



Getting Started with Automated Test Framework (ATF)

NowForum Sydney
9th October 2018

Gareth Oliver

Advisory Solution Consultant



Mark Graham

IT Developer Support Officer



Agenda

➤ Why ATF? What is it for?

➤ Customer stories

➤ How it works

➤ What is possible (and what is not)

➤ Planning for ATF

➤ Best Practice

➤ Development in ATF

➤ Hands on labs

Speaker introduction



Name: Gareth Oliver

Title: Advisory Solution Consultant

Company: ServiceNow



Experience/Expertise: Over 6 years working with and 4½ years working at ServiceNow. Born and raised in NZ And now living in sunny Brisbane, QLD

Function: Helping customers solve business issues with the ServiceNow platform

Fun Facts: Scout leader, soccer player, SCUBA diver

Speaker introduction



Name: Mark Graham

Title: IT Developer Support Officer

Company: Cairns Regional Council



Experience/Expertise: Have has SN for 3 years, Developing apps for 2. Primary focus on Service Portal/Scripting

Function: Using SN to solve business problems, Developing app to fill gaps in existing systems

Fun Facts: Learning Spanish, Cellist for 15 years

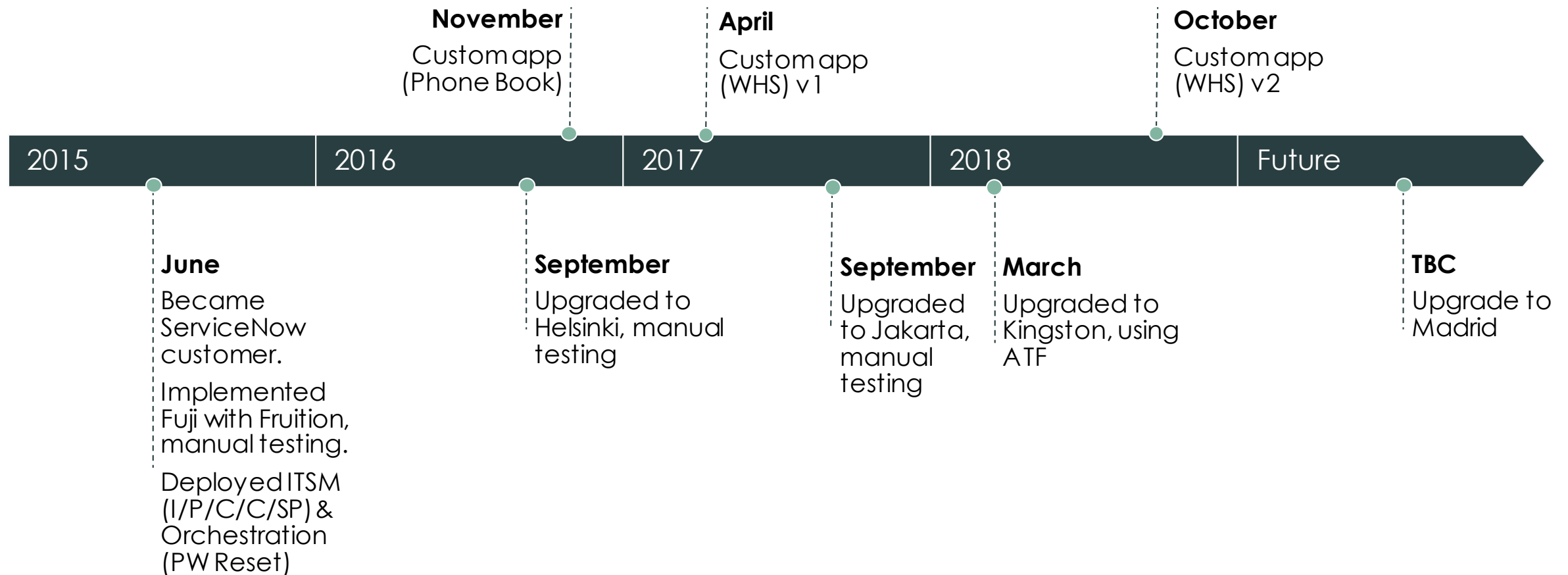
About Cairns Regional Council

- 8th largest LGA in QLD
 - ~1,300 employees
- Approx population 165,000
- Major industries include
 - Tourism (Great Barrier Reef, Atherton Tablelands)
 - Healthcare
 - Public service
 - Agriculture (sugar cane)
 - Education (JCU, TAFE QLD North)



Our ServiceNow journey

- Council decision to skip a major version
- 3 part-time SN developers (internal)



