AI-powered intelligent test automation with OpenText Functional Testing

Simplify and improve test creation, execution, and maintenance through capabilities based on AI and intelligent automation

AI-powered intelligent test automation:

- Detects and fixes defects before they escape into production
- Reduces costs through simple test creation and maintenance, reusable and resilient test scripts, and lower test infrastructure expenditure
- Ensures products meet customer expectations on all platforms

When it comes to building and delivering better software faster, organizations can no longer choose between speed and quality if they expect to remain competitive. They need a faster way to engineer quality into every application.

OpenText anticipated this need for faster, smarter testing and embedded multiple AI-powered intelligent automation capabilities into OpenText™ Functional Testing.

AI-based test automation capabilities

AI-powered intelligent test automation

Simplify and improve test creation, execution, and maintenance through AI-powered intelligent automation. AI-based machine learning and advanced OCR provide for advanced object recognition and, when combined with AI-based mockup identification, AI-based recording, AI-based text matching, and image-based automation, teams can reduce test creation time and test maintenance efforts, boost test coverage and resiliency of testing assets, and cut down on test maintenance efforts.

- **Advanced object recognition**
  Streamline test creation and execution by more naturally identifying objects, similar to how a human “sees” them. This is enabled through AI-based machine learning and advanced optical character recognition (OCR).

- **Object interaction**
  Increase test resilience by interacting with visual objects on the screen the same way as a user would. The OpenText Functional Testing Neural Network understands each object and its context and manipulates it in a natural way. Together, The OpenText Functional Testing AI-based object recognition and object interaction, along with natural language script creation allow a single script to run on multiple platforms without requiring any modification, increasing test accuracy, resiliency, and velocity.
- **Natural language test script creation**
  Reduce test creation time and ease test maintenance with the OpenText Functional Testing natural language processing (NLP) engine, which enables tests to be written in simple English.
- **Record AI-based test steps**
  Perform a business flow on an application and create a sequence of AI-based test steps. Within the recording session, you can also fine-tune the object identification and add checkpoint steps. The recording result is a resilient test that can run on multiple platforms and environments.
- **AI-Based Mockup Identification**
  AI-Based Mockup Identification inspects application mockups and identifies objects that could be used in a test. This allows for the preparation of tests much earlier, enabling for test design even before an application is fully developed.
- **AI Transformation Assistant**
  Run existing tests with the AI Transformation Assistant enabled and receive suggestions for transforming technology-based test steps to AI-based testing steps. Replace existing steps with the suggested AI object steps to create a more resilient and platform-agnostic test.

Intelligent test automation capabilities

Image-based processing

Keeps up with unpredictable UI changes by learning objects like humans do—through image-based automation, visual anchors, and embedded OCR with either the ABBYY OCR engine or the Google Tesseract OCR engine.

- **Image-based automation**
  Identifying objects using Insight (Insight): Insight enables OpenText Functional Testing to recognize objects in the application based on what they look like, instead of properties that are part of their design. This can be useful for working with an application running on a remote computer.
- **Visual anchors**
  Visual relation identifiers (VRI): To improve object identification, create a visual relation identifier, which is a set of definitions that enable for the identification of the object in the application according to the relative location of its neighboring objects.
- **Embedded OCR**
  Text recognition in run-time (Text recognition): When working with tests and scripted components, the text and text area checkpoint or output value commands can be used to verify or retrieve text in objects.

Resources

OpenText Functional Testing web page ›

OpenText Functional Testing Data Sheet ›

OpenText Functional Testing Help Center ›

DevOps Cloud Blogs ›

OpenText ›

Machine-driven regression testing

Find anomalies easily, such as scripting errors, visual regressions, broken links, and more.

- **Scripting errors**
  Smart identification (Smart Identification): When OpenText Functional Testing uses the learned description to identify an object, it searches for an object that matches all of the property values in the description. In most cases, this description is the simplest way to identify the object, and, unless the main properties of the object change, this method will work. If OpenText Functional Testing is unable to find any object that matches the learned object description, or if it finds more than one object that fits the description, OpenText Functional Testing ignores the learned description, and uses the Smart Identification mechanism (if defined and enabled) to try to identify the object.
- **Visual regressions**
  The Applitools Eyes OpenText Functional Testing SDK allows for visual checkpoints to be easily added to OpenText Functional Testing tests. It also produces screenshots of the application from OpenText Functional Testing, sending them to the Eyes server for validation and failing the test if case differences are found.
- **Broken links**
  Using Page checkpoints for broken links (Page checkpoints): Use page checkpoints to check statistical information for key web pages. These checkpoints inspect the links and the sources of the images on a web page and instruct page checkpoints to include a check for broken links.

Text analysis

Extract text and data values directly from an app for analysis or collect analog text directly from images.

- **Data extraction**
  Test Combinations Generator (TCG) enhancements: The OpenText Functional Testing's TCG tool supports an additional method of generating values from list objects, by pulling data directly from the application that is being testing.
- **Text from images**
  Text recognition in runtime: When working with tests and scripted components, use the text and text area checkpoint or output value commands to verify or retrieve text in objects. The OpenText Functional Testing identifies text in an application via an OCR mechanism.

Synthetic data creation

Create data intelligently using multiple algorithms to reduce the size of a test data set without serious loss of quality.

- **Data creation**
  Generate data to drive your test (Test Combinations Generator): The Test Combinations Generator helps to prepare test configuration data by using the parameters in the test and their possible values to create multiple data combinations. Once the data is specified, and depending on the number of parameters, this task can grow exponentially. Use the Test Combinations Generator to do the work automatically.

opentext™