

- CRM.exe

Path Property (continued)

The Path property does not support links (lnk), wildcards, or Regex text entries. If you are deploying a solution where the target application runs in different folders on the desktops, you can use:

- Different solution configurations
- The Folder sub-property under the Path property

The **Folder** property allows you to select a system folder and file location for the installed application directory.

TargetPath Property

The **TargetPath** property references an application launched as the result of one or more other processes (applications) occurring first. The Path in the TargetPath property refers to the executable that starts the target application.

To use the Path/TargetPath relationship, you must change the StartMethod property from Start to StartAndWait. Using the Path/TargetPath configuration, you cannot interrogate the Path application. You can interrogate only the TargetPath application.

Note: Java-based applications use this configuration often.

HookChildProcesses Property

The **HookChildProcesses** property configures the solution to integrate with an application and any application launched from that application. Set the property to True when:

- One application starts other applications
- Both parent and spawned applications are required for the solution

As a result, Pega Robot Studio must hook into, or integrate with, each application process.

StartMethod Property

The StartMethod property determines how the Path application starts.
Options include:

Option	Description
Start	Tells Pega Robot Studio to launch and hook the application defined by the Path property after the adapter starts. This is the default.
StartAndWait	Tells Pega Robot Studio to launch the application identified in the Path property after the adapter starts, wait for the TargetPath application to launch, and then hook the TargetPath application. Note: Pega Robot Studio does not hook the Path application; thus, you cannot interrogate or automate the Path application.
MonitorAll	Tells Pega Robot Studio to wait for the Path application to run and then hooks the application. In this case, starting the adapter does not cause the Path application to launch. The Path application launches independently after the adapter starts (for example, from an external application, or manually). Once the application launches, Pega Robot Studio hooks the application. When using the MonitorAll, you can omit the full path when specifying the application executable in the Path property and enter the executable file name. Note: Using MonitorAll lets the adapter to remain running if the Path application shuts down because of an application error or the end-user closing the application. When re-starting the application through an automation or by the end-user, Pega Robot Studio will hook the application.

StartOnProjectStart property

Use the **StartOnProjectStart** property to start the adapter when the solution or project starts. The default setting for this property is True.

If StartOnProjectStart is False, you must start the adapter first in an automation. When you set the StartOnProjectStart property to False, the Path and/or TargetPath applications do not launch when the project starts, regardless of the associated adapter StartMethod value.

Pega recommends leaving the default setting for those applications required to run when the solutions runs. A common usage of this property is to control the login process and application start up time for end-users.

Working Directory Property

Studio populates this field with the location of the **Working Directory property**.

Modify the working directory to point to the supporting installation files when different from the install directory of the executable file.

Arguments

Use the **Arguments** property to enter command-line arguments required for starting the application associated with the adapter.

- For example, you could enter the name of a file to launch when the application launches.

You can also use this property to specify the Java class names for Java applications.

HideApplicationAtRuntime Property

Set the **HideApplicationAtRuntime** property to true to hide the application during project runtime.

To show the application, call the Show method on the adapter. You should use this method sparingly. Instead, develop options such as progress bars or grayed-out applications to notify the user when an automation or process completes.

LESSON

Web Adapters

A second widely used adapter is the **Web** adapter.

After this lesson, you should be able to:

- Recognize the common adapter properties and values for Web adapters

Web Adapter

Web and Windows adapters share some properties. Developers should be able to recognize the most used web properties to understand how they influence the application's use in the solution.

The following adapter properties apply to a web application:

- StartPage
- StartOnProjectStart

StartPage Property

The **StartPage** property specifies the first HTML page to open for the application. The property accepts the site location in several formats.

The StartPage property is synonymous with the Path property for the windows application. However, the StartPage property is optional for a web application. If left blank, the adapter opens the browser and navigates to the users home page. The StartPage property sets the page to load when the adapter starts.

StartOnProjectStart Property

The **StartOnProjectStart** property applies to all Pega Robot Studio adapters. This property controls when an adapter starts during the course of loading and running the project. The default is True, which means the adapter starts when the project runs. If you set this property to False, specific automation logic is required, such as executing the Adapter.Start method, to start the adapter.

LESSON

Universal Web Adapter

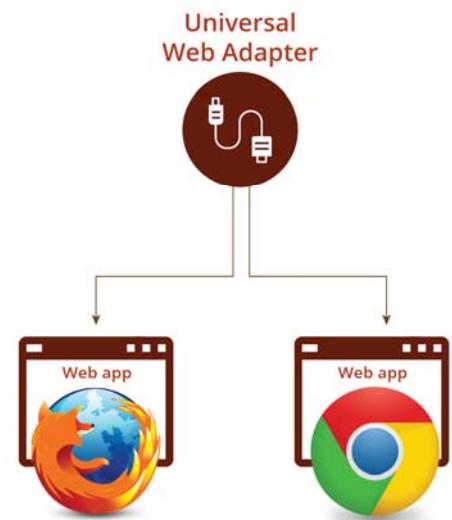
After this lesson, you should be able to:

- Recognize the common adapter properties and values for Universal Web adapters

Universal Web Adapter

The **Universal Web** adapter integrates Pega Robot Studio with web applications using Firefox or Chrome browsers.

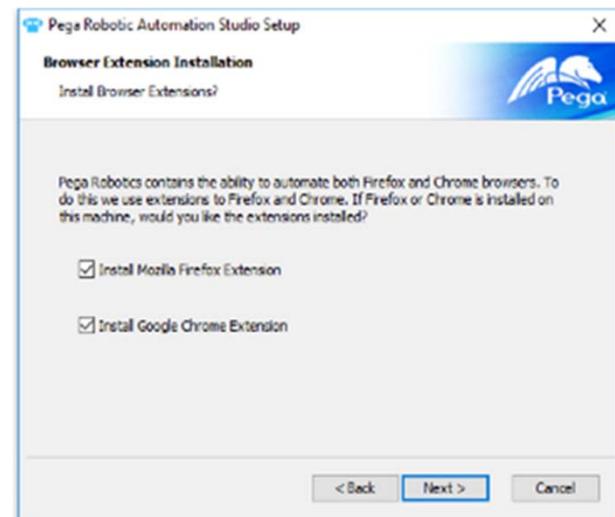
Pega Robot Studio works with only specific versions of each browser.



Installation

To develop a robotic solution and run a deployed end-user solution the extension must be installed and enabled for the Chrome and Firefox browser.

During the installation of both the Pega Robot Studio and Runtime, you are prompted to enable Firefox and Google Chrome extension.



Installation (continued)

If Pega Robot Studio and Pega Robot Runtime are installed initially without the Chrome/Firefox extension:

- Uninstall Pega Robot Studio and Pega Robot Runtime and reinstall Pega Robot Studio Studio and Runtime, electing the Chrome/Firefox extension within the install wizard.
- Contact Support for more detailed instructions
- Installation requires administrator privileges on development and user machines to install the extension correctly.

Properties

You can select to work on different browsers by configuring the BrowserType property after adding the Universal Web Adapter project item to your project.

BrowserType: This property determines the browser used for automating the web application.

Depending on the browser type selected, you need to configure browser specific properties.

StartonProjectStart and Startpage properties are shared between Universal Web adapters and Web adapters.

Chrome specific properties

- AcceptedDomains
 - This property specifies the list of domains that the adapter accepts.
 - An empty list means that all domains are accepted.
 - You cannot integrate any documents you exclude through this filter property, nor will those documents match at runtime.
- BrowserStopMethod
 - This property determines which tabs stop in the browser when the adapter stops.

Firefox specific properties

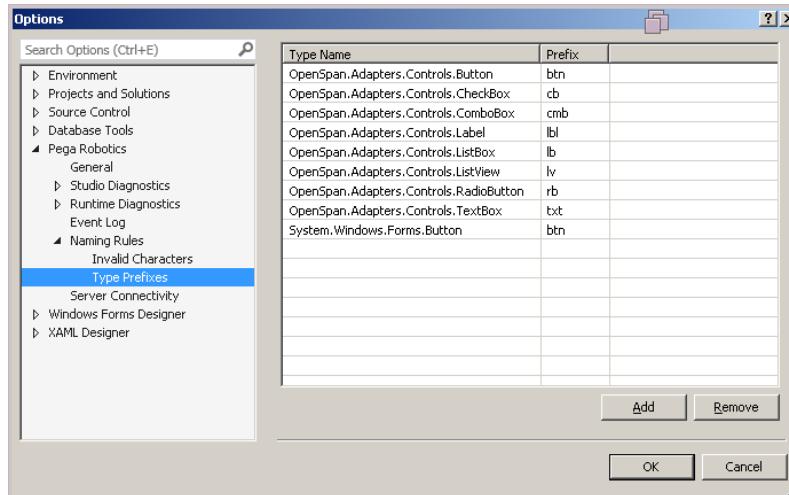
- **StopMethod**
 - This property sets how Pega Robot Studio stops the application when the Stop method is called on the adapter.
- **CloseTimeout**
 - It specifies the amount of time, in milliseconds, Studio waits for the process to exit after the Stop method has been called.
- **Terminate Timeout**
 - This value sets the amount of time Studio waits for the process to terminate in milliseconds.

Limitations

- Plug-ins, such as Java and Flash, are not supported by Universal Web adapter.
- You can interrogate cells within simple tables only. Complex tables cannot be matched based on its table schema.

Modifying the Properties of an Object

Establishing a standardized naming convention for controls and components is important and a Pega Robot Studio recommended practice.



© 2019 Pegasystems Inc.

27



DEMO

How to Add and Configure a Web Adapter



MODULE Basic Interrogation

This module contains the following lessons:

- Overview of interrogation
- Web application interrogation
- Global web pages
- Other integration techniques

Pega Robot Studio™ provides basic methods to interrogate the most common applications used in a robotic solution, a Windows platform, and a web platform.

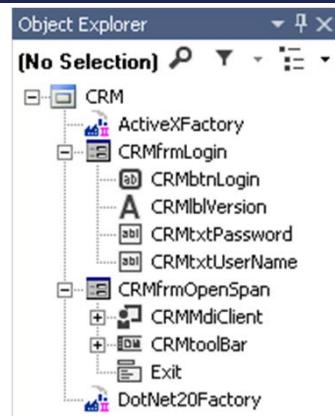
After this lesson, you should be able to:

- Differentiate between the elements created or used during interrogation

Interrogation

Interrogation is a technique that exposes the underlying objects of the enterprise application and uniquely identifies each object.

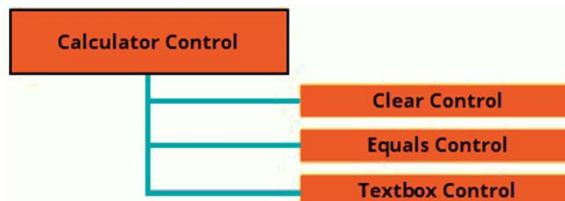
Interrogation results in the creation of controls corresponding to application objects. Pega Robot Studio creates a representation of the object in the Object Explorer.



Controls

A **control** is a representation of an application object created by interrogating the object. Pega Robot Studio organizes controls in a hierarchy.

For example, interrogating the Clear and Equals buttons and the Results text box on a Calculator application results in the following hierarchy:



Controls

A control performs the following functions:

- Saves the rules that are necessary to accurately identify the object for the next time the application runs
- Acts as a proxy for the object so that the object can be used in automations even when the user object has not been created within the target application
- Provides a consistent interface for the object regardless of the underlying application technology

Targets

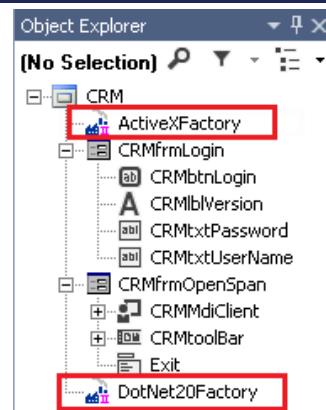
A **target** is a Pega Robot Studio object that corresponds directly with an application object.

A control represents the virtual object that does not change with different instances of the application. A target represents the actual application object that exists within the application at any given moment.

Factories

Factories, along with translators, enable communication between application targets and their platforms.

During interrogation, Pega Robot Studio automatically adds the required factories. They appear in the Object Explorer. They indicate successful interaction between Pega Robot Studio and the application platform.



Translators

A **translator** is a unit of code that lets Pega Robot Studio automate and receive events from an application object.

Targets communicate directly with translators. Pega Robot Studio injects translators into a target application and attaches translators to the appropriate application object.

Translators do not require any modifications or recompilation of the target application. Translators are in the same language as objects with which they interact.

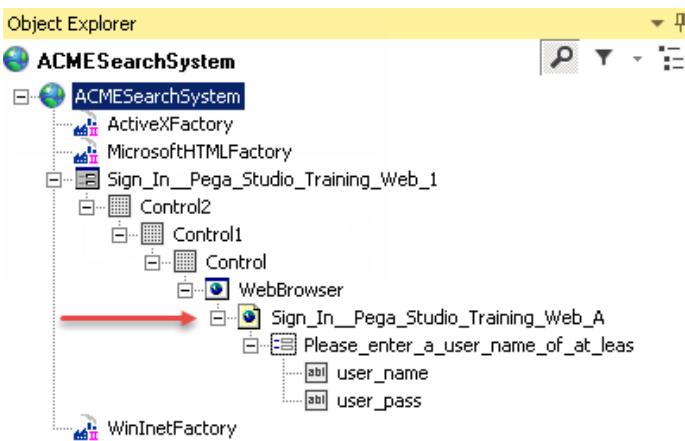
Best Practices for Interrogation

- Interrogate all controls needed for a business case.
- Method-test all objects to ensure that the automation can interact with them.
- Use interrogation to provide developer access to the applications used in the project.
- Used interrogation to perform a preliminary interrogation of the application controls.
- Provide developer access to give an advance look into possible issues with interrogation that may require adjustments to the interrogation process, or even the project itself, to meet the specific requirements.
- Find issues first to reduce development time before developing the solution.

Web Application Interrogation

Web application interrogation is similar to windows application interrogation.

Without modifying the web page source code, Pega Robot Studio evaluates an object's properties against **match rules** to ensure the object is a unique control every time the application runs within the solution.



After this lesson, you should be able to:

- Restate the purpose of Global Web Pages during interrogation

Global Web Pages

Global web page is a function of Studio that allows the interrogation process to ignore all window based objects of the browser and interrogate only the web-based objects. This functionality arises from two reasons:

- Internet Explorer changes between versions 7 and 8.
- Use of web page frames.