Massive thanks, credit



Name: Geoffrey Sage

Title: Senior Technical Consultant

Company: ServiceNow

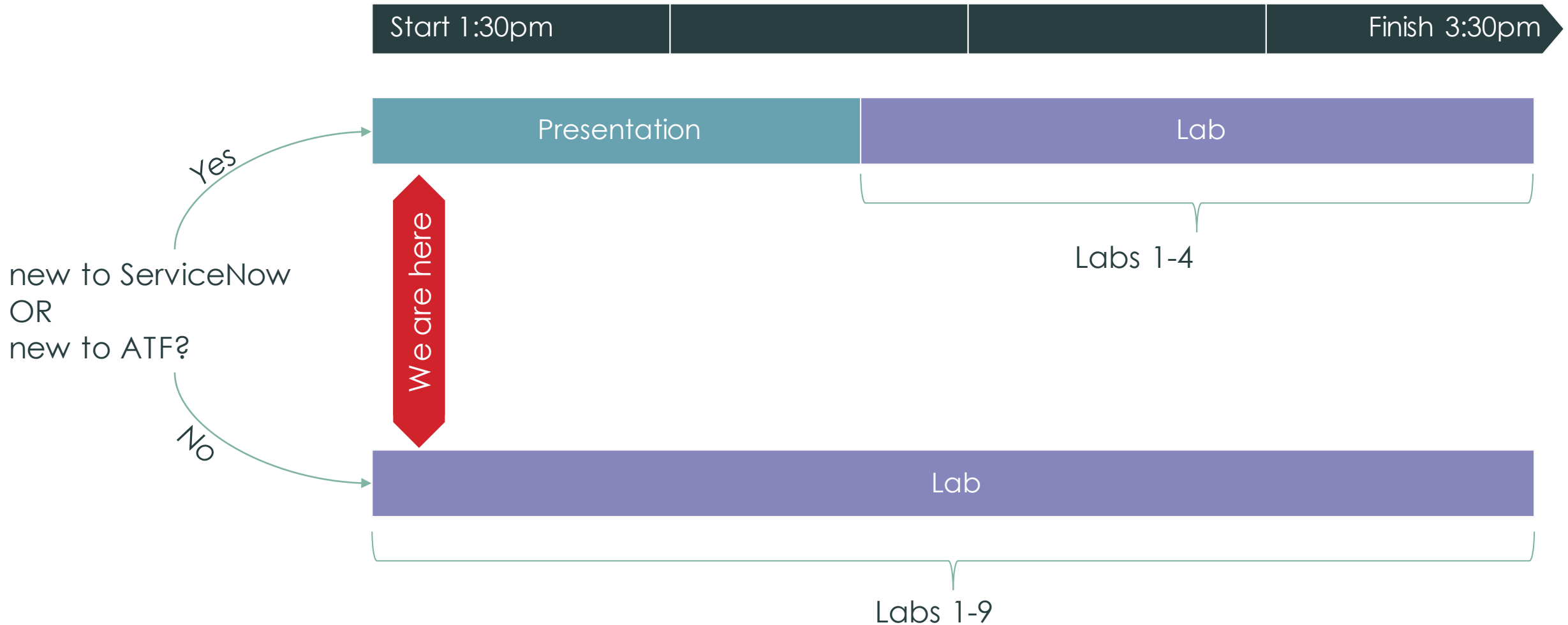


Buy this man a beer!



This presentation is a shortened version of a presentation Geoffrey & PS team made to Sydney & Melbourne Developer Meetups

Next two hours....



Why ATF?

Lets here it directly from the ServiceNow
Product Manager!



Play video here

Problem

- Upgrade testing consumes over 25% of time and resources for many customers
- Customer test frameworks are broken by each major upgrade or UI change
 - When using tools like Selenium (DOM changes)
- Most customers do not use an automated test framework – they test manually
- “Our last major upgrade took 13 weeks”



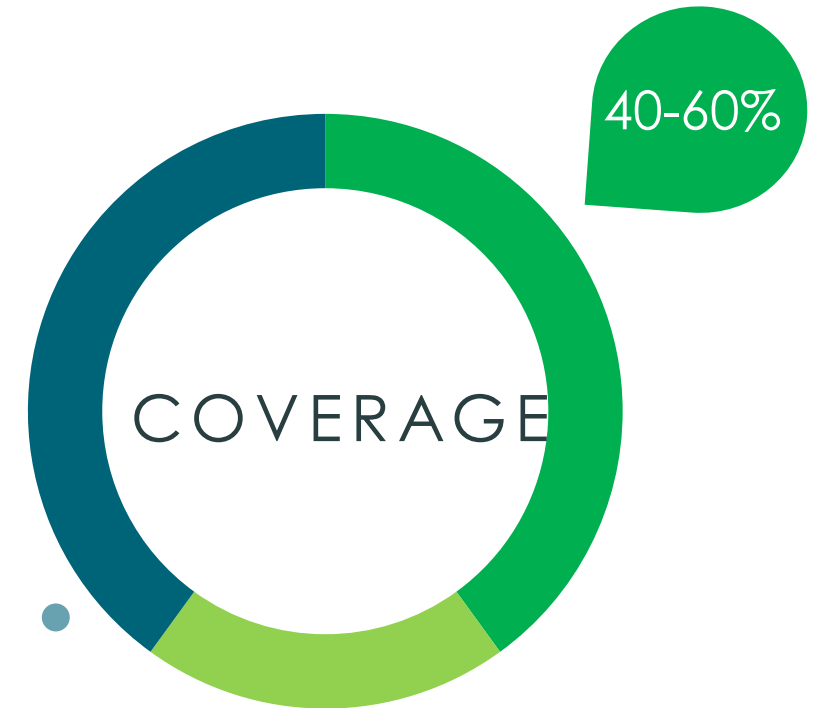
Upgrade Testing
Time & Resources

ServiceNow Product Lab Research

- ~ 300 customers responded
- Said testing & verification is #1 barrier to upgrading
- >50% of time spent on time doing regression testing

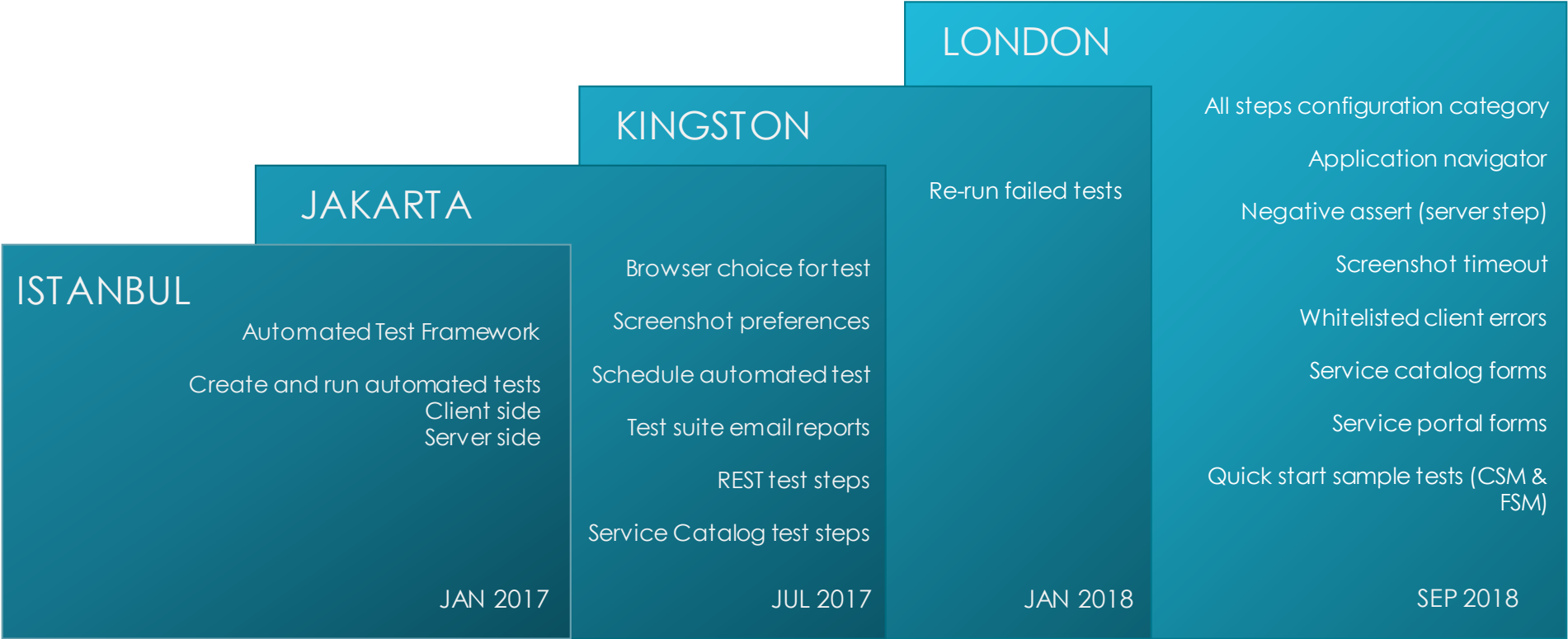
ATF Goals

- Framework to automate most of the manual test
- Make the framework UI independent
- Expect 40-60% of Tests covered
 - Most of the delta is in client-side functionality
- Platform feature



