January 25, 2016

MEDIATOR FOR INTEGRATION TESTING

Karin Lundberg

Kamakshi Kodur

MEDIATOR FOR INTEGRATION TESTING

ABSTRACT

The subject technology provides a software infrastructure that makes it easy to write integration tests for applications and related back-end processes that utilize services over a network (e.g., web applications, operating systems, mobile applications, browsers, etc.) to, for example, improve automation coverage and prevent certain security issues. A Mediator component provides a mechanism for connecting disparate infrastructures to allow applications and/or devices under test to utilize and communicate with processes that would not normally be available during integration testing.

PROBLEM

Development of web-based operating systems and applications may depend on multiple back end processes that are not part of the development environment. While it may be possible to run automated tests of these systems and applications inside a controlled infrastructure, for example to test capabilities of web applications such as search or email, or application synching, it may not be possible to reliably test integration with back end applications that reside in a controlled infrastructure outside of the immediate development environment. Integration testing may be particularly problematic when the code to be tested lives in repositories that are external to the code repository containing the code for the back end applications. Other testing methodologies have included using certain storage solutions for "communication" between different infrastructures. However, these solutions have the disadvantage of only allowing writing and reading files.

THE MEDIATOR ARCHITECTURE

The subject technology solves the foregoing problem by providing an application programming interface (API) that can be used from various infrastructures to start back end applications residing in disparate development and production environments and to provide restricted and secure access to these back end applications. Providing “restricted and secure” access to such applications, e.g., to applications in development environments, may be important in that the back end code may not have been published.

The subject technology provides a secure way to send specific requests between two infrastructures using a Mediator component that only allows authenticated requests using whitelisted accounts. The Mediator uses protocol buffers to specify the requests that can be sent and it is easy to extend it to support new types of requests. Furthermore, the Mediator supports extensive error handling making it easy to debug potential issues. In general, the Mediator is extensible, supports specific requests, and ensures secure and reliable communication..

The Mediator of the subject technology enables the start of integration test workflows from disparate environments using a proprietary, controlled infrastructure. These workflows start back end applications (e.g., in either a production environment or development environment) and keep them running until either the workflow times out or the Mediator gets a request to delete the workflow. When the back end applications are started, the Mediator registers the back end applications with a special (front end) Web Server (see Figure 1). In one or more implementations, uniform resource locators (URLs) for accessing the back end applications are provided to the Mediator to distribute to authorized applications and/or devices for accessing the back end applications. This allows whitelisted applications and/or devices to connect to the back end applications (which may be only available from inside a development environment) and to run tests against these back end applications (e.g., verifying a search query is performed in response to search input within the application). The applications and/or devices may be whitelisted by being connected to specific wireless access points, which may be password protected to avoid unauthorized connections.

Developers may require the ability to write, run, and debug integration tests while maintaining the code on the same continuous build infrastructure with other related code bases. Accordingly, a build system according to the subject technology may use Master and Slave components. For example, the Master component may monitor the source repository and coordinate the activities of the Slave component(s), and report results. The Slave component(s) may be responsible for building binaries and/or running tests. Figure 1, below, depicts an example data flow of an example build system, and how the system may be integrated with the Mediator via a private Mediator API.