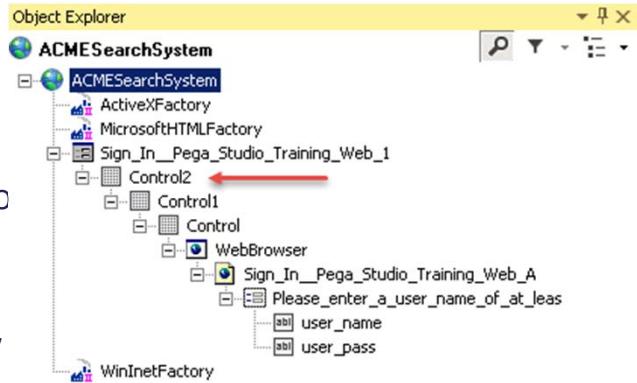
The Create Global Web Page function is the default option when starting the web interrogation process.

Internet Explorer Versions

With the release of Internet Explorer version 8, a new feature became available — tabs. No longer did users have to launch a new browser window to surf a second website.

The new tab structure caused issues with existing solutions when a client upgraded to the new browser version.

Using the Create Global Web Page function, you can make the Object Hierarchy and its objects browser version independent of one another.



Web Page Frames

Another feature used often on web pages is multiple frames. Multiple frames is a structure that allows a developer to use one web page to display other website pages all on one page. The following image shows the multiple frame web page.

Pega Robotic Training Product List - Beverages
[Go To Seasonings](#)

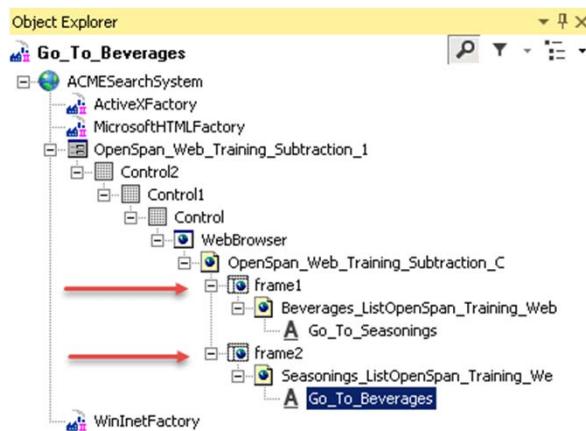
Product	Price	Product ID
Chai	18.00	1
Chang	19.00	2
Guarana Fantastica	4.50	24
Sasquatch Ale	14.00	34
Steeleye Stout	18.00	35

Pega Robotic Training Product List - Seasonings
[Go To Beverages](#)

Product	Price	Product ID
Aniseed Syrup	10.00	3
Chef Anton's Cajun Seasoning	22.00	4
Chef Anton's Gumbo Mix	21.35	5
Grandma's Boysenberry Spread	25.00	6
Northwoods Cranberry Sauce	40.00	8

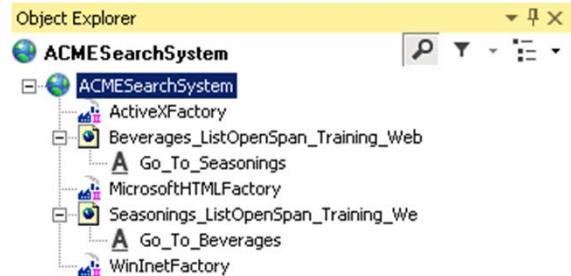
Web Page Frames

Global Web Page disabled



Global Web Page enabled

- Regardless of how users access the pages, they match each time because the abbreviated hierarchy structure only contains the webpage object and its children.



After this lesson, you should be able to:

- Restate why and when to use Select Element to interrogate
- Describe the Create Control method of interrogating

Other Interrogation Techniques

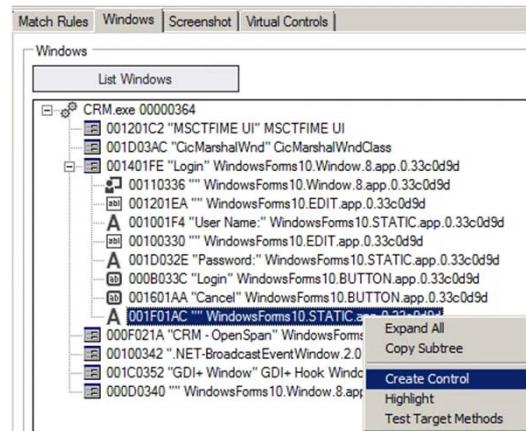
Because application development is inconsistent, you can interrogate an application in three other ways if the standard method does not work.

You can:

- Use Create Control
- Add Menu Items
- Use Web Controls

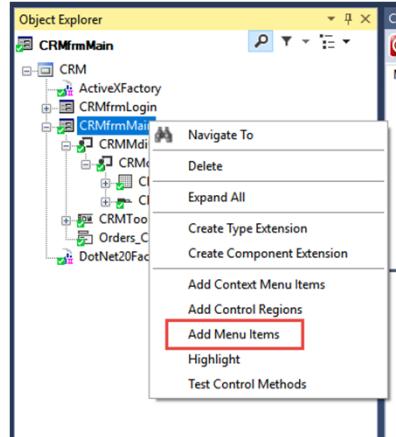
Using Create Control

You may encounter a target that is visible but positioned so the Target icon cannot focus on the object.



Using Add Menu Items

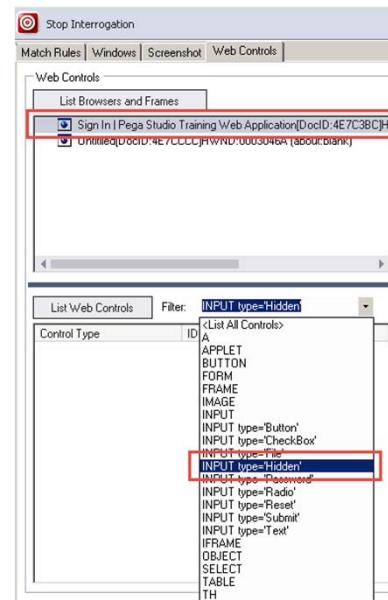
There are times that the business case requires a menu option for the solution. Using the interrogation target disables the menu options on the application window.



Using Web Controls

Use the **Web Controls tab** on the Adapter Design window to locate the hidden object and add it to your solution.

The Web Controls tab also provides a convenient way to list all web targets from a page currently open and matched in the interrogator.





DEMO

How To Interrogate a Web Application



MODULE Match Rules

This module contains the following lessons:

- Overview of match rules
- Web-based match rules
- Universal Web adapter match rules
- Common match rule issues
- Adapter match rules tab
- Modifying match rules

Pega Robot Studio™ provides an advanced matching system that uniquely identifies application interface elements across multiple instances of an application using a set of rules that capture the necessary persistent data.

After this lesson, you should be able to:

- Restate how match rules are used during the execution of a robotic solution.

Match Rules

Every time an application runs a new user interface, the underlying program renders the interface.

To automate an application, a developer must identify an application interface element across multiple application instances, just as a user would.

Additionally, the developer must distinguish between unchanging or persistent data used to identify an element and the changing or transient data the automation should ignore.

Match Rules (continued)

Developers can customize matching behavior by adding, removing, or modifying these rules.

At runtime, as applications run and the targets within the applications display on and remove from the user interface, Pega Robot Studio dynamically applies the match rules for the controls you interrogated. If a target is not currently displaying, then the match rules for the target have failed.

For web applications, the persistent data available to identify an element or control are the HTML tags. The tags provide a standardized format that allows Studio to match the element consistently.

After this lesson, you should be able to:

- Differentiate between web-based match rules applied during interrogation

Web-Based Match Rules

The following is a sample of the web object default match rules.

Object Name		
Anchor	Image	Table
<ul style="list-style-type: none">• Anchor Target• Anchor URL	<ul style="list-style-type: none">• Image Alternate Text• Image Name• Image Source	<ul style="list-style-type: none">• Table Border• Table Height• Table Width
Form	Input	Table Data Cell (TD)
<ul style="list-style-type: none">• Form Method• Form Name• Form Target• Frame Name• Frame Source	<ul style="list-style-type: none">• Image Button Alternate Text• Image Button Source• Input Index• Input Name• Input Size• Input Type• Input Text	<ul style="list-style-type: none">• Table Cell Column Index• Table Cell Height• Table Cell Padding• Table Cell Row• Table Cell Row Index• Table Cell Spacing• Table Cell Width

Web-Based Match Rules (continued)

If Pega Robot Studio is unable to match the control based solely on the web object rules, it uses the following standard default match rules:

- Attribute Value
- Control Children
- Element ID
- Element Index
- Element Inner Text
- Element Path

Element ID

The most frequently used default match rule is the **Element ID** match rule. HTML designates every object on the web page with a unique identifier; as a result, Pega Robot Studio uses this rule the most.

If a developer lets the web page assign the element ID for all objects, the web page creates a new element ID for the objects each time the pages load. Therefore, on the initial interrogation, the Element ID rule may have an initial value. When testing the solution, the object fails to match when the application runs because the element ID is now different.

Attribute Value Match Rule

All HTML tags have predefined properties or attributes for a developer to use to distinguish one similar tag from another. Because web developers can decide which attributes to use, when to use the attribute, as well as what values to use for the attributes, a Pega Robot Studio developer can use the **Attribute Value** match rule to define a unique attribute and value.

Element Index Match Rule

The **Element Index** match rule generates the index of the HTML tag of the control relative to the other tabs of the same type.

The index numbering starts at zero.

- If three <p> tags exist on the web page, the first tag has an index of 0, the second of 1, and the third of 2. Use this rule sparingly and as a last quick fix. Web pages are too dynamic, and the index can change too often.

Element Path Match Rule

The **Element Path** match rule is the routing of the HTML tags surrounding the interrogated object. Using the Select Element function during interrogation displays the element path of the interrogated object on the context menu.

The process of using match rules to identify objects is the same for Universal Web adapters and Web adapters. The difference is the number and scope of Universal Web match rules are different than web-based default match rules.

After this lesson, you should be able to:

- Recognize the three match rules for Universal Web Adapters
- Differentiate how web match rules are used between Universal web adapters and standard web adapters

Universal Web Adapter Match Rules

The Universal Web adapter match rules use three types of match rules during interrogation.

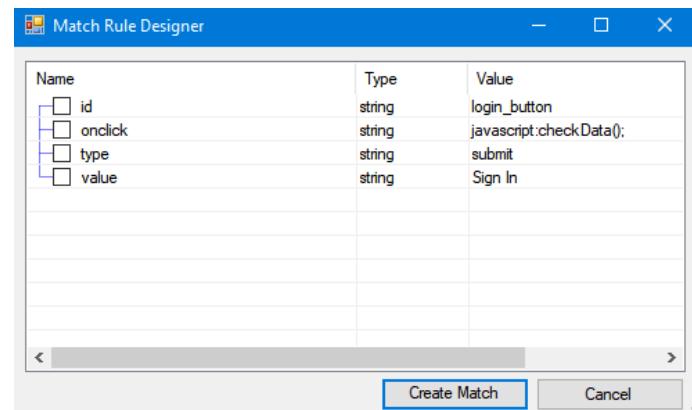
The three Universal Web adapter match rules are:

- Attribute Value match rule
- Property match rule
- Control Children match rule

Attribute Value Match Rule

All HTML tags have predefined attributes for a developer to use and distinguish one similar tag from another. Web developers decide which attributes to use, when to use the attribute, as well as what values to use for the attributes.

A Pega Robot Studio developer can use the **Attribute Value** match rule to define a unique attribute and value.



Property Match Rule

When a developer implements a specific control in a web application, those controls have inherited properties from the class or object that can differentiate from one control to the next.

These properties use the **Property** match rule to expose those inherit control properties and to identify an interrogated object. Properties from parents and children can define a control.

Name	Type	Value
ATTRIBUTE_NODE	number	2
attributes	object	
attributeStyleMap	object	
autofocus	boolean	false
baseURI	string	http://training...span.com/log
canHaveChildren	boolean	false
CDATA_SECTION_NODE	number	4
checked	boolean	false
childElementCount	number	0
childNodes	object	
children	object	
classList	object	
clientHeight	number	28
clientLeft	number	1
clientTop	number	1
clientWidth	number	68
COMMENT_NODE	number	8
contentEditable	string	inherit
defaultChecked	boolean	false
defaultValue	string	Sign In
disabled	boolean	false
DOCUMENT_FRAGMENT_NODE	number	11
DOCUMENT_NODE	number	9
DOCUMENT_POSITION_CONTAINER	number	16

Control Children Match Rule

The **Control Children** match rule uses the matching criteria of a subordinate object (child control) in Object Explorer to designate that the parent object is matched.

The rule matches the children objects first. When the children are matched, then the Control panel is matched. The Control Children match rule may be used between the parent and child. The parent and child do not have hierarchy levels between them.

After this lesson, you should be able to:

- Define the two common matching mistakes
- Restate how controls can become unmatched

Ambiguous Matching

A common match rule issue is ambiguous matching. Ambiguous matching occurs when the default match rules fail to identify a single target.

This failure usually occurs because the information available about the target is insufficient to differentiate it from other targets in the application or more than one target share the same information.



Ambiguous Matching: Generic Controls

Pega Robot Studio makes every attempt to identify each target.

For example, when you interrogate a text box in a Windows application, Pega Robot Studio tries to identify the object as a text box with all of its inherent properties, methods, and events so you can use the object fully in automations.

In some cases, Pega Robot Studio can define the targets only as a base Windows or generic control. These controls have a base set of properties, methods, and events that are often a subset of the full functionality available within the application.

Ambiguous Matching: Button Class

When interrogating web applications, HTML classifies most buttons as an <input> object.

So regardless of the button, radio button, or check box, the <input> is the only defining item.

The only difference may be a text or ID value of the object to differentiate.

Ways to Resolve Matching Issues

Pega Robot Studio provides two methods of helping to help resolve matching issues:

- Debug Matching
- Replace Control

Both items are found in the drop-down box on the Interrogation Form during interrogation.

To monitor and customize match rules that either fail or unmatch, Pega Robot Studio provides the Match Rules tab.

After this lesson, you should be able to:

- Recognize the working panes and their purpose on the Match Rules tab

Targets

While interrogating an application, the Targets pane lists all the targets that are similar to the control highlighted in Object Explorer. The Target pane provides the properties of each target.

Targets				
Tag Name	Text	Name	Value	Attached To
<input checked="" type="checkbox"/> INPUT		user_name		ACMETxtUserName

View Target Properties

The following items are listed for each target.

Properties – Target properties used in applied match rules. (In the following example of a web application interrogation, you see the Tag Name, Text, Name, and Value properties.)

Attached To – Target name associated with the control in the Object Explorer.

Sort and Rematch Targets

Use **Target** function buttons to sort and rematch the targets. The buttons allow for the comparison of similar targets to determine which target requires match rule editing to uniquely identify the control.