- 6 weeks for previous upgrade (to J)
- Used ATF to upgrade to Kingston
- 200 tests in Excel; took ~1 week to put into ATF; about 90% coverage of all tests
- Result: took 1 week to upgrade to Kingston, taking 5 weeks out of the process

Constant stream of innovation



Q1 2019

Information Sources



Search for

Docs & Community:
“Automated Test Framework”

YouTube: “ServiceNow Automated Test Framework”

What is ATF for?

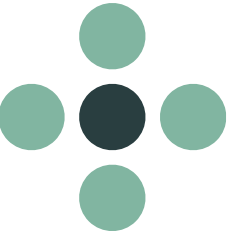


What ATF Is For

- ATF is intended for **Regression Testing**.
 - That means retesting functionality that you know already works, to make sure that something hasn't broken it.
 - Use it after an Upgrade to make sure that your existing functionality is still working.
- ATF is **NOT** meant for testing new functionality.
 - Why? ATF takes time and effort to develop.
 - Why? Its value is in its repeatability.
 - Why? You don't want to put yourself in a situation where you develop an ATF Test for functionality that is still being built. If that new functionality is changed, then you need to redevelop your ATF Tests. You end up causing additional development in ATF.
- When you are building new functionality, you should put it through your normal, manual test and release cycle.
 - Then you would build an ATF Test to ensure that the new functionality doesn't break in the future.

When to Develop ATF

- When should ATF be developed for **new functionality**?
 - Ideally, ATF Tests should be developed **after UAT** has completed.
 - ATF is **not** meant for Unit Testing, or User Acceptance Testing (UAT).
 - You want to ensure that the functionality you are testing with ATF is **stable**.
- Developing ATF in parallel with the new feature will likely result in having to redevelop the ATF Tests.
 - You need to make sure that the new features aren't going to be updated any time soon, otherwise you're going to increase your ATF development costs.
 - If you are using an Agile methodology where new development is constantly evolving, then you should wait until you development iterations (Sprints) end.
- In practice, if the people developing the new features are the same people developing the ATF Tests, then UAT is a good time to start ATF since the development workload is dropping off and changes to the new features should (hopefully) be minimal.



ATF Scope

- ATF is a **platform-wide**, global, feature
- ATF is **not limited** to any product, app or module
- The standard **form view** of (almost) any record can be opened in the Client Test Runner (client-side)
- There are **no limitations** on the server-side
- ATF can be used to test **any** scoped apps, custom apps, or out-of-the-box products
- ATF can be used to test **Service Portal** (limited in Kingston, more in London)
- ATF can test custom features, but Hi Support probably won't support troubleshooting ATF issues testing custom features
- Uses **Global** scope

NEW

Now Platform™

Notify

Visual Task Board

Connect

Push

SMS

Service Portal

Email

Mobile

Guided Tours

User Experience & Notification Services

Intelligent Automation

NEW

Virtual Agent

Anomaly Detection

Performance Forecasting

Supervised Machine Learning

Developer Instance

Git Integration

Automated Testing

Studio IDE

Code Sharing

Delegated Development

Portal Widget Editor

Script Debugger

Dev Tools

Service Creator

Form Designer

NEW

Virtual Agent Designer

Flow Designer

Portal Designer

UI Policies

Table & Schema Builder

No/Low Code

AJAX

Bootstrap

HTML/CSS

JavaScript

Angular JS

Pro Code

Reporting Engine

Orchestration

Peer Benchmarking

CMDB

Knowledge Management

Time Series Database

Service Catalog

Workflow Process Engine

Application Scope

Business Rules

Core Services

Encryption

Tokenization

2-Factor Authorization

OAuth 2.0

SAML

SSO

LDAP

ACLs & Roles

Security

Integration Services

Scripted APIs

API Explorer

JSON

SOAP/XML

REST

Integration Hub

MID Server

Import & Export

ATF Licensing

- ATF is a **platform** feature
- ATF is **part of your** ServiceNow subscription
- **No additional license** cost is required**
- Where is the cost?
 - Your time to develop the ATF use cases
 - Are your ATF Admins & Test Designers new users?