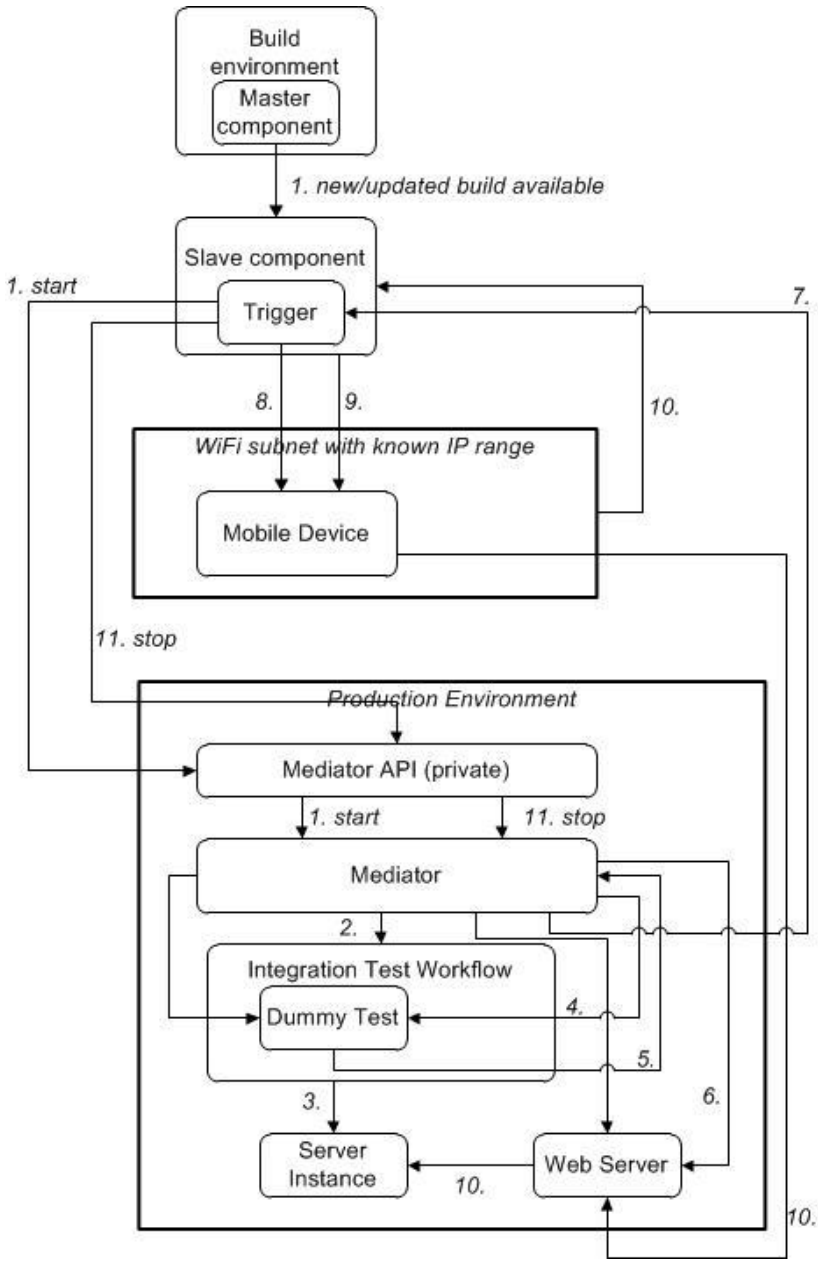

Figure 1.

The Mediator makes it possible to run web-based operating system tests against unreleased back end applications in a secure way even though the devices may not connected to an authorized network, the two code bases may be in separate repositories, or the two code bases may use different infrastructures. The tests may also be run on the same infrastructure as all other operating system tests, making it easy for developers to monitor and debug the tests.

The Mediator provides a bridge between the build infrastructure and production services (e.g., production back end servers and applications). It may be implemented as a production

service and, as depicted in Figure 1, may expose a private Mediator API (e.g., only available to internally authorized developers). In some implementations, the Trigger may communicate through the private API using authenticated requests. The Mediator API may be implemented as a "stubby" service running in a production server.

The Mediator API enables developers to trigger Integration Test Framework (ITF) Workflows, update workflow objects and end (e.g., delete) the Workflows. The Mediator API is private and may require authentication using security credentials associated with a known account to avoid unintended access and to avoid leaking internal URLs. Users may also be authenticated using credentials. Credentials may be stored, for example, on a machine running the Slave component(s). Credentials for the account(s) may be stored, e.g., on servers hosting the Slave components. The Mediator may be made publicly available; however, security features prevent making back end URLs (e.g., within a development environment) available to the public (which may risk exposure of unreleased code or features).

As depicted by Figure 1, a Slave component is triggered by changes or a build being available (1). The Slave component downloads binaries containing the application(s) under test and installs them on the mobile device, machine or Virtual Machine and performs general device setup (2). The Slave component uses a trigger module to send a "create workflow" request to the Mediator (3). In this regard, the trigger module sends the request to the private Mediator API. The request may contain, for example, a unique identifier for the request:

```
<build_app_name>
<buildnumber>
<testsuite>
```

The request may also contain, for example, the integration test framework (ITF) Workflow that should be started, and the revision to use for the ITF Workflow. The request may be authenticated using a service account stored on the actual machine running the Slave component.

The Mediator then sends a request to ITF Workflow to start the appropriate workflow, passing the unique request identifier to the Workflow (4). The ITF Workflow starts instance(s) of the back end applications and runs a "Dummy Test" that sends a request to the Mediator when it starts to let the Mediator know that the Workflow is up and running, and continuously polls the Mediator whether the Workflow should continue to run (5). The mediator will reply "Yes" until it receives a "Delete workflow" request from the Trigger. The Mediator uses one or more URL

service clients to send requests to a Web Server to register a unique URL with the backbone network service (BNS) address for each server started by the ITF Workflow (6).

The Web Server makes it possible to dynamically reconfigure hostnames to point to a specific backbone network service (BNS) (or host:port). The Web Server uses a server where requests carry an endpoint (e.g., host:port or BNS) as an argument, and it allocates a currently unused web server hostname and configures it to point to that endpoint. Any DNS names with a predetermined pattern may resolve and get dispatched to the service. Without the Web Server, the mobile devices cannot connect to the proprietary back end instances started by the ITF Workflow. Since the Web Server only allows connections from certain IPs, a WiFi lab with a known IP range may be used, with the range whitelisted on the Web Server, and the Mobile devices may connect to this WiFi to allow them to use the Web Server to connect to the proprietary back end servers.

The Mediator returns the map of back end applications to URLs to the Trigger which has been blocked upon getting this response (or potentially timed out) from the Mediator (7). The Trigger sets up the appropriate mapping on the device (for example by setting command line flags) based on the response from the Mediator (8). The Slave component starts the tests on the device (9). When the tests are done, the Slave component uses the Trigger to send a "Delete Workflow" request to the Mediator but this time without waiting for a response (10). The next time the Dummy Tests polls the Mediator, it will get a response that the workflow should terminate and the test will terminate with a passed status (independent of whether the integration tests passed or failed) (11). The ITF Workflow will stop at this point and the servers will be brought down. The Slave component then makes the test results available on a page associating with the build application (12).

DM_US 61459583-2.096553.0123