Target function button	Name	Description
	Show Matched	List only the matched objects for the selected match rule. This is the default option.
	Show Not Matched	Show unmatched targets.
	Show All	Show both matched and unmatched targets
	Refresh Matching	Refresh the list to either determine other matching targets or apply new match rules and values to a selected target.

Selected Targets

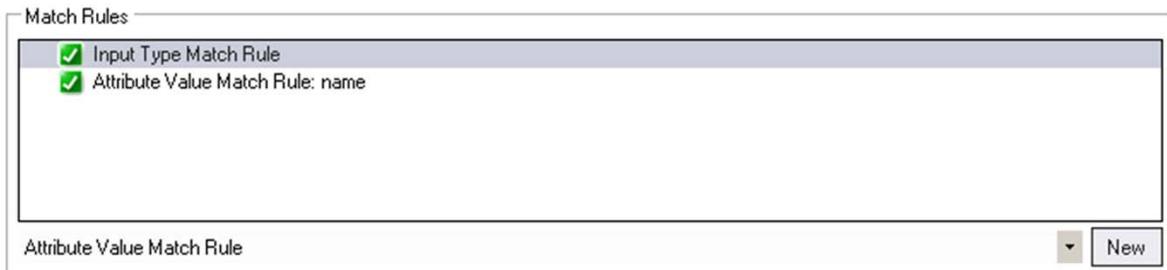
The **Selected Target** pane displays the properties of the highlighted target in the Targets grid.

You can use the properties to compare Match Rule properties to determine how to match a control. You can also test the interrogated control by changing the properties in the Selected Target pane.

Selected Target		
AlternateText	True	
CanHaveChildren	False	
Checked	False	1, 1, 332, 32
ClientRectangle		
DefaultChecked		
DefaultValue		
Duration	0	
Enabled	True	
Height	0	
HorizontalSpace		
InnerText		
IsAttached	True	
LogicalContainer		
MaxLength		2147483647
Name		user_name
NativeTypeName		INPUT
ObjectId		HTMLELEMENT(DEPRECATED)
OffsetRectangle		10, 10, 334, 34
Enabled		Indicates if the user can interact with the element.

Match Rules

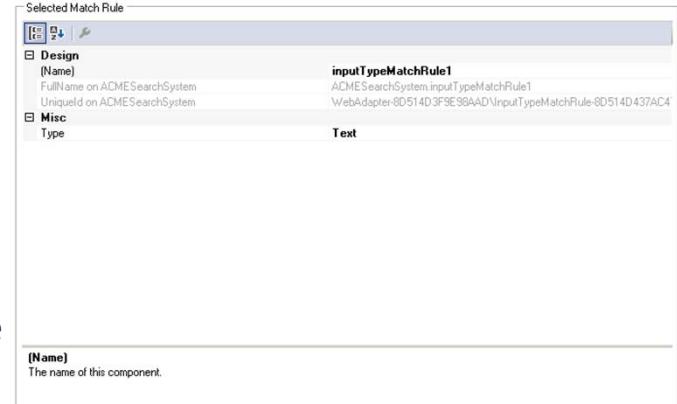
For a selected target in Targets pane, the Match Rules pane lists the match rules the system uses to identify the object.



Selected Match Rule

The **Selected Match Rule** pane displays the properties of the selected match rule. You can check and edit match rule properties and values. Once entered, the targets requires a refresh to determine a successful match.

Match rules automatically determine their needed property value and update the rule and target. Other match rules require user input to identify the property value.



Due to the dynamic nature of web applications, rules can fail or change without notice, which creates more opportunities to modify and update match rules.

After this lesson, you should be able to:

- Add and edit a match rule for an interrogated control



DEMO

How To Add/Remove a Match Rule

Using a Match Rule Editor

Often, Pega Robot Studio uses match rules that validate against any text associated with the object.

Windows forms and web pages may contain text that changes with each user or each account accessed.

This dynamic text causes issues because the match rule considers the text static or persistent text. You must modify text-based match rules to eliminate dynamic text and evaluate the object solely on the static text.

Pega Robot Studio provides a Match Rule Editor window to perform this task.

Web Page Frames

The Match Rule Editor consists of the following options:

Property	Description
Text	The label text of a target object, such as the window title bar text or a control button label. For the sample CRM.exe application, the Login window text is Login . Modify the text to support the Mode selection.
Culture	Language (culture) used by target application users. The default is User Culture, which uses the operating system setting (see region and language options under Control Panel).
IgnoreCase	The default is True , which indicates that text can appear in either upper or lowercase.
Mode	The default is Simple , which means the text must match literally. Available mode types include Regex, StartsWith, and EndsWith. Regex mode indicates that the Text property contains Regex syntax. Select the mode in conjunction with the Text property value.



DEMO

How To Remove an Element ID Match Rule Search



MODULE Basic Robotic Automations

This module contains the following lessons:

- Automation development
- Object explorer and automations
- MessageDialog
- Types of automation links
- Managing automation workflow
- Automations as procedures
- Label/Jump To component

LESSON

Automation Development

In terms of developing a robotic solution for attended or unattended robots, developing automations is the last basic concept in Pega Robot Studio. You develop automations to execute from a specific object event or called as a procedure to execute when needed.

After this lesson, you should be able to:

- Implement message dialogs within an automation for debugging or user guidance

Automation Development

Automations are the backbone of Pega Robot Studio™.

Automations convert a user's manual processes to consistent and repetitive operations. Because manual processes drive the development of automations, user actions or application actions make automation development into event-driven project items.

When attempting to mirror or automate a manual process, pay attention to how users interact with the application as well as how the application responds to the process.

LESSON

Object Explorer and Automations

Automation development drives its structure from the Object Explorer. The Object Explorer displays interrogated application controls as well as .NET controls added to a windows form.

After this lesson, you should be able to:

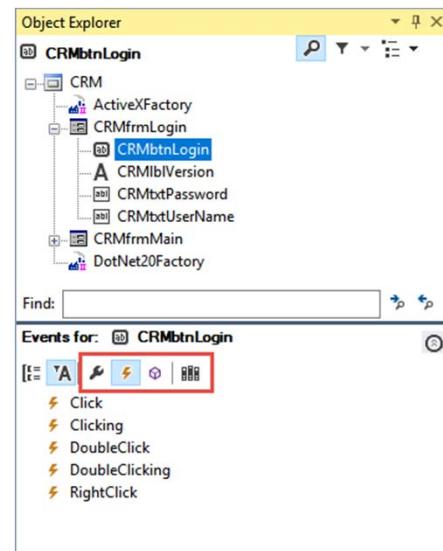
- Describe the relationship between the Object Explorer and automations
- Implement the Object Explorer in automation development

Object Inspector

The Object Inspector provides access to the object's properties, events, and methods.

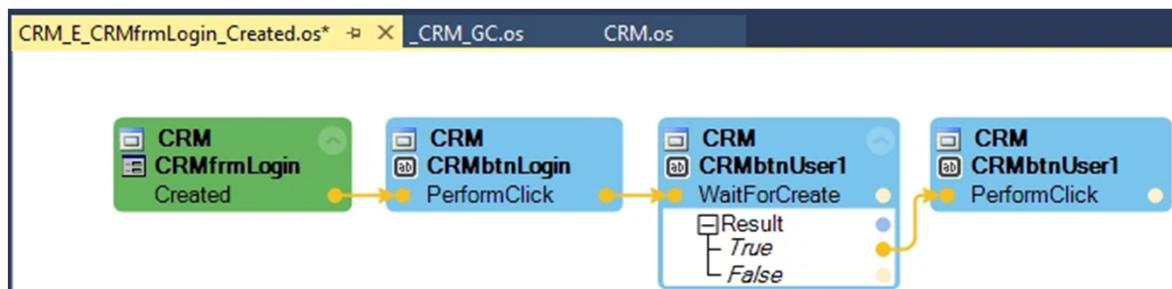
In the Object Inspector image, the three buttons — **Show Properties Only** (wrench), **Show Events Only** (lightning bolt), and **Show Methods Only** (box) — display an object's properties, events, and methods when clicked.

The fourth button, **Configure type** (books), provides access to a library of all properties, events, and methods for a specific object type.



Automations

Automations usually begin with an object's event, meaning that the automation does not begin until a window is displayed or a user clicks a button. Design blocks are the items added to an automation.



Target Creation

When creating automations you must consider is the amount of time a web page spends to load or the wait time for an application to respond. For instance, some web applications load quickly, while others do not.

Numerous variables can cause this variance, such as time of day, number of users, and background application processing.

Created Event

When Pega Robot Studio matches a target, the Created event for the object triggers. For example, in a CRM_E_CRMfrmLogin automation, when the CRM Login is fully loaded, a Created event is triggered.

Remember that the Created event for any object is triggered when the object matches, meaning that navigating back to a previous page or window retriggers the Created event for all interrogated objects on that page or window.

IsCreated Property

This property is False until the object's Created event triggers. Returning to the example of the CRM_E_CRMfrmLogin automation, the CRMfrmLogin window **IsCreated** property turns True when it matches the Studio match rules for the object.

When you click the Login button, the window un-matches, and the IsCreated property returns to False. The IsCreated property is one property that a developer cannot modify through an automation.

WaitForCreate Method

The **WaitForCreate** method sets up automation logic to wait until the Created event triggers for an object before proceeding to the next step of the automation.

The method provides a Boolean result of the Created event. If the Created event occurs within the wait time, the automation continues through the True logic. If it does not occur within the wait time, the automation continues through the False logic.

If the WaitForCreate method is not used, all objects have an implicit wait of 30000 milliseconds (30 seconds) by default.



DEMO

How To Add Properties, Methods, and Events to an
Automation



DEMO

How To Add Properties, Methods, Events to the Object
Inspector



DEMO

How To Access the Component Properties



DEMO

How To Create an Automation

MessageDialog is a component that displays a message box with a given text. MessageDialog can display useful information to the user during the process execution.

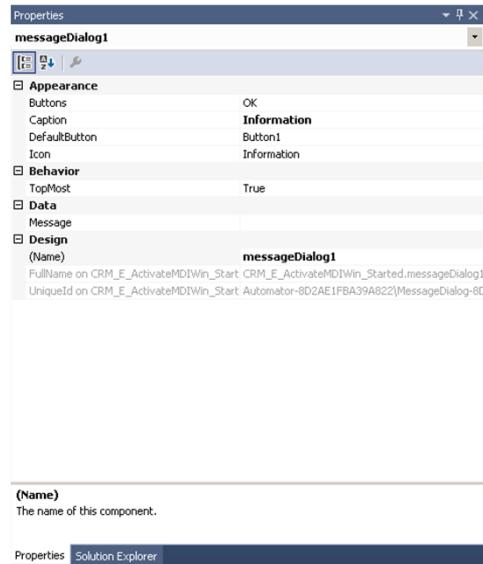
After this lesson, you should be able to:

- Implement message dialogs within an automation for debugging or user guidance

MessageDialog

You can determine the design factors of a MessageDialog such as the style of dialog, button, or text appearance on the dialog box.

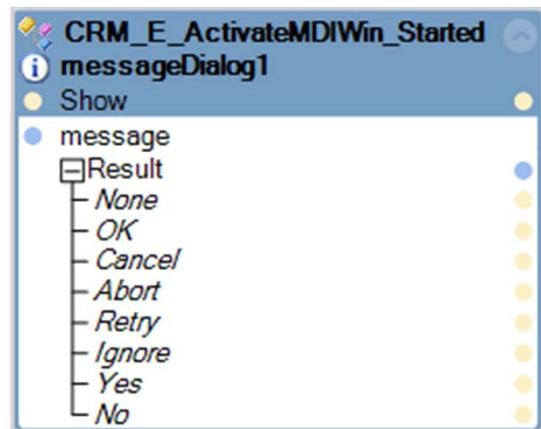
You can choose the execution path based on the option selected from the MessageDialog box options. You can provide various options to the user through the **Buttons** property.



MessageDialog (continued)

MessageDialog provides error information to the end-user along with guidance. During the development of an automation, MessageDialog assists as a debugging tool.

The MessageDialog component is located in the **Advanced** tab of the Toolbox. Click and drag the MessageDialog to the automation to add the component to an automation.



MessageDialog (continued)

The MessageDialog component provides methods with different parameters. You can choose to use any of them depending on the dynamic content you want to display.

Method	Description	Parameters
Show (no params)	Displays the MessageDialog. No dynamic input parameters. The message, buttons, and caption are set through the properties on the Properties window.	None
Show (1 param)	Displays the MessageDialog. Enables the dynamic input of the message.	String message
Show (2 params)	Displays the MessageDialog. Enables the dynamic input of the message and caption.	String message, String caption
Show (3 params)	Displays the Message Dialog. Enables the dynamic input of the message, caption, and owner of the window. To use the window owner parameter, connect the output of the parent window to this parameter.	String message, String caption, Win32Window owner

Types of Automation Links

Pega Robot Studio projects use threads to paint the screen with the project's windows, application bars, Windows forms, the associated controls, execute automations, and integrate with the project applications. At project runtime, when users interact with a Windows form or its child controls, Windows sends messages describing the user's activity to the user interface thread.



Managing Automation Workflow

When automating a process that incorporates a web or Windows application, it is recommended to perform a check to determine the user's location within the application before attempting to perform or begin another process.

A best practice is to use a consistent starting point in the application to begin the process. Typically, this is the home page or landing window in the application. Using a consistent starting point allows you to create a reusable function automation that you can call at any time you need to begin a new process.

You should not create an automation that initiates on a page or window created or a control created event. At any point in time when navigating an application, users may mistakenly initiate an automation because the automation runs when the page or window matches. To avoid users from mistakenly initiating an automation, it is recommended to have the automations check the user's location prior to starting the process within the application.

While developing automations, if a logical process is reusable, create the automation process as a procedure that you call or execute when needed. You may declare parameters to pass into the procedure as well as pass parameters out of the procedure.

After this lesson, you should be able to:

- Describe how automations are used with other automations and the solution workflow

Best Practices

- Use the following naming convention: **Adapter Shorthand name + P + DescriptiveName**
- Keep procedure automations in a Procedures folder in each project.
- Procedures must have an Entry Point and should have at least one Exit Point. Multiple Exit points should be used to terminate all potential end points in the execution path.
- Exit points should be labeled such that they make logical sense, for example, Success, Failure, and NoResults.

Best Practices

- Jump labels can be used to consolidate some of the Exit paths if they are related (for example, Success/Failure).
- Procedures must return a string value indicating any error messages that may have occurred during the execution. Use Message Manifest to manage and maintain the solution messages.
- Procedures should return any values that may be needed elsewhere in the application with the Exit Point.

While creating an automation, you may find that you want the automation to complete many processes and tasks. However, it is important that you keep the number of tasks and processes that you want the automation to complete to a minimum.