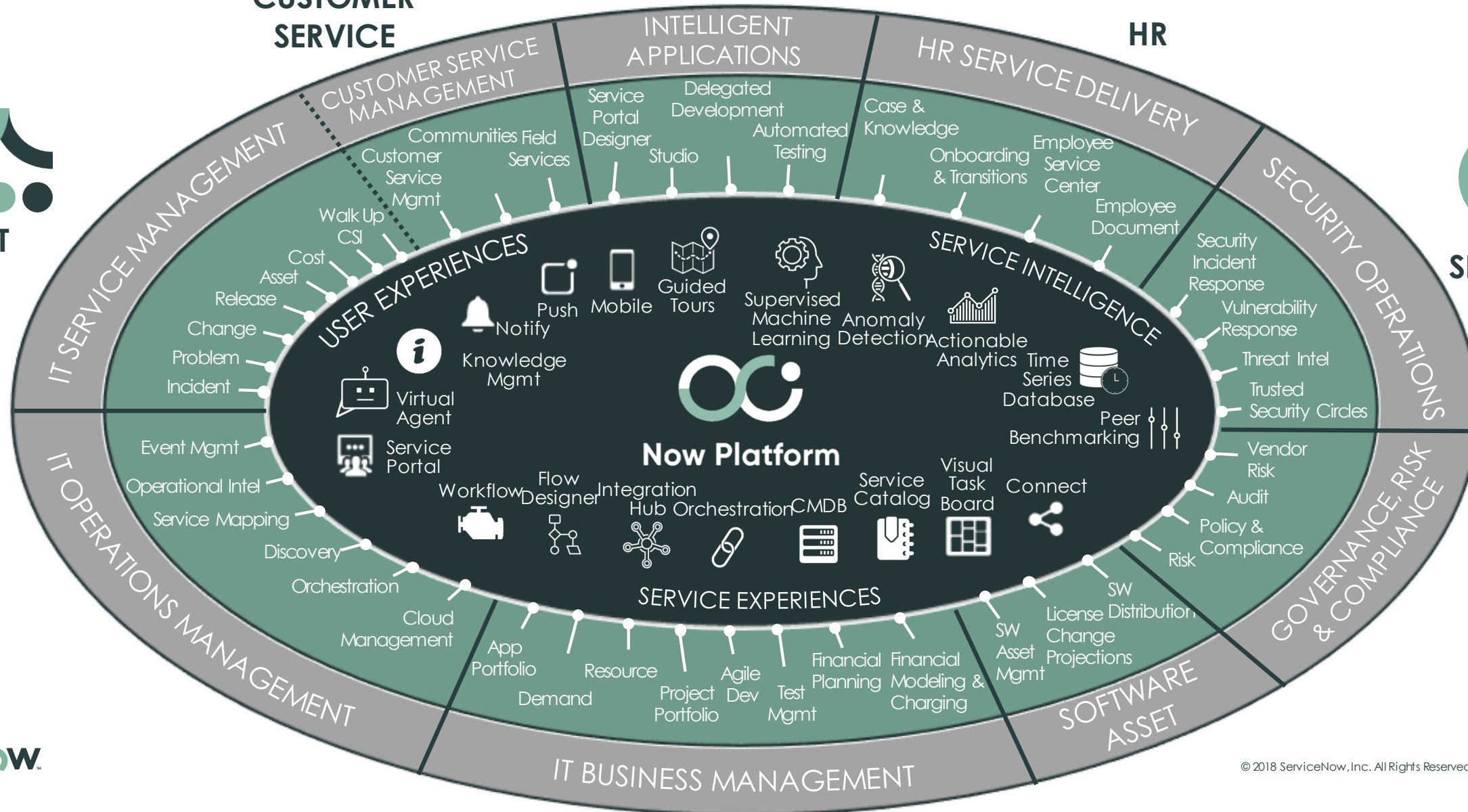
****** *If your ATF users **only** have an ATF role, they consume an application 'Fulfiller' license (i.e. ITSM, ITBM, CSM)*



**CUSTOMER
SERVICE**

INTELLIGENT APPS

HR

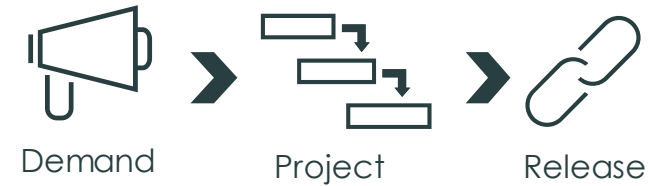




ATF and existing SDLC Products/Processes

- Q: How does ATF fit into existing ITBM related products/processes?

- SAIF/SIM
- Release Management
- Agile Development
- SCRUM
- Test Management



- A: ATF is not currently built into any existing processes

- None of the existing processes cater for Regression Testing
- ATF requires its own development lifecycle
- WORKAROUND: Add a **Test Management** task to run ATF
- WORKAROUND: Add an **ATF Test** field to Stories

Customer story: Cairns Regional Council

ATF at Cairns Regional Council

- Used for the J > K upgrade
- Why?
 - Patches / Upgrades
 - Patch verification process is long and arduous
 - Happens fairly regularly
 - Wanted to lower the overhead with patches/upgrade
 - Will allow us to be more up to date, more frequently
- Outcome
 - We saved around 1.5 days of effort using only basic tests
 - We could expand our testing to be much more comprehensive
 - Would probably lose value in the time it takes to create the tests
 - Slower more organic growth seems more valuable
 - Add new tests as new applications are developed/used
- Findings
 - ATF has limitations (found in Jakarta)
 - Could only move test cases that were relevant to entered data
 - Forms
 - Forms data verification
 - Business rule/client scripts that alter data
 - Only applied to less than a dozen of our tests
 - Most of our testing is around customisations
- How?
 - Moving from manual
- Time Saving
- Future
 - Develop tests for our custom applications
 - Expand testing to use REST endpoints

Customer story: AC3

AC3 used ATF to upgrade J to K in 10 days

WHAT WE LEARNED BY UPGRADING SERVICENOW IN 10 DAYS

Posted: 4 April 2018

“...we reduced manual testing time significantly, down from two people testing for eight hours, to one person for three hours.”

*“While the ATF did a lot of the heavy lifting for us, it also had to be **underpinned by a robust, pragmatic upgrade plan.**”*

- Google “ac3 servicenow atf”

<https://www.ac3.com.au/about-us/news/what-we-learned-upgrading-servicenow-10-days>



Customer story: ServiceNow “Now On Now”

Presented at Knowledge18 by & all credit to:

Prosenjit Sengupta

Quality Engineering Manager, IT Applications Development, ServiceNow

Business requirements



Typical challenges



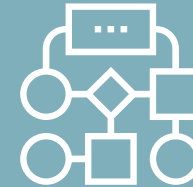
Velocity

- Catch-up mode
- Specialized test automation engineers
- Transparent test result analysis
- Framework update with every platform upgrade



Reliability

- False failures during upgrade testing
- Frequent flappers
- Wait synchronization issues
- Test data management

