

QA Wizard Pro Best Practices

This document addresses some of the common activities in QA Wizard Pro and offers best practices for each. These best practices form the basis for effective test automation using QA Wizard Pro. Implementing them can help you avoid common mistakes and improve your testing process.

Application Repositories and Workspaces

Create an intuitive file structure

QA Wizard Pro stores workspaces in the QA Wizard Pro Workspaces folder in My Documents. Workspaces can grow quickly as you develop scripts. To effectively manage scripts and data, you can create a workspace for each functional area you are testing and save the workspaces in separate directories. Suppose you are creating a regression suite for WysiWrite, a word processing program. One of the areas you will test is the Tools menu, which contains eight options.

To stay organized, you create a directory for WysiWrite and a subdirectory for the Tools menu. Then you create workspaces and subdirectories for each option on the Tools menu to organize the scripts, batches, and data used to test each option. Figure 1 shows the recommended directory structure. You can also use QA Wizard Pro folders to further organize the scripts, batches, and data in your workspaces.

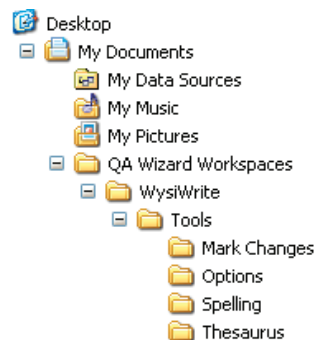


Figure 1: Sample directory structure

Test changes in a local application repository first

QA Wizard Pro includes local and global application repositories. Local repositories are specific to one computer. They are generally used when only one tester is recording and running scripts against an application version or when setting up an application before it is promoted to a global repository.

If more than one tester will be testing an application, begin by having one tester record several test scripts to capture the control

data, then adjust the names and search criteria, and review the tests with the new test data. When the test scripts are running properly with the local application repository, and you are confident that all the controls in the test application have been captured, promote the local application to the global application repository.

Establish a repository naming convention

After new windows or controls are added to the application repository, review the names to make sure they are intuitive. QA Wizard Pro tries to create a meaningful name during recording, but you may want to rename some windows and controls so they are more consistent with the tested application.

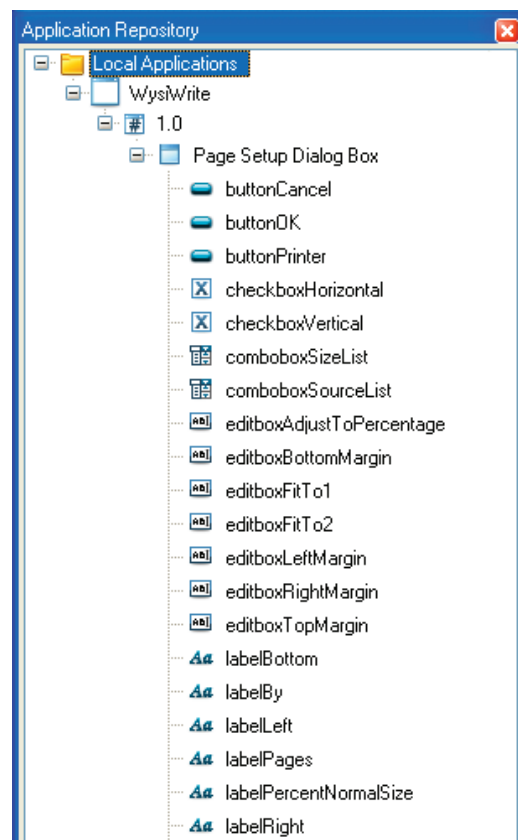


Figure 2: Windows and controls naming example

Windows and controls are displayed alphabetically in the application repository, and controls are grouped by the window they are captured in. This can impact how you name controls. While there are no hard and fast rules for naming controls, it is important to establish naming conventions if more than one person will be accessing the application repository.

QA Wizard Pro displays different icons based on control type. Some testers prefer to append an abbreviation to the control name to group controls by type (see Figure 2) or to distinguish between browser windows if a limited number of windows were captured during recording.