After this lesson, you should be able to:

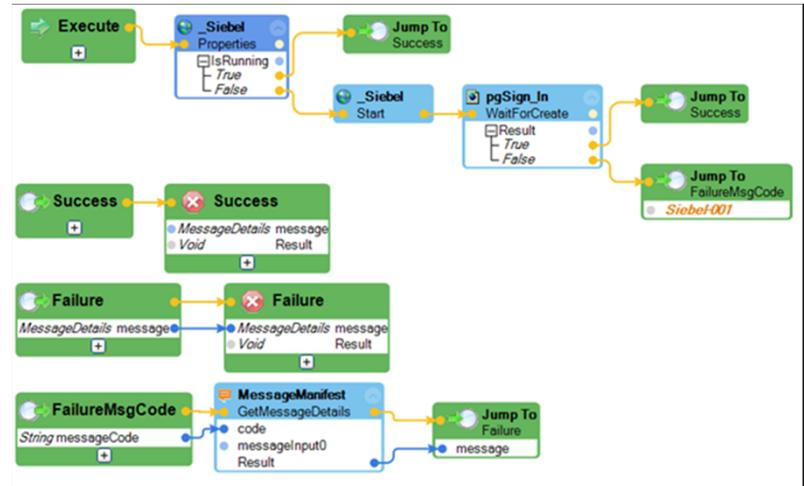
- Create an automation as a procedure using the Label/ Jump To components

Label/Jump To

- The **Label - Jump To** component lets you connect one part of an automation to another part of the same automation.
- The Label/Jump To component cannot navigate the process to another automation. For example, if you have a very large automation in which you need to connect a variable and method, instead of a long execution line or data path, you can connect them with a Label-Jump To pair.

Label/Jump To (continued)

- The procedure automation provides success, failure, and failure message code exit points by using the Label/Jump To components to pass the message code and message details.





DEMO

How To Create a Procedure Automation



MODULE Debugging

This module contains the following lessons:

- Overview of debugging
- Debugging automation data values

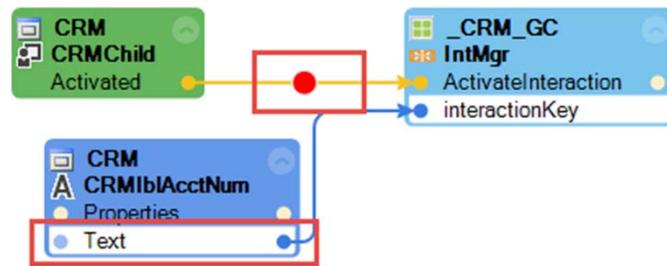
Regardless of the programming and developing environment, developers use debugging concepts to test and rework the program continually.

After this lesson, you should be able to:

- Recognize the reasons for using debugging for Pega Robot Studio
- Start and stop the debugging process

Debugging

Because Pega Robot Studio™ uses Microsoft Visual Studio as its platform, the concept of debugging is similar. There are some slight variations with its implementation.





DEMO

How To Use Breakpoints

LESSON

Debugging Automation Data Values

The Pega Robot Studio debugging feature provides two main windows that allow you to track and monitor data values during the debugging process.

After this lesson, you should be able to:

- Recognize the difference of automation locals and watches.

Automation Locals

The automation Locals window displays values and types of all data ports in the active automation.

- **Access**
 - From the main menu, select **Debug > Windows > Automation Locals** to open the Automation Locals window.
 - Your automation must include at least one breakpoint.

Automation Locals

- **Updates**

- Automation locals are updated as you Step Into, Step Over, or move to each breakpoint.
- You can also use the Automation Locals window to navigate while the debugger is active.
- Double-clicking a data value line in the Automation Locals window highlights the automation design block.

Automation Locals

- **Data variables**
 - The Automation Locals window changes the list of data variables depending on the active thread, stack, or active automation during the debugging session.
 - In the value column, you can select an item and edit the value to test different scenarios.
- **Multiple threads**
 - You can run multiple threads during the debugging session; the Automation Locals window updates the displayed variables depending on the active thread.

Automation Watches

The Automation watches window displays data for selected data ports on any project automation.

- **Access**

- From the main menu, select **Debug > Windows > Automation Watches** to open the Automation Watches window.
- Your automation must include at least one watch to view the Automation watches.
- You select which objects to monitor to view runtime values and types.

Automation Watches

- **Updates**
 - Double-clicking a data value line in the Automation Watches window highlights the automation design block of the value.
 - In the value column, you can edit a value of a value item to test different scenarios.
 - You cannot delete or remove a data watch once added to a blue data port.
- **Data variables**
 - On a design block, right-click a blue data port and Select Add Watch to designate the Watch ports in the automation.

Automation Watches

- Multiple threads
 - The Automation Watches window focuses on the designated data ports and values. Activity is not impacted by the current thread or active automation window. The window does not update based on thread or active automation.



DEMO

How To Check Data Values



MODULE Diagnostics

This module contains the following lessons:

- Diagnostic settings
- Modifying diagnostic settings
- Diagnostic logging
- Pega Robotic Automation Playback
- Log file cleaner

Because Pega Robot Studio™ uses Microsoft Visual Studio as its platform, the diagnostic components for Pega Robot Studio are similar.

After this lesson, you should be able to:

- Update the diagnostic settings to produce the necessary diagnostic log files
- Add a diagnostic log component to an automation to include specific text in a log file

Diagnostic Settings

Within Pega Robot Studio, you can control the diagnostic settings for messaging for both Pega Robot Studio, during development, and Runtime, during debugging.

The diagnostic settings are stored in the StudioConfig.xml. The Runtime settings are stored in the RuntimeConfig.xml.

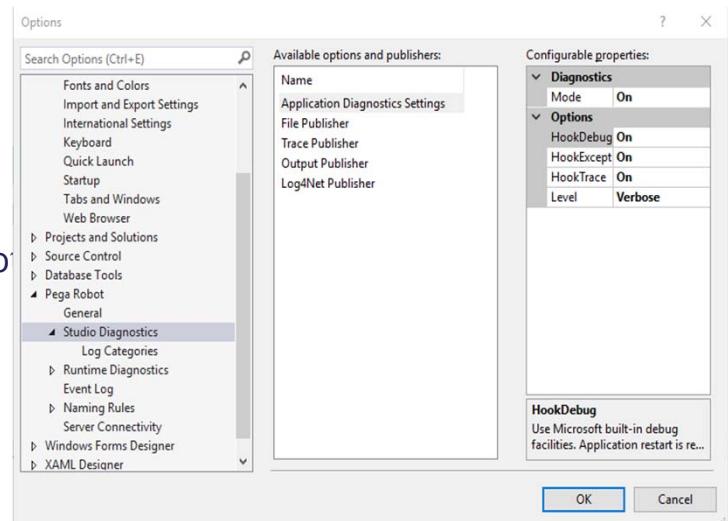
The diagnostic settings consist of two parts:

- Diagnostic message generation
- Publisher Options

Diagnostic Message Generation

The general category of Application Diagnostic Settings controls the message generation.

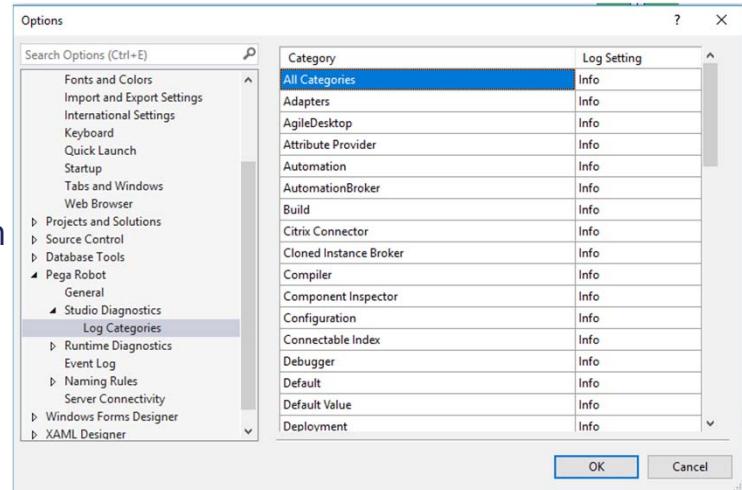
The settings available for Pega Robot Studio are the same as the Pega Robot Runtime settings.



Log Categories

Log Categories, located under the Studio Diagnostics and Runtime Diagnostics levels, provide a more granular level to set error levels.

Set the error level on the Application Diagnostic Settings to Verbose to ensure that you write all messages regardless of the Log Category error level.



Publisher Options

For both Pega Robot Studio Diagnostics and Pega Robot Runtime Diagnostics, there are four publisher types:

- File
- Trace
- Output
- Log4Net

The Trace Publisher enables the generation of diagnostic traces for detailed application messages for Pega Robot Studio and Pega Robot Runtime. The Output Publisher displays the diagnostic messages to an Output window in Pega Robot Studio or Pega Robot Runtime.



DEMO

How To Modify Diagnostic Settings

LESSON

Diagnostic Logging

Pega Robot Studio and Pega Robot Runtime both provide a mechanism for recording events during run time.

After this lesson, you should be able to:

- Dissect a log file to isolate the necessary log entries for issue resolution
- Use the Automation Playback feature to isolate a log entry for issue resolution
- Clear a log file of specific information to ensure company and account privacy



DEMO

How To Add a Diagnostic Log Component To an Automation



DEMO

How To Dissect a Log File

LESSON

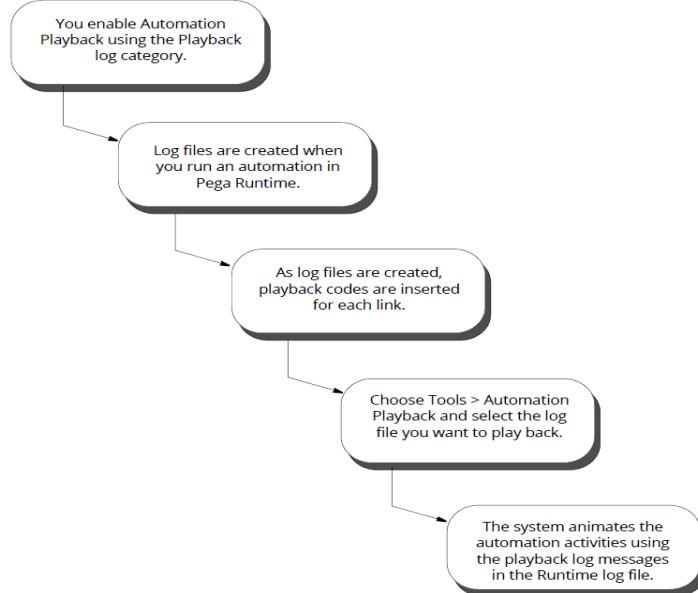
Automation Playback

The Automation Playback feature lets you animate the automation activities in Runtime log files so you can better see what is happening as your automations execute. Automation Playback also works with files generated with the Log4Net Publisher.

After this lesson, you should be able to:

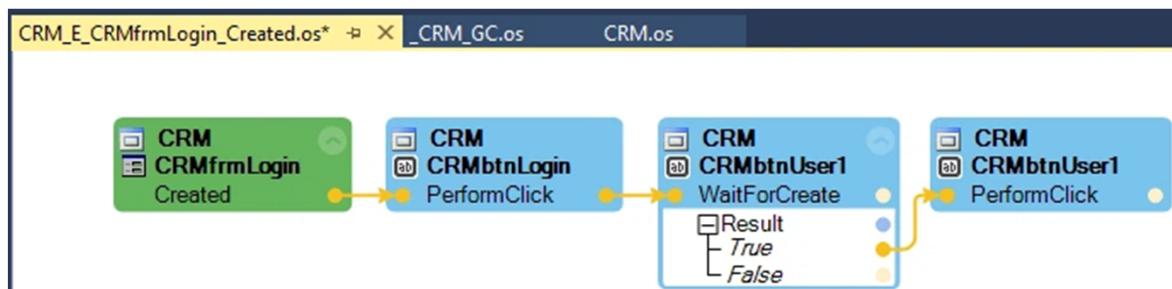
- Use the Automation Playback feature to isolate a log entry for issue resolution

Automation Playback



Automation Playback (continued)

Automations usually begin with an object's event, meaning that the automation does not begin until a window is displayed or a user clicks a button. Design blocks are the items added to an automation.



Playback Window Features

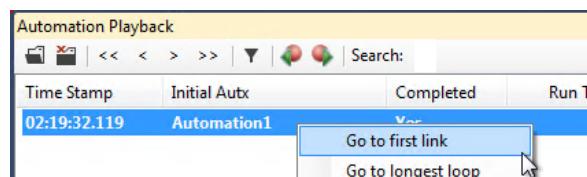
The Automation Playback window is displayed as a designer window in the Pega Robot Studio IDE. You may float, dock, or relocate the window to fit your needs

Column	Description
Time Stamp	This is the time at which Automation Playback starts playing back this link.
Initial Autx	This is the name of the first automation that is run.
Completed	This lets you know if the system completed the processing of an automation and return. Use this status to determine which automation failed.
Run Time (ms)	This is the amount of time, in milliseconds, it takes to process the link.
Links Executed	This is the total number of links executed.
Longest Loop	This is the number of iterations in the longest loop.
Thread Count	This is the number of threads started. Typically, this should be a low number. If the automation spawns a high number of threads, check the automation and reduce the number of threads.
Autx Count	This is the number of automations used within the execution context.

Alternative Navigation Options

Log files can be long and contain thousands of line entries. This depends on the configuration of the diagnostic settings found in the RuntimeConfig.xml file for both Studio and the user's Runtime application.

The Playback feature provides a quick alternative navigation option that allows you to jump to specific points in the log file.



LESSON

Log File Cleaner

Often with the creation of diagnostic log files, Pega Robot Runtime captures personal and confidential information during a solution run. Because of this, you should clean the log file of any information of this type to ensure or follow company confidentiality policies.

After this lesson, you should be able to:

- Clear a log file of specific information to ensure company and account privacy

Log File Cleaner

Pega Robot Studio provides a Log File Cleaner utility to remove personal or account-sensitive data from the log files quickly.

You can use the Log File Cleaner utility with the diagnostic log files generated from Pega Robot Studio, Pega Robot Runtime, Supertrace, and dump files.

The utility examines each line in the log file and either masks or removes the data elements you specify, such as Social Security numbers. You can select to mask or remove on the data elements from a line, or you can mask or remove the entire line.

Installation

The Log File Cleaner utility is not part of Pega Robot Studio installation.

In the Related Content section, click the **Pega Robotic Log File cleaner** link to download the cleaner utility ZIP file.

No installation is required.

Unzip the files to a folder location of your choice and run the executable file: **OpenSpanLogFileCleaner.exe**.



MODULE UseKeys Property

This module contains the following lessons:

- The UseKeys property
- Application of keys

LESSON

The UseKeys Property

Pega Robot Studio™ provides a function that allows a developer to assign a unique key to like objects to differentiate them from each other.