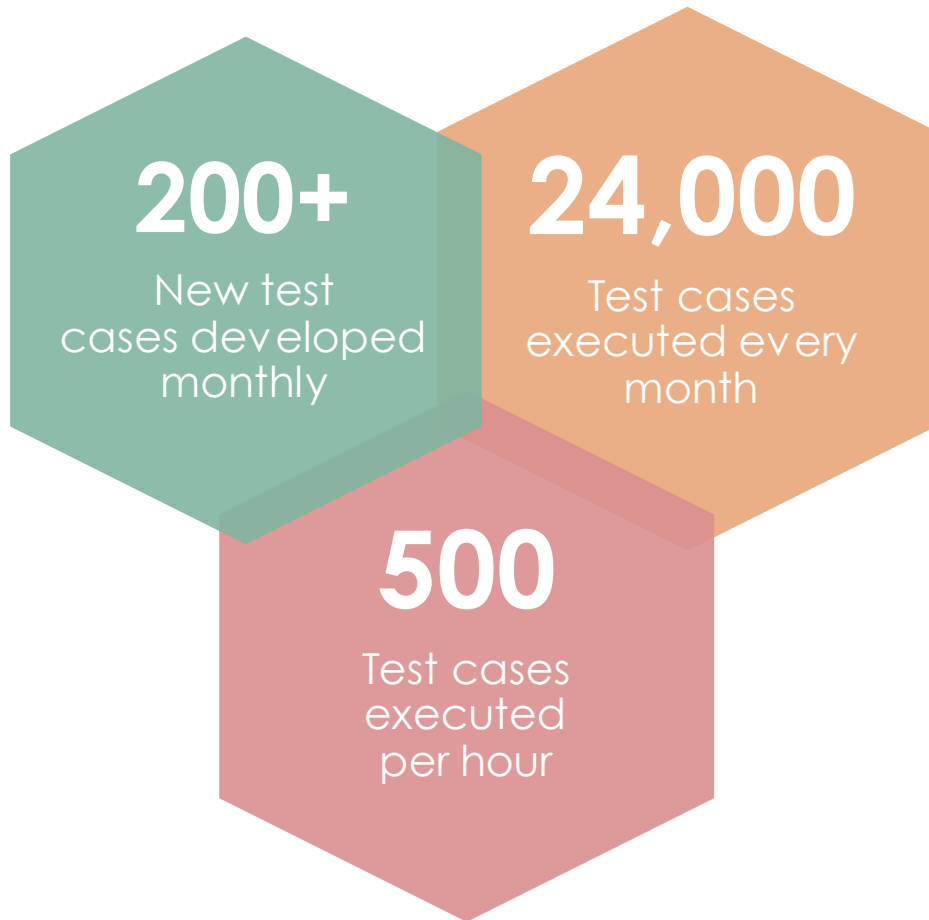


Complexity

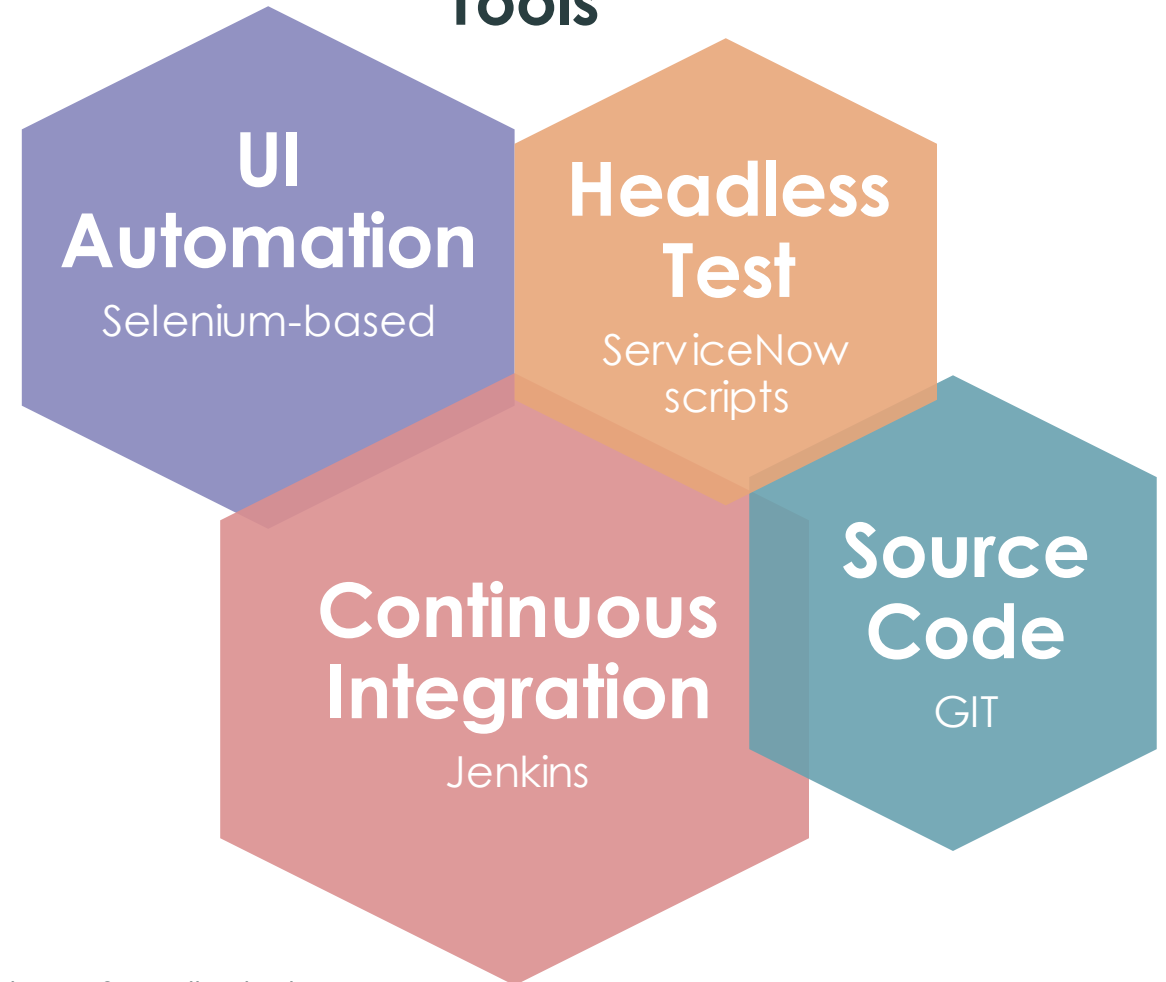
- Multiple test automation platforms
- Integration tests via separate Rest Soap layers
- Separate suite of single-user performance tests