Change the control type for generic controls

Generic controls are controls that QA Wizard Pro cannot recognize. Scripts can only use low-level action statements to interact with generic controls. You can change a control type if you want to identify it more accurately so you can use other statements to interact with the control. For example, you can change a generic control that you enter text into to an edit box. This allows you to add statements that perform object actions, such as Set Text, on the control instead of relying on low-level action statements.

Define unique search criteria

When selecting search criteria, the goal is to create a unique identifier for the control. QA Wizard Pro smart matching can automatically update search criteria for windows and controls used by scripts to determine the difference between similar controls. However, you may want to select different search criteria based on your knowledge of the application. Following is a list of common properties you can use as search criteria.

Name—The name of the control. This property value is always provided by the application but may not be captured for all controls.

Left Control Text—Text to the left of the control. This can be the control label or the text in a button or check box to the left of the control. (Not applicable for web controls.)

Top Control Text—Text above the control. This can be the label for the control or the text in a button or check box above the control. (Not applicable for web controls.)

Vertical Repeat Index—The order of two or more controls of the same type that have the same X coordinate. For example, a form includes several vertically aligned edit boxes.

The first edit box has a Vertical Repeat Index value of 0. The next edit box has a value of 1 and so on. QA Wizard Pro numbers the controls of the same type that share the same X coordinate sequentially based on their Y coordinate.

Window Class—The Windows-registered class name for a control. Examples include button, static, and edit. This property is not unique and should be used with another property.

Href or URL—The URL associated with a hyperlink. If you are testing a web application with dynamic URLs, you may want to create a URL format rule to ignore individual parts of a URL.

Text—The default text that appears in an edit box or combo box or the text that appears on a button or as a label. Text can generally be used alone to define search criteria for buttons or check boxes.

For advanced web control identification, you can use the XPath search method. XPath queries search for HTML elements that are dynamic or not stored in the application repository. While some controls use the XPath search method by default, only use this method if you understand how the search options may impact your scripts.

Wait to create new versions

Each application in the repository contains one or more versions. Each version contains information for starting the application and window and control data that is captured during recording. Maintaining versions allows you to continue testing previous versions of an application while creating new scripts to perform tests on new versions.

An application can have hundreds of controls stored in the application repository. When you create a new version of that application, you end up with two copies of the controls that must be maintained. Minor or maintenance releases usually do not require a new version in QA Wizard Pro. As long as most of the scripts run against the new version, you can continue using the version in the application repository. You can simply record a script to capture new or changed controls. Remember that you are committing to maintaining twice the number of controls when you create a new version.

Create a new application version in the repository if a large number of scripts begin to fail after a new release. You do not have to re-record existing scripts to work with the new version. You can copy an existing script, change the Set Context statement to reference the new version, and then edit the script steps.

Streamline scripts

The more steps a script has, the harder it can be to interpret what it is doing. Some action types are required but others can be deleted without affecting script execution.

Required

- Type Text
- Select

Can be deleted based on the situation

- Mouse click
- Double-click
- Low-level mouse up
- Low-level mouse down

Scripts only require steps that perform an action such as entering a value or submitting data. Generally, you do not need steps that select or move to another field.

Use the Call Script statement to keep scripts small

QA Wizard Pro allows you to quickly capture all the steps of even the most complex process. Capturing an entire test scenario in a single script can make it difficult to troubleshoot and limit the script's flexibility. It is best to create scripts that perform one small activity, such as selecting a menu or completing a dialog box. You can then combine scripts using the Call Script statement to create scripts that test multiple areas of an application at the same time.

Manage shared test data in datasheets

If multiple scripts use the same test data, you can store the shared data in a local or external datasheet. Separating the test data from the script makes it easier to modify scripts or data as needed without making the same change in every script.

Use repository variables for dynamic data

If scripts test data that changes based on the application or conditions, you can create repository variables to maintain reusable scripts without having to modify individual scripts every time the data may change. Using repository variables also reduces the need for maintaining multiple data sources.