After this lesson, you should be able to:

- Define the implementation of UseKeys logic

UseKeys Property

Pay attention to how an application responds and behaves to user interaction. For example, a user logging into the same application more than once concurrently.

The CRM application produces the same window for all customers, and it allows users to work on more than one customer at a time.

How do you differentiate the accounts so that all applications reference the same account at the same time?

Key Property Values

Pega Robot Studio creates a key property for all interrogated controls and uniquely identifies the object at runtime. The Key property has these two values:

Value	Description
Relative	Pega Robot Studio determines if the value is Active or Null
Absolute	Pega Robot Studio uses the GUID (Global Unique Identifier) of the object

Pega Robot Studio provides you the ability to assign your own key to an object. In order to do this, you must change the UseKey property for the cloneable object.

LESSON

Application of Keys

Pega Robot Studio™ creates and uses keys whenever a context exists for cloneable objects. Here, the term context defines any instance of an object for which multiple instances (clones) occur.

After this lesson, you should be able to:

- Set the UseKeys property on an interrogated object

Keys and Context

A set of rules determines the creation of keys and the setting of keys for contextual objects.

- An event that creates the instance sets the context.
- An event from No Context to a Context requires a key assignment.
- An event from Context to Child Context requires a key assignment.
- An event from Context to Parent Context does not require a key assignment.
- Logic within the same context does not require a key assignment.



DEMO

Setting the UseKeys property



MODULE Automation Techniques

This module contains the following lessons:

- Raising events
- Automation data elements

Depending on the application development, certain events that happen with a user interaction may not occur when automated. In an automation, when moving data into that field, the tab event does not occur and then the evaluation check does not occur either.

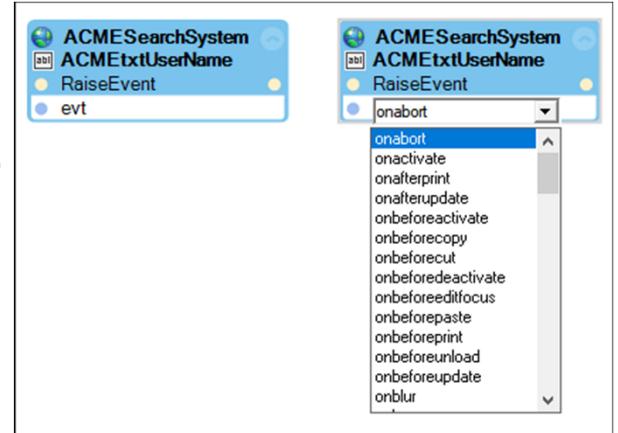
After this lesson, you should be able to:

- Explain the uses of data proxies for automations
- Implement the extraction of data proxies within an automation

RaiseEvent

Pega Robot Studio™ provides a method for some controls that enable you to mimic user interactions in the application.

This **RaiseEvent** method allows you to keep the integrity of the coded application processes while still automating the full process.



© 2019 Pegasystems Inc.

3

From the Object Explorer, highlight the control requiring the event and search for the RaiseEvent method from the Configure Type option in the Object Inspector. After adding the RaiseEvent method for the control to the automation, click the parameter to select the desired user event to mimic.

LESSON

Automation Data Elements

Interrogated items and components added to automations or global contexts appear and are directly available in the Object Explorer.

After this lesson, you should be able to:

- Explain the uses of data proxies for automations
- Implement the extraction of data proxies within an automation

Extracting a Proxy

In Pega Robot Studio, creating proxies within automations meets the concise and legible coding standards.

A **proxy** is an agent of a real object that provides the developer with access to methods to adjust, search, or manipulate the proxy data.

It is often required to use proxies when creating automations incorporating components that contain collections of complex objects such as the PDFComponent, ExcelConnecter, ListViews, and DataTables.

Note that the proxy value is not valid until the automation passes the data port where it was originally extracted.

Using the This Property

The **This** property method is best used within a procedure automation that contains input parameters and allows the developer to minimize the automation design complexity by reducing the crisscrossing of execution and data links throughout an automation.

Note that the proxy value is not valid until the automation passes the data port where it was originally extracted.



DEMO

How To Extract Data Proxy



DEMO

How To Use the **This** Property of a Data Element



DEMO

How To Create a Windows Form



MODULE Toolbox Components

This module contains the following lessons:

- Automation variables
- Expressions and Comparisons
- String, DateTime, and File utilities
- Automation looping
- Using wait logic
- Excel File Connector

LESSON

Automation Variables

Recognizing the library of components along with their functions and methods within the Toolbox is critical for developers to create automations that perform the simplest to most complex tasks required of the project and solution.

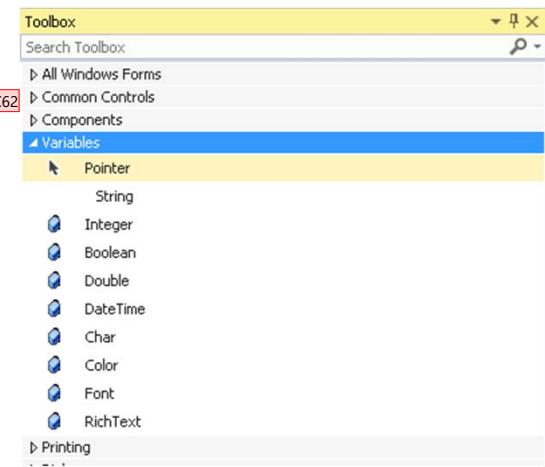
After this lesson, you should be able to:

- Differentiate between local and global as the terms relate to a robotic automation and project
- Describe the use and limitations of the global container project item
- Describe the access to creating variables within a robotic automation

Variables

Pega Robot Studio™ provides many standard variables that can hold different values, including the most common ones of String, Integer, Boolean, Double, Char, and DateTime.

Pega Robot Studio also provides variables that can store color, Rich Text Format, and font. All these variables are similar to .NET variables.



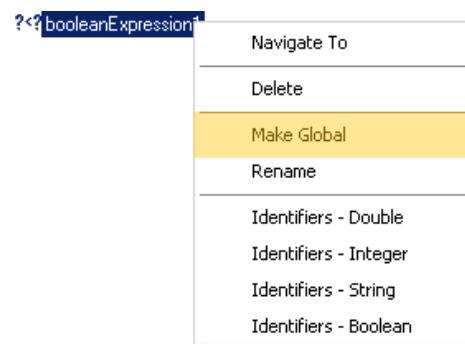
Slide 3

MC62 This can come out here just to make slide concise.
Malcuit, Caitlin, 7/12/2019

Global Variables

Similar to a programming language, you can define the scope of the variable as local or global. Local variables are accessible only in the individual automation.

Global variables are accessible to all the automations throughout a project, but not the solution with multiple projects. Pega Robot Studio provides another mechanism to share variable values across projects.



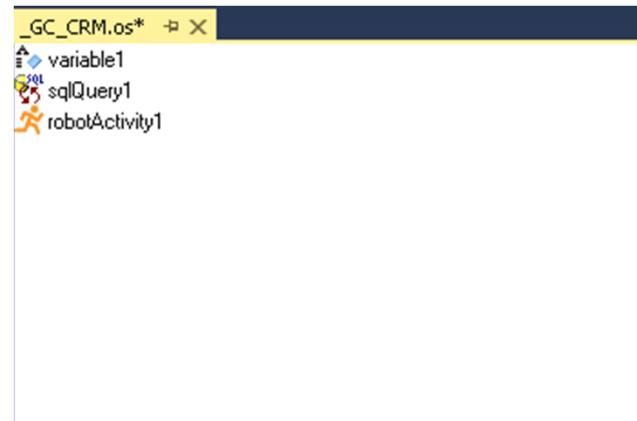
Local Global

You can change the local variable to global by right-clicking on the variable and selecting **Make Global**.

Global Container

It is difficult to manage project-level or Global variables and remember which automation contains them. To avoid such difficulty, Pega Robot Studio provides a project item called **Global Container**.

The Global Container provides a common area to store variables that of global scope. The Global Container can also store Pega Robot Studio components such as SQLQuery, RobotActivity, and so on.





DEMO

How To Add a Component To the Global Container

LESSON

Expressions and Comparisons

Pega Robot Studio allows you to implement expressions and comparisons to drive specific decisioning rules and to create the desired outcomes related in the project requirements.

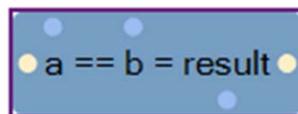
After this lesson, you should be able to:

- Define the three types of comparisons and expressions
- Explain the expression input parameters

BooleanExpressions

The **BooleanExpression** component of Pega Robot Studio allows you to add a logical statement that is either TRUE or FALSE.

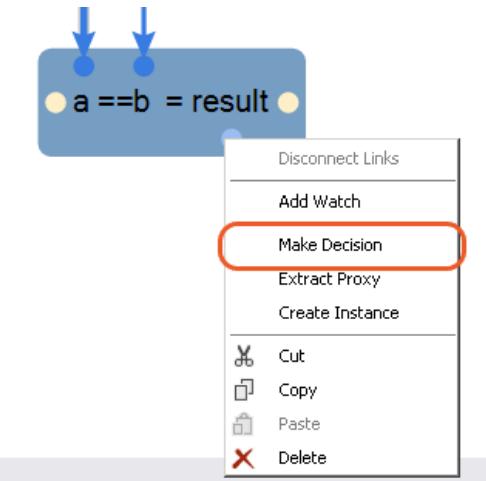
The BooleanExpression component can evaluate double, integer, string, and boolean datatypes and identifier values. You must specify the type of input used in the expression. Setting different data types for two parameters can invalidate the expression.



Decision

The Boolean Expression component provides you with an option to make a decision in a workflow. You can right-click on the output data node of Boolean Expression and select **Make Decision**.

A decision design block appears on the screen and allows you to control the flow path of the workflow by using TRUE or FALSE logical paths.



StringExpression

The **StringExpression** component allows you to add standard string functions (such as concatenation to two string inputs) to the solution.

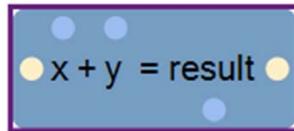
The values can be dynamic or hard-coded data. In the case of hard-coded values, enclose the value in quotation marks.



NumericExpression

The **NumericExpression** component allows you to perform standard mathematical functions such as addition, subtraction, and so on.

You can also pass decimal values to the expression. The number of significant digits (precision) of decimal numbers is set using **Decimals** property.



© 2019 Pegasystems Inc.

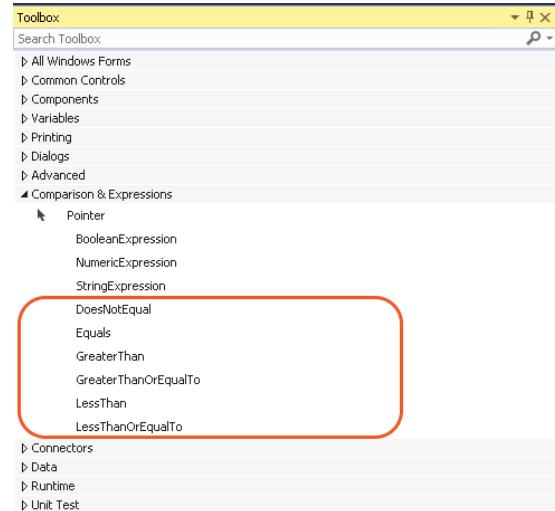
11

Only BooleanExpression allows you to make a decision in the workflow. StringExpression and NumericExpression do not have the provision to add a decision design block.

Comparisons

Pega Robot Studio provides you with a group of comparison components to perform simple logical comparisons. Comparison components yield either True or False result events.

You can find the comparison components in the **Comparisons & Expressions** section of the Toolbox.



The default data type of the values in comparison components is Double. In the case of dynamic input, the data type of the value automatically changes to match the input value.

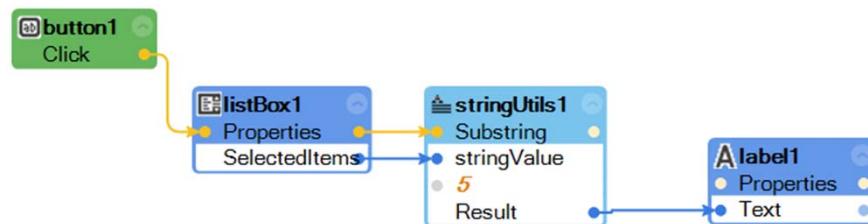
Pega Robot Studio provides utility components to manipulate and format data values for strings, datetimes, timespans, and files. The Utility components are similar to components available in the Microsoft .NET Framework.

After this lesson, you should be able to:

- Describe the use of the utility components for strings, dates, times, and files

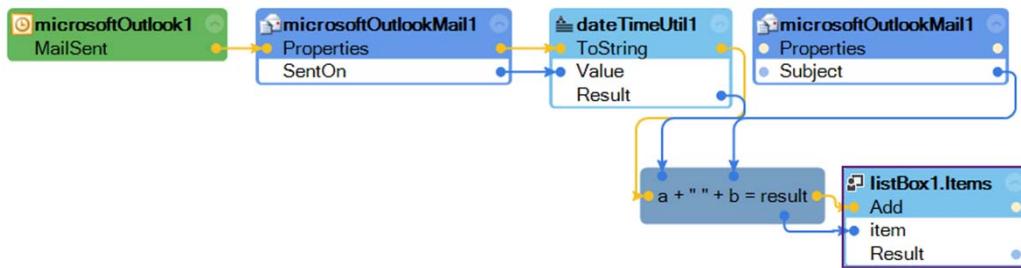
StringUtils

The **StringUtils** component provides a comprehensive selection of string manipulation methods. The component uses the **StringValue** input parameter to specify the string for which you want to apply a method such as Concat, IndexOf, Trim, Insert, and Replace.



DateTimeUtil and TimeSpanUtil

Use the **DateTimeUtil** component to manipulate an instant in time, typically expressed as a date and time of day. Use **TimeSpanUtils** to manipulate a time interval. Pega Robot Studio provides additional methods besides those from the Microsoft .NET Framework.



FileUtils

To manipulate directories and files, use the FileUtils component. The FileUtils methods expose corresponding .NET methods.

When implementing the FileUtils, do not include invalid path characters or lengths in the path parameters and be sure to comply with all security and data protocols. Unauthorized access or I/O exceptions prevent the system from executing the selected method.



If the FileUtils component does not appear in your Pega Robot Studio toolbox, you can add the component by using the **Choose Toolbox Items** window.

LESSON

Automation Looping

Looping is a common method that allows the developer to iterate through a section to validate, perform, or complete an action against a collection of items. Pega Robot Studio provides two components for looping: **ForLoop** and **ListLoop**.