Automated tests development and execution

Volumes



Tools



Note: 1 test case maps to 5–10 smaller tests

Solution components



Orchestration

- ATF remote test runner plugin
- ATF remote test result analyzer



Continuous integration

- Scheduled remote test runner
- Unattended test runner via Jenkins integration



Test library

Reusable test configs

- Server
- REST



Source control

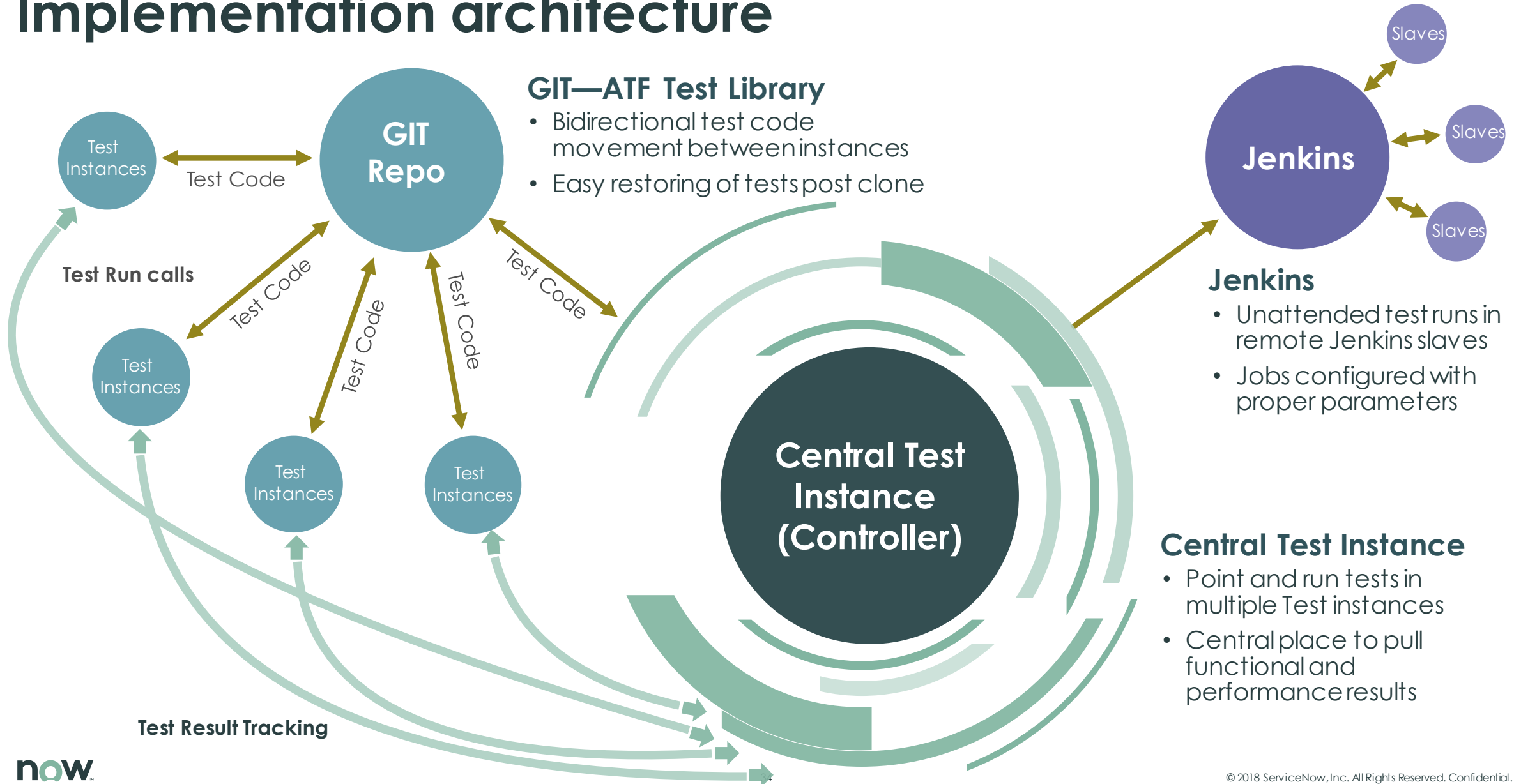
- GIT Integration
- Custom app for code movement between instances



Performance

- Performance result analyzer for each test step
- Performance trends

Implementation architecture



Lessons learned

- 1 Ensure reusability of test steps
- 2 Conduct integration with source control
- 3 Include test logs for easy debugging
- 4 Employ a mix of UI and headless tests
- 5 Use REST steps for integrations
- 6 Set automated test properties suited to business needs

How it works

Configuring ATF

1. Create a Test Suite (optional)

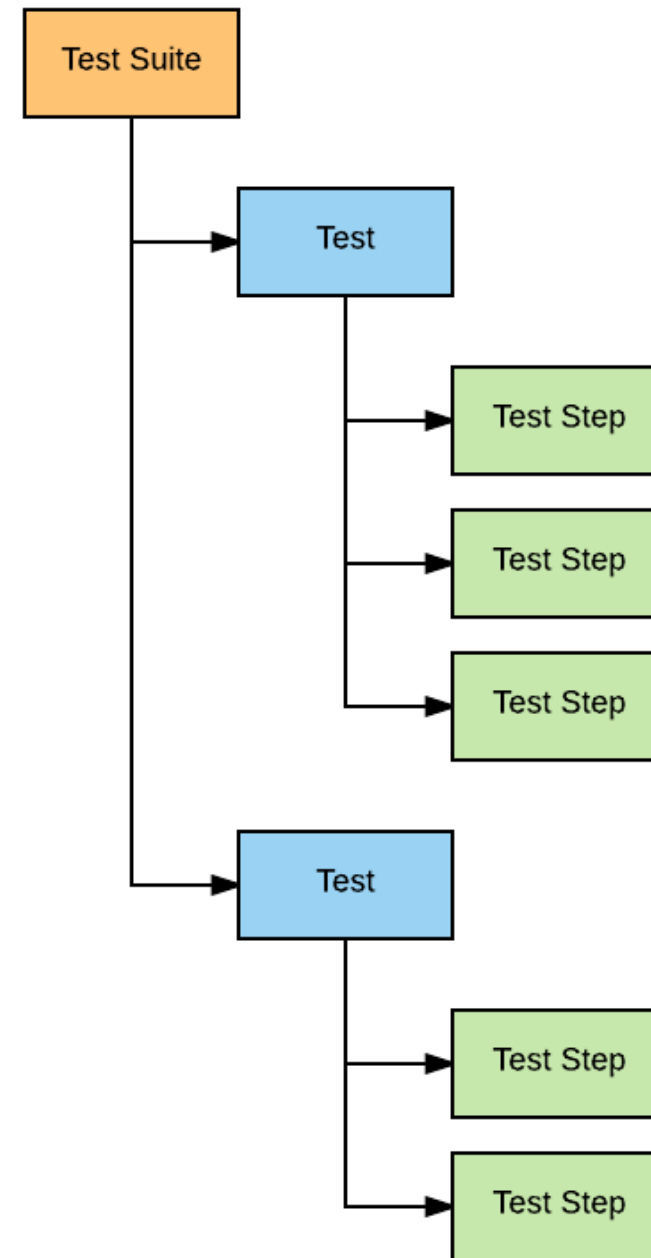
- Represents a set of Test Cases
 - Example: Service Catalogue
- Allows you to execute many Tests with one click

