Recommended Practices for Solution Building

Pega® ACADEMY

Trademarks

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. may make improvements and/or changes to the publication at any time.

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 information 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

Chapter 2: Recommended Practices for Solution Building

OpenSpan lets you create solutions that integrate and automate desktop applications without writing code. When creating solutions, you go through many of the same logical steps and practices required in software programming. This chapter describes both general advice applicable to any programming project, and specific practices recommended for creating enterprise solutions with OpenSpan Studio.

Key Points to Consider when Designing Solutions Based on User Actions

- Capture the sequence of actions you want to automate in a series of screen shots and make a note of all your keyboard and mouse interactions. The more details you capture, the easier the design and testing process are.
- Base your solution design on the user interactions with the applications and processes you plan to automate.
- Each process that you are automating in your solution must begin with an event or entry point — user and system actions play an important role in solution design.
- Each project should contain only one adapter.

Key Points to Consider when Interrogating Applications

- Interrogating every application you need in the solution before building the automations is recommended. By doing this, you can uncover potential application integration issues before building the automation logic, and discover how to plan around them.
- Test applications in the OpenSpan environment by unit testing solutions.
- Follow recommended practices when finalizing match rules.

Key Points to Consider when Naming Controls

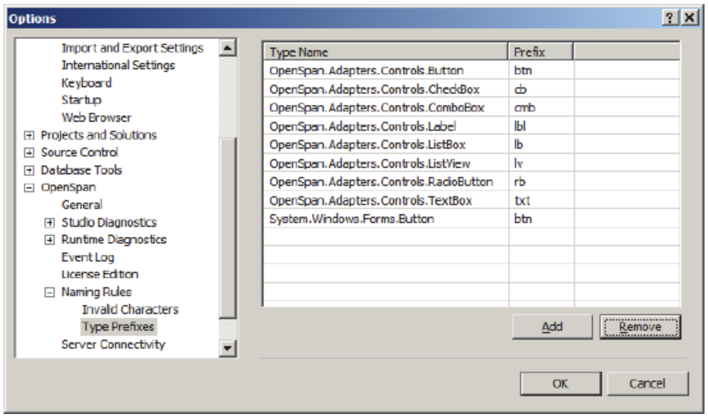
- Solutions and projects should be named in a way that helps you easily identify the high-level function they perform.
- Adapters should be named so you can easily identify the application they represent. Assign names to your adapters that identically match their current application shortcut names.
- Automations should be named in a way that helps you easily identify the action they perform. Automations should follow the same naming conventions as methods in their code counterparts.

Establishing standardized naming convention for controls and components is critical. Whether you are building an automation, debugging a solution, or troubleshooting match rules, having the ability to quickly identify the controls helps simplify the task.

Here are some examples of standardized naming convention prefixes for several common controls.

Common Control	Standard Prefix
Button	btn
Check Box	cbx
Combo Box	cmb
Label	lbl
List Box	lbx
List View	lv
Radio Button	rbtn
Text Box	txt
Toolbox	tb
Windows Form	frm

You can modify the default prefixes used by OpenSpan Studio for common controls by using the Auto-Naming Rules function. Access this function from the OpenSpan Studio Tools | Options | OpenSpan | Naming Rules | Type Prefixes menu option.



Type Name	Prefix
OpenSpan.Adapters.Controls.Button	btn
OpenSpan.Adapters.Controls.CheckBox	cb
OpenSpan.Adapters.Controls.ComboBox	cmb
OpenSpan.Adapters.Controls.Label	lbl
OpenSpan.Adapters.Controls.ListBox	lb
OpenSpan.Adapters.Controls.ListView	lv
OpenSpan.Adapters.Controls.RadioButton	rb
OpenSpan.Adapters.Controls.TextBox	txt
System.Windows.Forms.Button	btn

Key Points to Consider when Managing the Solution Hierarchy

- Keep the Solution Manager hierarchy organized. User folders to group related items based on Adapters and Events | Public and Private Automations.
- Recommend using the top-level folder to group items into systems. A system is defined as a group of related functionality surrounding an individual adapter (or multiple adapters) with the ability to share the entire contents of the system between different solutions.

Key Points to Consider when Working with Automations

- Add comments text as a header to all automations. This is helpful when working in multi-designer environments.
- Keep automations with the task or adapter to help make locating easier when you debug or edit your solution.
- Set the automation's ShowDesignCompName property to True, so OpenSpan Studio displays the full design name of the components in the automation design block.
- Before using the properties, events, or methods of a component in an automation, use a WaitForCreate or an IsCreated design block to confirm that the component is created and matched. Typically, only one design block of this type is needed per automation.
- All automation Boolean branches have a logical termination point.
- Set up Global Containers to store variables or components that can be called from multiple locations with the solution.
- Include an automation or a Miscellaneous Text file for storing notes and tracking any revisions to the solution.
- Run one connector line from a single design block execution point.

Note The order in which the connector links are created dictates the order in which they are executed during the running of the solution.

Key Points to Consider when Debugging

- Add MessageDialog components to automations while building and testing solutions to confirm event handling.
- Use error trapping around data entry. Be sure to provide the logic to check that the input is valid for use in your solution.
- Add diagnostic logging to set customized messages, using unique words or characters, to write to the diagnostic file for easy searching within the file. Consider for easy searching.
- Set up a Universal Diagnostic Automation that can be called from anywhere in the project for an efficient way to implement diagnostic logging in a larger, more complex solution.

Key Points to Consider when Deploying Solutions

- Use a single deployment location.
- Use a Common Extract Directory.
- Use error trapping around data entry. Be sure to provide the logic to check that the input is valid for use in your solution.

Note Please refer to the following Knowledge Base articles and Help for additional information about working with OpenSpan Recommended Practices:

- Building Solutions
- Establishing Coding Standards
- Understanding Threading
- Using ExceptionThrown Logic
- Using Match Rules
- Using Source Control