After this lesson, you should be able to:

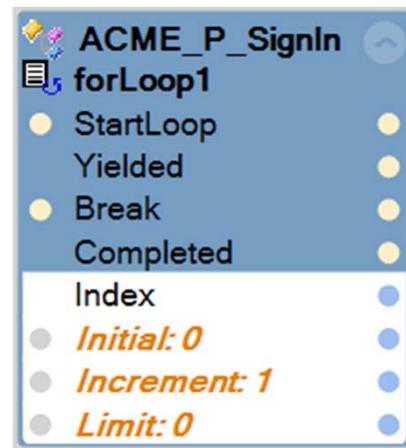
- Differentiate the use and parameters of the Looping components

ForLoop

The ForLoop component enables iterative execution of logic within an automation.

It contains the same elements as the ForLoop structure in programming language that allows you to execute a logic a specific number of times or until a condition is true.

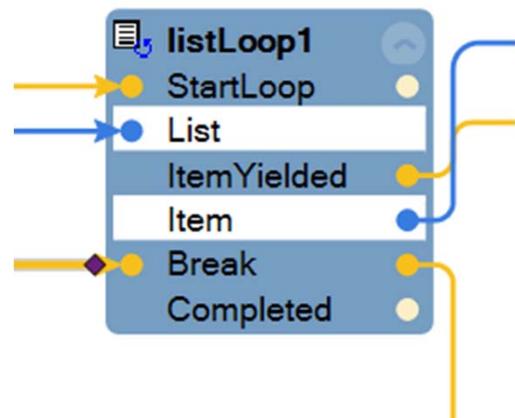
If the break event node triggers, or when you reach the iteration limit, the iteration ceases.



ListLoop

The ListLoop component allows you to read individual items of a list and execute events for each item.

The component not only loops through a count of items (like the Connectable Loop) but also reads the contents of the list and outputs each item.



LESSON

Using Wait Logic

Pega Robot Studio provides a logic that causes an automation to wait or pause until certain objects such as screens, forms, or applications appear on the desktop. Pega Robot Studio contains two wait components: **WaitAll** and **WaitAny**.

After this lesson, you should be able to:

- Describe the use and parameters of Wait components

WaitAll

The WaitAll component triggers an event only if all the objects specified in a group are created.

If the components are not created within the interval specified in the Timeout property, the WaitAll component triggers the Timeout event.



To connect each object to the **WaitAll** component, select each object in the Object Explorer and then add the **WaitHandle** property to the solution. Connect the blue data port from the **WaitHandle** property and connect the port to the **WaitAll** component.

WaitAny

The **WaitAny** component triggers an event when any one object in a group is created.

If an object is not created within the **Timeout** property value, the **WaitAny** component triggers the **Timeout** event.



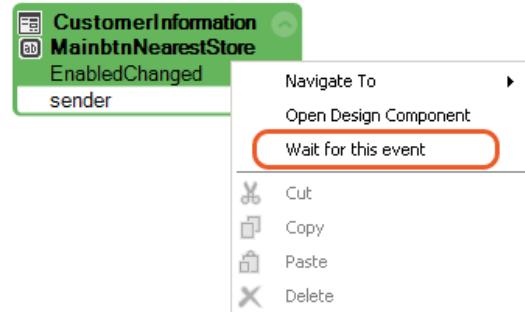
To connect each object to the **WaitAny** component, select each object in the Object Explorer and then add the **WaitHandle** property to the solution. Connect the blue data port from the **WaitHandle** property and connect the port to the **WaitAny** component.

WaitForEvent

When the execution control reaches a **WaitForEvent** design block, it triggers the logic of the Setup event.

Completion of the Setup event logic triggers the Fired event to proceed with the downstream logic.

If the Setup event fails to complete within the timeout period, the timeout event is triggered.



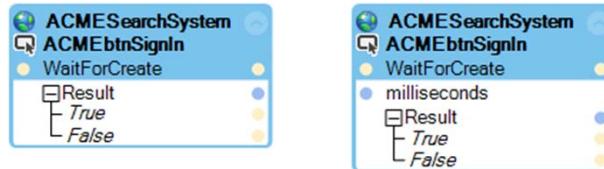
© 2019 Pegasystems Inc.

23

To add the WaitForEvent function on an event, right-click the event and select the **Wait for this event** option.

WaitForCreate

The **WaitForCreate** method applies to an adapter or a control. It enables the automation to wait for the creation of the selected control before triggering a **Result** event. The Result event can be connected to a downstream logic of the automation.



© 2019 Pegasystems Inc.

24

The method without a parameter waits the time specified by the control's **Timeout** property (default of 30000 milliseconds) for evaluating the matched status of the control.

The method with one parameter lets you specify the amount of time, in milliseconds, to wait before triggering the **Result** event.

The **Excel File Connector** expands the functionality of Microsoft Excel connector in Robotic Process Automation (RPA) unattended scenarios when there is no interaction between the user and the Microsoft Excel Workbook.

After this lesson, you should be able to:

- Explain the scope and use of the ExcelFile connector

Benefits

The ExcelConnector provides the following benefits:

- You can create, modify, and read from Excel workbooks by using Pega Robot Studio without an Excel application installed in the environment. The system uses the Open Office XML format to create Excel workbooks.
- The ExcelConnector includes additional methods that enhance your ability to work with an Excel workbook. With the MicrosoftExcel Connector, you can only get or set cell values and export or import data as data tables.
- The ExcelConnector lets you define ranges of data to work with, such as a table. You can define the range during design in Studio or by using an automation during runtime.

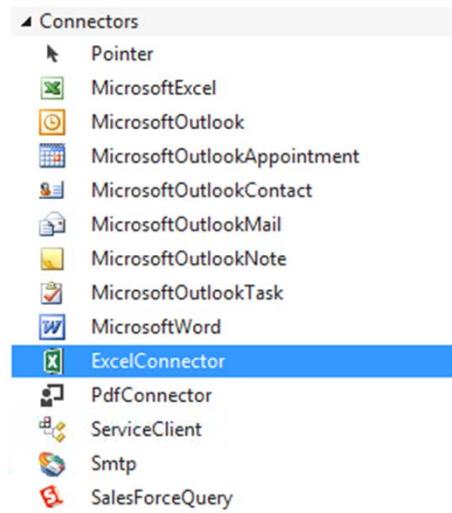
Use the MicrosoftExcel Connector in Robotic Desktop Automation (RDA) automations. For example, use the MicrosoftExcel Connector when you need to automate using events from Microsoft Excel, scenarios where the user needs to interact with the workbook in Excel, and scenarios where you need to run macros in an Excel document.

Benefits

The ExcelConnector component is located in the Connectors section of the Toolbox.

You can add the ExcelConnector component to the automation. By default, this component is located on the **Global** tab.

You can also add the Excel File Connector to the Global Container (preferred).



If the component is not in the Toolbox, you can add the ExcelConnector component to the Toolbox by using the **Choose Toolbox Items** wizard.

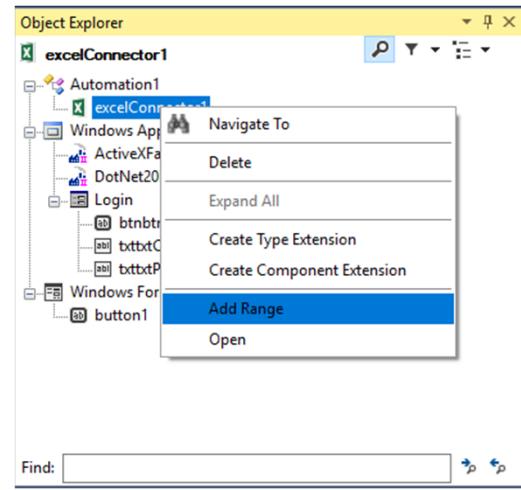
Guidelines for Using Excel File Connector

- Do not use the **CopyCells**, **CopyRows**, **CopyColumns**, **MoveColumns**, and **MoveRows** methods if the destination range contains merged cells. The destination range cannot overlap a merged range.
- When you use methods that read through a worksheet or range, the system starts with column 1 of row 1 and completes all columns of row 1 before proceeding with other rows in sequence.
- The Excel File Connector supports a wide range of formulas. For more information, see the Robotic Automation product page on Pega Community.

Excel File Range Object

Create an Excel File Connector range object to specify the subset of the data in a worksheet for an automation.

To define the range, specify the sheet name and the start and end addresses of the range.



You can create a range object for an Excel File Connector regardless of its location (Automation or Global container).

Excel Range

To determine which Excel File Connector the range object belongs to, look at the name in the component tray (excelRange on excelConnector) or look at the Excel File Connector property for the range. The system keeps this range in memory.

If you make a change to the range and you want to save that change to the Excel File Connector, call the **Commit** method in your automation. If you have made changes to an Excel File Connector that overlap with a range, call the **Load** method to add the new values to the range.

You can create a range object for an Excel File Connector regardless of its location (Automation or Global container).



DEMO

How To Create a Range Object for an Excel File Connector



MODULE

Out-of-the-Box Configurations

This module contains the following lessons:

- Assisted Sign-on
- Start My Day
- Message Manifest

LESSON

Assisted Sign-On

The Assisted Sign-On (ASO) component is an easy-to-use innovation that provides a way to automate the sign-on processes of solution applications.

After this lesson, you should be able to:

- Restate the purpose and structure of Assisted Sign-On
- Implement Assisted Sign-On for adapters

Assisted Sign-On

Assisted Sign-On works with adapters within a solution regardless of cross-referencing or project-to-project reference of the adapter project.

Using Assisted Sign-On:

- You can enter and maintain user ID and password information in a credentials window when Pega Robot Runtime™ launches.
- You can enable automatic login when individual applications present their login screen, automatically populating the user ID and password fields and automatically clicking the login button, to eliminate the need to create an automation that performs the task.

You can enter unpersisted or changed information before you proceed, or you can choose persisted information. You can also reopen the Enter Credentials window by using an automation for credential modifications; the window can show all credentials for all applications or the credentials of one application at a time.

Application Credentials

Pega Robot Studio™ uses the Data Protection Application Programming Interface (DPAPI) to encrypt the application credentials.

DPAPI encrypts data using a private key derived from the Windows identity of the user. Only the same Windows user can decrypt the data.

Credentials are stored locally on the machine in an encrypted file with the file name **ASO.db**, which is located under the user application data roaming directory by default.

Credentials are encrypted in memory by using a randomly generated entropy that is valid only for the current Pega Robot Runtime session.

Extracting a Proxy

The Assisted Sign-On component persists the following strings:

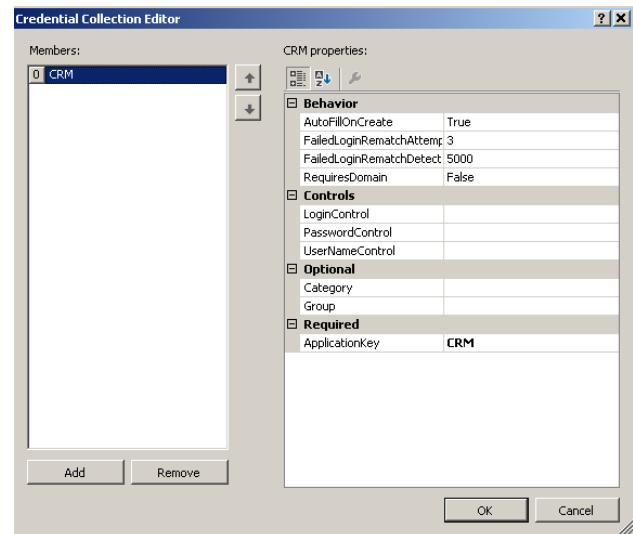
- Application Name
- Username
- Domain

Optional: You can store the password if you set the **StorePassword** option in the AssistedSignOn section in the RuntimeConfig.xml file to True. The default is False.

Implementing Assisted Sign-On

To implement Assisted Sign-On, you must configure an adapter and interrogate the appropriate user ID and password fields along with the login button.

Use the Credential Collection Editor to set up Assisted Sign-On.



Implementing Assisted Sign-On (continued)

You can use settings in the **RuntimeConfig.xml** file to determine whether the Credential dialog is displayed each time Runtime starts, whether the credentials are stored locally on the workstation, and determine the location of the stored credentials file (**ASO.db**).

Property	Description
ShowDialogOnStart	With a value of True , the Assisted Sign-On Credential dialog launches automatically when Pega Robot Runtime starts. The default value is True .
StorePassword	When set to True , the password saves to the local disk of the user after saving the credentials on the Assisted Sign-On Credential dialog.
FileLocation	Enter the path location to store the ASO.db file.

ASO Manager

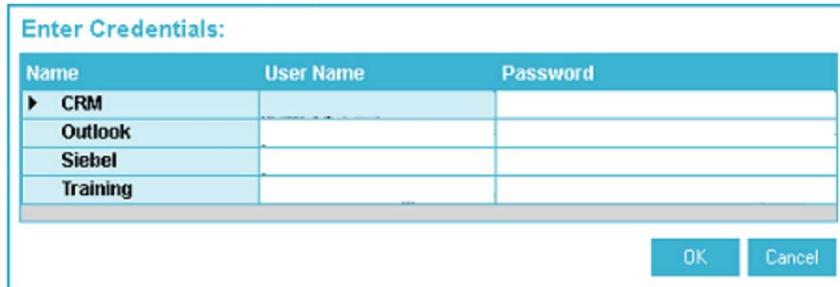
The ASOManager component provides the developer with the ability to access, check, or update any ASO-configured adapter within an automation.

Depending on the project requirements and the nature of the application, using the ASO Manager component provides seamless integration with the end users by removing the accountability of the end users to launch and access solution applications.

ASO utilizes a toolbox component called ASOManager, which provides different properties, events, and methods. The ASOManager helps to automate any necessary processes or tasks related to the end user signing on when the ASO is implemented.

Enter Credentials Window

After setting up Assisted Sign-On, the system displays the **Enter Credentials** window when it needs credential information from the user, such as a username or password. The following image is an example of the window appearance.





DEMO

How To Configure Assisted Sign-On

LESSON

Start My Day

The Start My Day (SMD) component is an easy-to-use innovation that lets you start all applications required for your workday with a single click.

After this lesson, you should be able to:

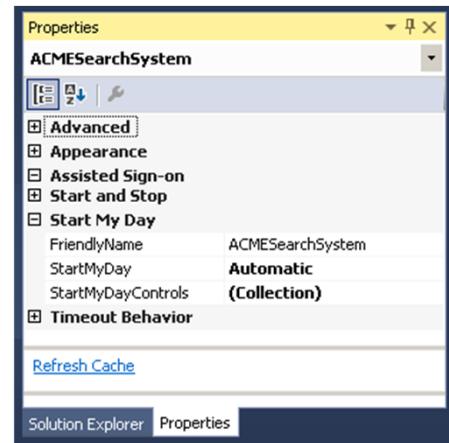
- Restate the purpose and structure of Start My Day
- Implement Start My Day for a solution

Start My Day

The Start My Day innovation can launch, log in, and organize the applications.