2. Create a Test

- Represents a Test Case
 - Example: “Adobe Acrobat Pro” Catalogue Item

3. Add Test Steps

- Represents a single action in a Test Case
 - Example: Load the Catalogue Item, Populate Variables
- Where the main functionality is
- Uses existing **Step Configurations**



ATF Roles

- **Admin** (atf_test_admin)
 - Create Step Configurations (custom Test Steps)
 - Create Test Templates
 - Update System Properties
- **Test Designer** (atf_test_designer)
 - Build Tests
 - Run all Tests and Test Suites
 - Use the Client Test Runner
 - No coding required
- **Web Service Test Designer** (atf_ws_designer) (Jakarta+)
 - Can build REST web service tests
 - Gives access to other Web Service modules

Automated Test Framework
Tests
Suites
Test Results
Suite Results
Schedules
▼ Run
Client Test Runner
Scheduled Client Test Runner
Active Manual Test Runners
Active Scheduled Test Runners
Waiting/Running Test Runs
Waiting/Running Suite Runs
▼ Administration
Properties
Step Configurations
Test Templates
Step Configuration Categories
Table Cleanup
All Test Runners

Test Steps

- There are 2 Test Step **Environments**

- **Server**

- Runs in the background on the server (only visualised as a progress bar)

- **UI** (client-side)

- Runs in Client Test Runner (in the browser)

- **Categories**

- Forms in Service Portal **(UI)** (London)
 - Service Catalog in Service Portal **(UI)** (London)
 - Application Navigator **(UI)** (London)
 - Form **(UI)**
 - Service Catalog **(UI)**
 - REST **(Server)**
 - Server **(Server)**

Add Test Step


Category	Step Configuration	Description
All Steps	Form	Category
Forms in Service Portal	Open a New Form	Form
Service Catalog in Service Portal	Open an Existing Record	
Application Navigator	Set Field Values	Open a New Form
Form	Field Values Validation	Opens a new form for the specified table.
Service Catalog	Field State Validation	Additional Considerations
REST	UI Action Visibility	Optionally, you can specify the form's view name. Keep in mind that this can only be done for users that have access to that view.
Server	Click Modal Button	
	Click a UI Action	
	Submit a Form	

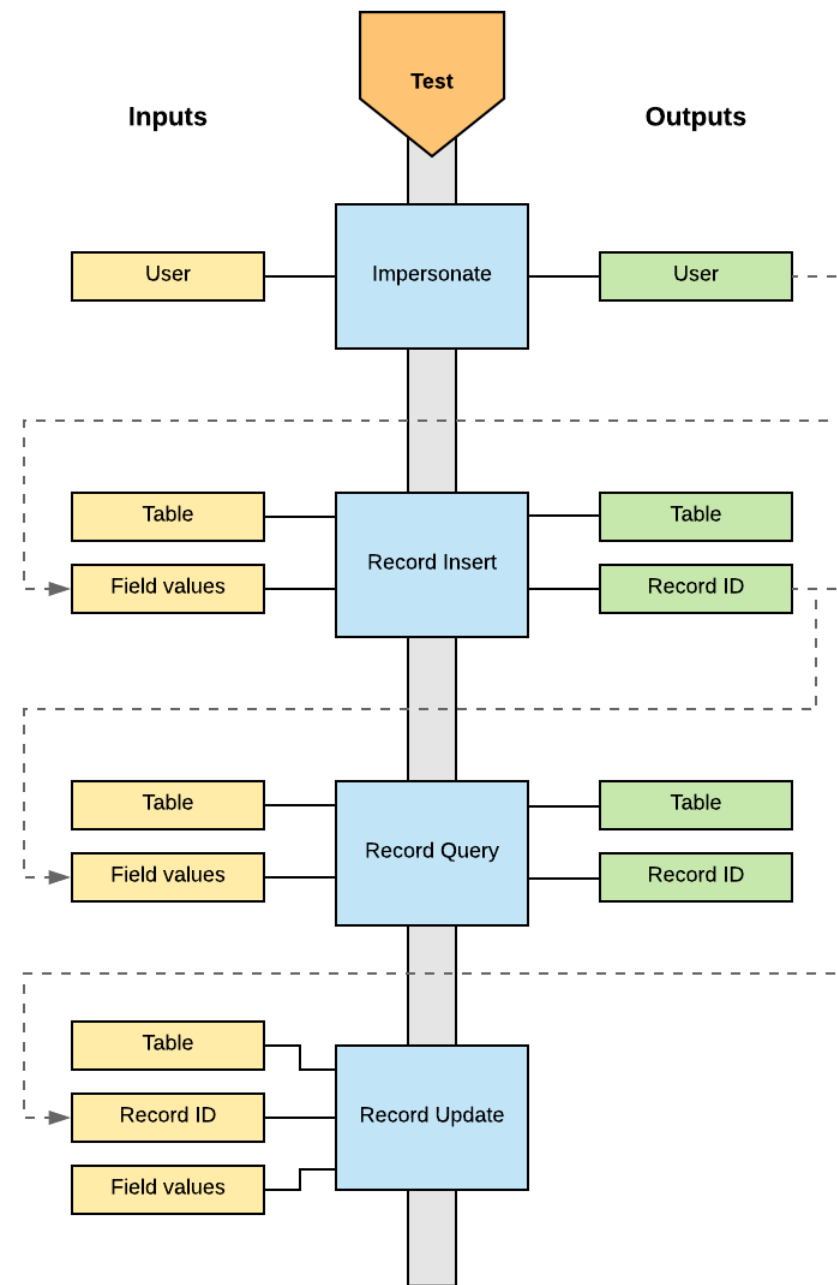
Execution Order Insert after **Step 5 - UI Action Visibility** **Next**

Step Configurations

- Test Steps use functionality as defined in **Step Configurations**
 - Step Configs contain the code that drives the Test Step
 - Test Designers can create Tests without code as long as there is an existing Step Config to perform the function
 - Admins can create custom Step Configs for server categories (not UI)

Test Steps: Inputs & Outputs

- Test Steps have **Input** and **Output** variables:
 - **Inputs** are populated by the Test Designer when the Test Step is created. It is the data that is being used for the Test.
Example: Field values
 - **Outputs** are generated when the Test Step is executed. They can be used as inputs into following Test Steps.
Example: sys_id for a new record
- Data Pill Picker 
 - Used to select an Output (Data Pill) from a previous Test Step as an Input for the current Test Step
 - Quite flexible in allowing you to mix Reference and Document ID fields