This innovation lets you specify activities that users see when they first start Pega Robot Runtime.

This list can include non-Pega programs and URLs, as well as Pega Robot Studio solutions.



Events and Methods

The Start My Day component includes these events and methods to use in automations:

Method/Event	Description
OnStartAdapter	Event raised when the adapter starts.
StartMyDayCompleted	Event raised when the Start My Day component finishes its tasks.
GetStartUpApplications	Method to get list of applications required to start.
ShowDialog	Method to display Manage Applications window.
StartMyDay	Method to execute StartMyDay functionality.

Manage Applications Window

After setting up Start My Day, the end user can use the **Manage Applications** window to prioritize and manage the applications configured to launch using the Start My Day functionality.

Manage Applications:

Wrap Text

Application	Enabled	Type	Position	Size	Path
CRM	<input checked="" type="checkbox"/>	Adapter			C:\Program Files (x86)\OpenSpan\CRM Setup\CRM.exe
ACMESearchSystem	<input checked="" type="checkbox"/>	Adapter			
Google	<input checked="" type="checkbox"/>	Web			https://www.google.com/
notepad++	<input checked="" type="checkbox"/>	Exe			C:\Program Files (x86)\Notepad++\notepad++.exe

Add... Remove Capture Screen Details OK Cancel

Manage Applications Window (continued)

The end user can use the **Manage Applications** window to change the order in which the applications start, enable, or disable executable programs using the **Enabled** option, and add or remove applications from the configuration by using **Add** and **Remove** buttons.

Runtime Configurations

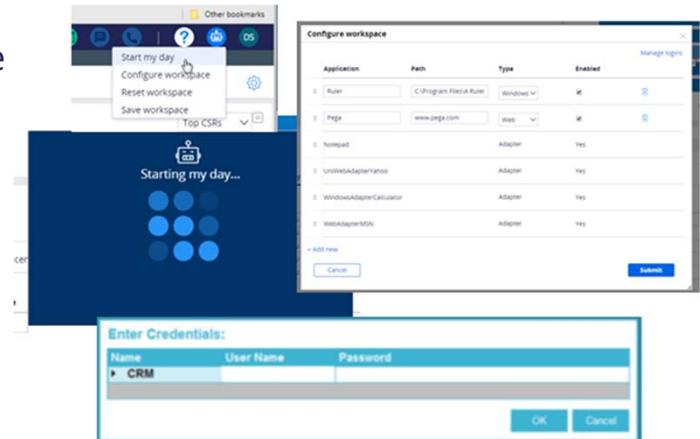
The **RuntimeConfig.xml** contains the settings that allow the end user the ability to perform the actions.

Property	Description
AllowLocalApplications	True/False. Indicates if Runtime users are allowed to add local or web applications as Start My Day applications.
ShowPathColumn	True/False. Indicates if the Path column will be shown in the Start My Day grid.
FileLocation	Path to store the ASO.db file.

SMD with Pega Customer Service

Pega Customer Service™ has the Start My Day functionality, which seamlessly integrates with the application once the application adapters are configured in the solution.

With Pega Customer Service, you can manually organize and size the applications on the desktop.





DEMO

How To Configure Start My Day

As robotic solutions grow, so do the number of messages created to handle any issues. Messages are defined in automations, which results in messages being distributed throughout the solution and projects.

After this lesson, you should be able to:

- Describe the purpose and benefits of Message Manifest
- Describe the structure of Message Manifest
- Implement Message Manifest in a project

Message Manifest

Message Manifest:

- Provides a platform to create standardized messages for all robotic solutions.
- Provides the ability to export messages for reviewing, editing, and sharing of messages with other departments.
- Provides a central location within the project that owns the message definitions.

Message Guidelines

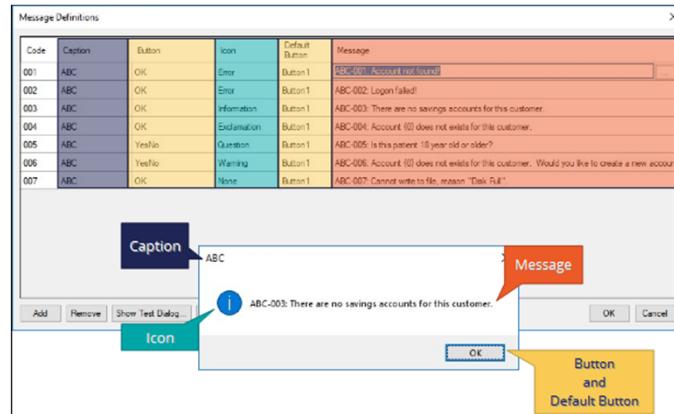
To create a good error message, follow these recommended guidelines:

- Include an indicator as to where the error occurred; this provides the developer with information on which automation contains the error.
- Provide message prefixes to identify the point of failure.
- Provide an error code or exception in the message.



Message Definitions

The message store is a Pega Robot Studio-generated XML file of the message definitions that a developer creates on the Application tab of the project properties.



MessageManifest Component

The MessageManifest component is located in the Advanced category of the Toolbox and must be added to the global container of the project to use the Message Manifest feature.

The component provides:

- The ability to access, manipulate, and display messages
- The JSON serialization and deserialization of the message details of a string variable

Implementing Message Manifest

There are three items to consider and recognize when implementing Message Manifest in a robotic solution.

- How does Pega Robot Runtime™ manage Message Manifest definitions in a multi-project solution?
- How does a developer use the MessageManifest component methods?
- How does a developer pass and return a MessageDetails object?

Multi-project Solutions at Run Time

Consider a solution that contains three projects. Each project has its own Message Manifest definitions created in the properties of each project.

- Pega Robot Runtime starts and loads the solution projects.
- Pega Robot Runtime merges the message definitions from each project into one message definitions store.
- The MessageManifest component in each project provides access to the one central message definitions store.

MessageManifest Component Methods

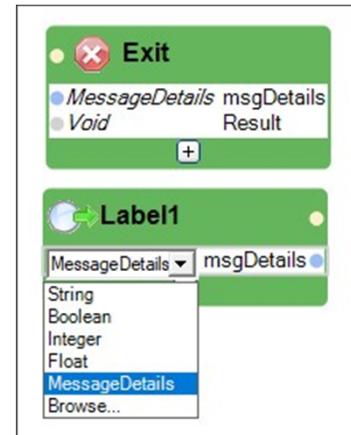
There are four MessageManifest component methods:

Method	Description
GetMessage	Returns the individual properties of the message definition for the specified message.
GetMessageDetails	Returns a MessageDetails object containing the individual properties for a message definition. Input values can be supplied for messages with substitution items.
ShowMessage (1 parameter)	Displays message based on MessageDetails object.
ShowMessage (2 parameters)	Displays message given its code and any inputs needed for substitution.

MessageDetails Object

When the developer uses the GetMessageDetails method, it returns a MessageDetails object representing the specified message. The MessageDetails object exposes the message definition as properties.

As a result, an automation can pass the MessageDetails object as an automation output and as a Jump label input because both the Jump labels and Exit points support the MessageDetails object as a data type.





DEMO

How To Configure Message Manifest



This module contains the following lessons:

- Components of Interaction Framework
- The interaction.xml
- Interaction Manager component
- Project-to-project references

Interaction Framework is a methodology that bridges the capabilities of Microsoft Visual Studio when developing a multi-project solution.

After this lesson, you should be able to:

- Describe the reasons and use for Interaction Framework

Components

The Interaction Framework is a systematic approach that provides Pega Robot Studio™ developers with a collection of tools to incorporate into the development of projects and solutions.

By project-to-project references, Interaction Framework allows projects to communicate with each other. The Interaction Framework simplifies development, improves project reusability, and streamlines solutions for implementing new features

Elements

The Interaction Framework consists of four basic elements:

- Interaction
- Interaction.xml
- Interaction Manager component
- Activity component

The basis of the framework is an interaction.

The Interaction Framework components require access to all other project items within a project and the solution.

LESSON

The interaction.xml

The interaction.xml, or binding contract, pulls all the referenced projects together and allows the projects to communicate information and events between one another.

After this lesson, you should be able to:

- Modify the interaction.xml file to meet the project requirements

Interaction.xml Configuration File

The interaction.xml configuration file defines the context values and activities used in the framework.

The interaction.xml is the only framework component that is solution-wide, meaning a solution requires only one interaction.xml.

By using other framework components, scoped to each project, you reference the one interaction.xml file for the solution.

Visually, you can consider the XML file as the center or hub of the solution, and each project, through the other framework components, read and communicate to each other using the XML file.



DEMO

How To Modify The interaction.xml Configuration File

After this lesson, you should be able to:

The Interaction Manager Component

The Interaction Manager component connects all projects and customer interactions through the XML file by accessing the XML through their properties, events, and methods in the Object Explorer.

These context values appear as the Interaction Manager properties in the Object Explorer.

Each referenced project in the solution has only one Interaction Manager — it connects all the projects to the framework's contexts and activities configured in the XML file.

Slide 9

MC62 Added Lesson Title slide ahead of this one as a placeholder.
Malcuit, Caitlin, 7/12/2019



DEMO

How To Add the Interaction Manager Component

LESSON

Project-to-Project References

Pega Robotic Automation relies on the Microsoft Visual Studio platform. Therefore, some functions of Visual Studio are available to developers. One of these functions is the project-to-project reference.

After this lesson, you should be able to:

- Explain the necessity of project to project references as it relates to Interaction Framework
- Create a project-to-project reference in a multi-project robotic solution

Project-to-Project References

When building a solution or project, Pega Robot Studio creates assembly files. When you have a project that produces an assembly, use project-to-project reference.

The advantage of a project-to-project reference is that it creates a dependency between the projects in the build system.

Pega Robot Studio builds the dependent project if it has changed since the last time Pega Robot Studio built the referencing project.

Project-to-Project References (continued)

When working in an environment where many Pega Robot Studio developers are creating projects that reference the same application, you should create and maintain a single project dedicated to the interrogation of the application and the automation of basic application functionality.

Other Pega Robot Studio developers within your organization may use this application project as a base for developing specific business automations or event data collection or both for other solutions.

Benefits

The benefits of this approach include:

- **Maintenance:** If the primary application changes, new interrogated targets are added or match rules modified, and Pega Robot Studio updates the changes made to the primary application project to all other referenced application projects.
- **Reusability:** A single developer can interrogate the main application and develop the login/navigation logic. Other Pega Robot Studio developers can then focus on building projects that perform specific business automations and event data collections.

Interaction Framework can work with or without creating a project-to-project reference; however, typically a multi-project solution is deployed to the user as one package containing all projects. Therefore, using the project reference with Interaction Framework ensures that the framework works as one unit for the solution and not by project.

Application of Keys

A set of rules determines the creation of keys and the setting of keys for contextual objects.

- An event that creates the instance sets the context.
- An event from No Context to a Context requires a key assignment.
- An event from Context to Child Context requires a key assignment.
- An event from Context to Parent Context does not require a key assignment.
- Logic within the same context does not require a key assignment.



DEMO

How To Create a Project-to-Project Reference



MODULE Implementing Interaction Framework

This module contains the following lessons:

- Interaction Framework context values
- Interactions and activities
- The Activity component
- Interaction versus activity activation
- Updated Framework context values

Implementing the Interaction Framework methodology allows the developer to recognize its strength in the development process for creation of unique and reusable projects that can be shared and quickly modified across several solutions at a client site.

After this lesson, you should be able to:

- Define the structure of context values in the Interaction Framework

Context Values

The interaction.xml file contains the configuration of the context values that represent the data referenced in a solution.

The context values are displayed as properties of the Interaction Manager component in the Object Explorer.

Each project has its own Interaction Manager component, configured to reference the interaction.xml. This configuration allows for storing, editing, and sharing of the context values across all solution projects through the Interaction Manager.

Context Values (continued)

When using the framework, you must initiate the framework by starting an interaction. Once the interaction starts, a virtual data table creates a structure column to represent each context value.

Every interaction started receives a new row in the data table. The data table also has an interaction key column.

The interaction key is similar to a database's primary key, storing a unique identifier of each table row or interaction available in the framework.

When an automation closes an interaction in the framework, the framework deletes the data row and values of the interaction.

Context Values (continued)

The interaction.xml file is the hub of the framework, so the basic data flow is for the data source applications to store the context values in the framework first.

Once stored, all other applications can update or use the context values to complete specific activities.

If data does not store correctly into a context value, the context value uses its configured default value.

Implementing the Interaction Framework methodology allows the developer to recognize its strength in the development process for creation of unique and reusable projects that can be shared and quickly modified across several solutions at a client site.

After this lesson, you should be able to:

- Differentiate between interactions and activities
- Define the structure of an activity
- Implement an activity in the Interaction Framework

Interactions

An interaction is a customer contact achieved through communication such as a phone call, a face-to-face meeting, or a received document. Any of these contact methods initiates the interaction to perform one or many activities to meet the need of the customer.

The Interaction Framework provides the flexibility for a solution to manage one interaction or many interactions simultaneously.

The framework creates a data table to store the interaction context values. A user may have several interactions opened at the same time, but can only work on one interaction at a time.

© 2019 Pegasystems Inc.

7

Compare this situation to your computer desktop: you may have several applications open and running at the same time, but you can work in only one of the applications at a time.

Activities

Once an interaction starts in any project in the framework, all other projects in the framework can complete work or activities for the interaction.

Activities run with the same logic. Once an activity starts in one framework project, the activity can complete in any framework project, including the one in which it started.

You must start or initialize an interaction first to complete any associated activities so that the framework can communicate the success or failure of the process across the projects.

© 2019 Pegasystems Inc.

8

Once all activities are completed and the customer contact has ended, stop or close the interaction to remove the interaction's context values from the data table.

The Activity Component

The Activity component represents any work process for an interaction. Activities can reference actual processes such as add customer, close account, and update customer information.

Activities can also reference back-end work to allow the solution to flow and function, such as refresh windows and search database.

The Activity component can use parameters to pass needed information through to each project to complete the activity successfully.

You add the Activity component to the global container where you have access to manage the activity through its methods in the Object Explorer.

The Activity Component Methods

The following table lists common methods for activities:

Method	Description
CancelActivity	Use this method to stop an activity if the activity hangs for a WaitForCreate method within an automation.
Start	Use this method to queue the activity for the current interaction.
StartNow	Use this method to add the activity to the top of the queue. Any currently running activity does not stop.
StartAndWait	Use this method to place an activity at the bottom of the activity queue and then block the automation from proceeding until the activity has reached the top of the queue and has executed completely.
StartNowAndWait	Use this method to place an activity at the top of the activity queue and then block the automation from proceeding until the activity has executed completely. Any currently running activity does not stop.

You add the Activity component to the global container where you have access to manage the activity through its methods in the Object Explorer.



DEMO

How To Add an Activity Component

The Interaction Framework allows for concurrent interactions; however, the users can only work on one interaction at a time. Understanding this makes development decisions easier.

After this lesson, you should be able to:

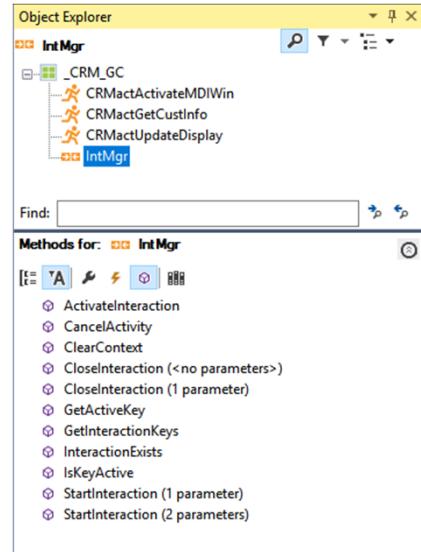
- Implement the activation of an interaction and an activity in a solution
- Implement automations to respond to a change in context values in a solution

Interaction

Pega Robot Studio provides a method to activate an interaction on the Interaction Manager component.

When you invoke this method in an automation, Studio makes all other interactions inactive.

Activating an interaction can initiate other activities to refresh the supporting project applications to match the new active interaction.

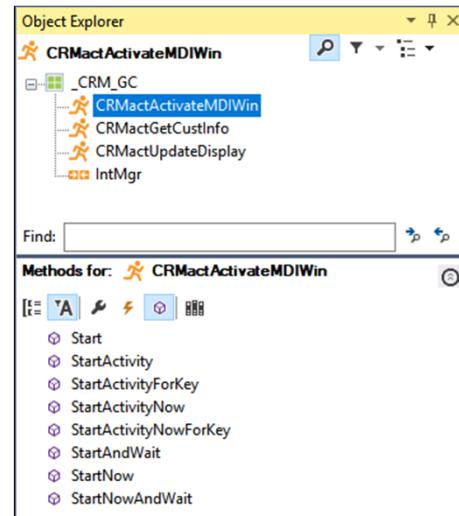


You can consider this logic similar to a relational database or your computer desktop. A database contains several to thousands of rows of data. You can edit only one row of data at a time.

Activities

Activities do not have an Activate method, they have a Start method. Similar to interactions, once an activity starts in the project in the Framework, all other projects can respond to the activity started.

When starting an activity, the related automations process the activity based on the current active interaction.



Updated Framework Context Values

When implementing the framework within a solution, it is common to have different processes or activities update an interaction's context values.

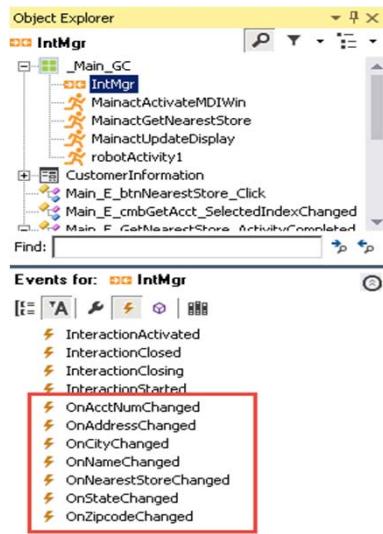
A context's value may also force an automation to perform another series of activities based on its value.

Because of these common occurrences, the framework provides a mechanism that allows a developer to respond to a change in a value.

Updated Framework Context Values

The Interaction Framework's flexibility creates a dynamic event for the Interaction Manager component.

For every context value added to the Context section of the interaction.xml, the Interaction Framework creates an event based on that context value.



A single automation may contain a line to handle each specific context change event, not a separate automation for each context value. Remember to use procedure automations to perform the actual process of updating any interface or applications.



This module contains the following lessons:

- Robotic solution deployment
- Robotic configuration files
- Project properties
- Deploying to Robot Manager

LESSON

Robotic Solution Deployment

Pega Robot Studio™ uses a deployment process that developers can implement regardless of the distribution strategy used by an organization. The process allows developers to adjust the solution's configuration settings.

After this lesson, you should be able to:

- Explain the recommended deployment method
- Recognize the deployment package files
- Explain how the package files are used to update and deploy new packages to users

Solution Deployment

Deployment is the compilation of all project and solution components into a usable package. On completion of development, you deploy the robotic solution to run on a Pega Robot Runtime™ application.

As a developer, you must configure the solution and its projects to create deployment package files to distribute the solution.

Deployment Package Files

The deployment package consists of two file types:

File type	Description
.openspan	Contains the compiled version of the project with all the referenced assemblies, translators, and configuration files needed to run the project in the run-time application.
.manifest	Contains project information such as a list of all .openspan files, project versions, and so on.

© 2019 Pegasystems Inc.

4

For example, deployment of the CRMPPrj project results in the following files:

CRMPPrj.manifest

CRMPPrj.openspan

Solution Deployment

Pega Robot Runtime validates the deployment package before executing the solution by comparing the manifest files of the user's machine and new deployment location.

The details of the location and how to use deployment package files are present in the **Runtimeconfig.xml**.

You can change the version of the solution by updating the .manifest file in the deployment package directory.

Solution Deployment

To deploy a solution for either an attended (Pega Robotic Desktop Automation™) or unattended (Pega Robotic Process Automation™) solution, the desktop must have Pega Robot Runtime installed.

The Pega Robot Runtime must have access to all the applications required to execute the solution.

While you can deploy the robotic solution using various deployment strategies, it is preferable to use Pega Robot Manager.

The process of deployment, execution of solutions, and projects from Pega Robot Studio is the same for any deployment strategy.

Consider a solution that uses a web application. During the application discovery, you realize that the application has three different URLs for training, testing, and production. How does a developer adjust the project and solution for deployment?

After this lesson, you should be able to:

- Explain how the package files are used to update and deploy new packages to users
- Define the configuration file project item
- Explain the benefit of adding configuration files to a project
- Create a configuration file for a project
- Add a new value to a configuration file

Configuration Project Item

The Configuration project item stores different configurations for a single solution.

As a developer, before creating a deployment package for the solution, you can add configuration files to create different deployment packages based on each configuration file.

You can use the **Configuration** project item to set the properties of a control to different values and run or deploy a single project using each of the property-value configurations. In this way, you use a single solution to create multiple deployment packages of the solution.

When using Interaction Framework and Configuration project items in a multi-project solution, it is a best practice for all projects to have configuration project items that share the same name. Doing so ensures that the controller project has all the necessary configuration property values.



DEMO

How To Create Project Configuration File

You can define properties of a project in the Properties window (project design properties) and in the Project Property Pages. These properties let you specify the project behavior at run time and specify how to build the deployment package.

After this lesson, you should be able to:

- Define the use of each project properties tab
- Create a deployment package

Application

Property	Description
Target Framework	Use this property to specify the version of .NET you will use. The drop-down lets you select from the versions you have installed. Note that the version of .NET you select should match with the version of .NET Runtime uses.
Startup Application	Use this property to specify the path and executable name you want to use to load the project. The default is Runtime.
Configuration File	Use this property to add the location and name of the configuration file that defines the interaction properties, activities, and tasks. By default, the configuration file is named Interaction.xml .
Miscellaneous Files	Use this property to add files to the runtime project package. To add files, click Browse in this property to open the Deployment Files Editor. Use the Deployment Files Editor to add members and set corresponding files.
Configuration	Lists all Configuration profiles created for this project. To create a deployment package for a specific Configuration Item profile, specify the profile in this property. Otherwise, no configuration profile is applied and object default values for properties are used when building the deployment package.
Message Definitions	Lists all the message definition details for user message display using Message Manifest components.
Startup Form	If the project contains multiple Windows forms, AppBars, or both, use this property to specify the item you want to be in focus when the project initially loads.

Build

The Build properties are used whenever you build and deploy a project. These properties depend on the Solution Configuration selected (Release, Debug, and so on).

- For example, you can set the **Debug Symbols** property to False for the Release Solution Configuration and the Debug Solution Configuration to True.

The most important property of the Build Property Page is **Output Path**. The Output Path specifies the folder which contains the compiled project files.

Signing

The Signing properties configure whether the assemblies are strong-name signed. Strong-naming an assembly creates a unique identity for the assembly and can prevent assembly conflicts on the end-user machine.

When you reference a strong-named assembly, you can expect certain benefits, such as versioning and naming protection.

Property	Description
Key File Path	Specify the relative or absolute path to the strong name signing key pair (.snk file) to use for signing the project assembly. The default is an automatically generated key that the system creates for each project.
Sign	Select True to enable strong name signing for the project assembly. Strong name signing the main project assembly lets you reference it from a strong name signed custom executable. The default is False. Normally, you would only use strong name signing if you have a strong name signed executable that references the package.

Deployment

Property	Description
Deployment Version	The version of the deployed project. The version number starts at 1.0 for a new project. Each time the project is deployed, the Deployment Version is incremented. For example, if a project has been deployed three times, the Deployment version would be 1.2.
Include Assemblies	True/False. This property lets you control whether the assemblies required for the project are included in the built deployment file (.openspan). Usually, you would set this to True so all project assemblies are included in the package. If, however, you want to limit the size of the project deployment file, you can set this to False.
Include Run Settings	True/False. This property lets you deploy the project with the Run Actions currently set for the project in Solution Explorer.
Include Translators	True/False. This option lets you control whether the translators required for the project are included in the built deployment file (.openspan). Usually, you would set this to True so all project translators are included in the package. If, however, you want to limit the size of the project deployment file, you can set this to False.
Increment Deployment Version	True/False. Defines whether the Deployment version is incremented each time the project is deployed. Enter True to increment the Deployment version.
Create Deployment Package	True/False. Set this property to True to create a deployment package for the project as part of the build process. If the project is used strictly as a reference within the solution, set this property to False.
Output Path	Indicates the location for the deployment package files. Use the relative project path.



DEMO

How To Deploy a Solution

LESSON

Deploying to Robot Manager

Pega Robot Manager is the recommended deployment, distribution, and maintenance strategy for robotic solutions. To deploy the solution package to Pega Robot Manager, each development environment requires a server connection along with a Package Server.

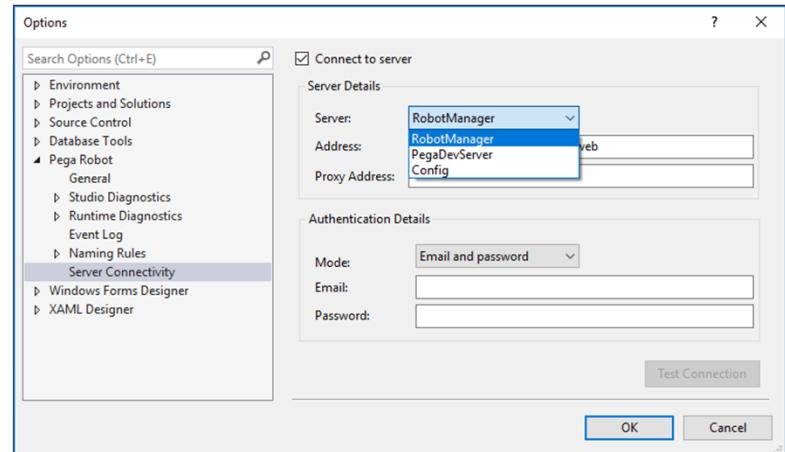
After this lesson, you should be able to:

- Relate the location of Pega Robot Manager server connectivity in Pega Robot Studio
- Describe the results of the server check box during the deployment process

Deployment

The server connection for Pega Robot Manager is set up during the installation of Pega Robot Studio.

However, the developer can modify the settings when required. For more information about Pega Robot Manager, refer to the Pega Community.



© 2019 Pegasystems Inc.

17

PegaDevServer refers to the Pega Platform™ development environment where you can send an automation package to the Pega platform for the development of a Pega application. The PegaDevServer option works with the Pega Robotic Automation Importer (8.0 SP1 2029). It allows the sharing of automation packages between the Development, UAT, and Production platform instances. Currently Pega Robotic Automation Importer only works for automations used for attended Robotic Desktop Automation (RDA).

Deployment

There are two types of configurations: Server Details and Authentication Details:

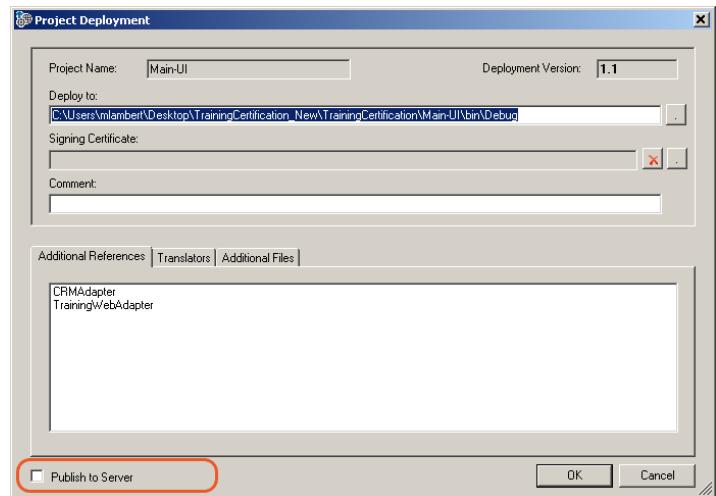
Server Details	Description
Server	Lets you choose the preferred deployment method. The default value is RobotManager.
Address	Defines the URL of the selected server.
Proxy Address	Defines the proxy address of the server if required.

Authentication Details	Description
Mode	Indicates the mode of authentication, such as Email and password (default), Active Directory , and Kerberos .
Email	Indicates the email Id for authentication. This option is available only if the authentication mode is Email and Password .
Password	Indicates the password for the authentication.

Deployment

The above settings help you configure Pega Robot Manager as the default deployment method.

It enables the **Publish to Server** option in the Project Deployment window. No matter what deployment method you choose, the process of deployment remains the same.



Deployment

Use the comment section in the Project Deployment window to provide a description that describes the purpose of the project.

The values you enter here display in the Description column of the deployment package details in the Pega Robot Manager and help determine which version to deploy to users and robots.

Available versions			
Version	Description	Published by	Published on
v1.4	Updated fetchStoreAddress	Admin For RoboSample App	5/20/2019
v1.3	Updated FetchStoreAddress automation	Admin For RoboSample App	5/20/2019
v1.2	Added functionality for retrieving & formatting address	Admin For RoboSample App	5/20/2019
v1.1	FetchStoreAddress public automation refined	Admin For RoboSample App	5/20/2019