Execution order

Active ☒

Description Open the 'Change Request' form with id '{{Step 6: Submit a Form.Record}}'

Application Global

* Test Change Request - Form and Workflow

* Step config Open an Existing Record

Variables

* Table Change Request [change_request]

* Record Step 6: Submit a Form > Record X Data pill document_id

View

document_id

Step 6: Submit a Form.

will

Step 1: Impersonate	glide_var	>	Record	document_id
Step 2: Generate Relative Date/Time	glide_var	>	Table	table_name
Step 3: Generate Relative Date/Time	glide_var	>		
Step 6: Submit a Form	glide_var	>		

At Step 7, use an output from step 6

UI Step Configurations (London) 1/3

Forms in Service Portal:

- Open a Form (SP)
- Set Field Values (SP)
- Field Values Validation (SP)
- Field State Validation (SP)
- UI Action Visibility Validation (SP)
- Click a UI Action (SP)
- Submit a Form (SP)

Service Catalogue in Service Portal:

- Open a Record Producer (SP)
- Open a Catalogue Item (SP)
- Set Variable Values (SP)
- Validate Variable Values (SP)
- Variable State Validation (SP)
- Validate Price and Recurring Price (SP)
- Set Catalogue Item Quantity (SP)
- Add Item to Shopping Cart (SP)
- Order Catalogue Item (SP)
- Submit Record Producer (SP)

UI Step Configurations (London) 2/3

Application Navigator:

- Application Menu Visibility
- Module Visibility
- Navigate to Module

Form:

- Open a New Form
- Open an Existing Record
- Set Field Values
- Field Values Validation
- Field State Validation
- UI Action Visibility
- Click Modal Button
- Click a UI Action
- Submit a Form

UI Step Configurations (London) 3/3

Service Catalogue - NOT Service Portal:

- Open a Catalogue Item
- Open a Record Producer
- Set Variable Values
- Set Catalogue Item Quantity
- Validate Variable Values
- Variable State Validation
- Validate Price and Recurring Price
- Add Item to Shopping Cart
- Order Catalogue Item
- Submit Record Producer

Server Step Configurations (London)

Server:

- Impersonate
- Record Insert
- Record Update
- Record Delete
- Record Query
- Record Validation
- Replay Request Item
- Run Server Side Script
- Log

REST:

- Send REST Request - Inbound - REST API Explorer
- Send REST Request - Inbound
- Assert Status Code
- Assert Status Code Name
- Assert Response Time
- Assert Response Header
- Assert Response JSON Payload Is Valid
- Assert Response XML Payload Is Well-Formed
- Assert XML Response Payload Element
- Assert JSON Response Payload Element
- Assert Response Payload

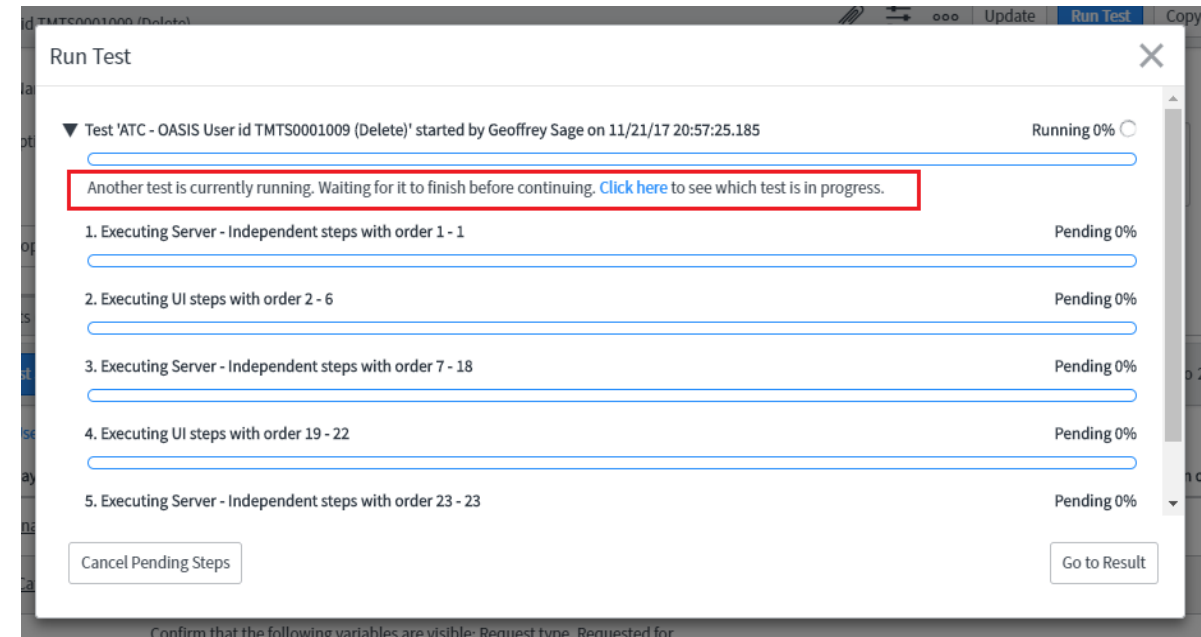
Run Test

- When you run a Test, each Test Step will be executed in order until the Test finishes, or a Step fails.
- If a Step fails, all following Steps in that Test are skipped.
- Failures are rolled up to the Test and the Suite.
- When running a Test Suite, a failure in one Test will not affect any other Test. Every Test in the Suite is executed.
- For each UI Step, a screenshot is taken and attached to the Test Result record (depending on configuration).
- Results are recorded in the Results tables:
 - Step Result (has Summary of success or failure)
 - Test Result
 - Suite Result

Execution order ▲	Step	Status	Summary
1	Impersonate	Success	Impersonated ATF ESS User
2	Open a Catalog Item	Success	Successfully opened the 'catalog item'
3	Variable State Validation	Success	Successfully validated all field assertions.
4	Set Variable Values	Success	Successfully set field 'IO:4a24c39ddb92cb00a9e6788bbf961919' to value 'grant'
5	Validate Variable Values	Success	Variables correctly matched the condition.
6	Order Catalog Item	Failure	FAILURE: Order Now button was not found. See screenshot
7	Record Query	Skipped	This step did not execute due to a failure in a previous step
8	Record Validation	Skipped	This step did not execute due to a failure in a previous step

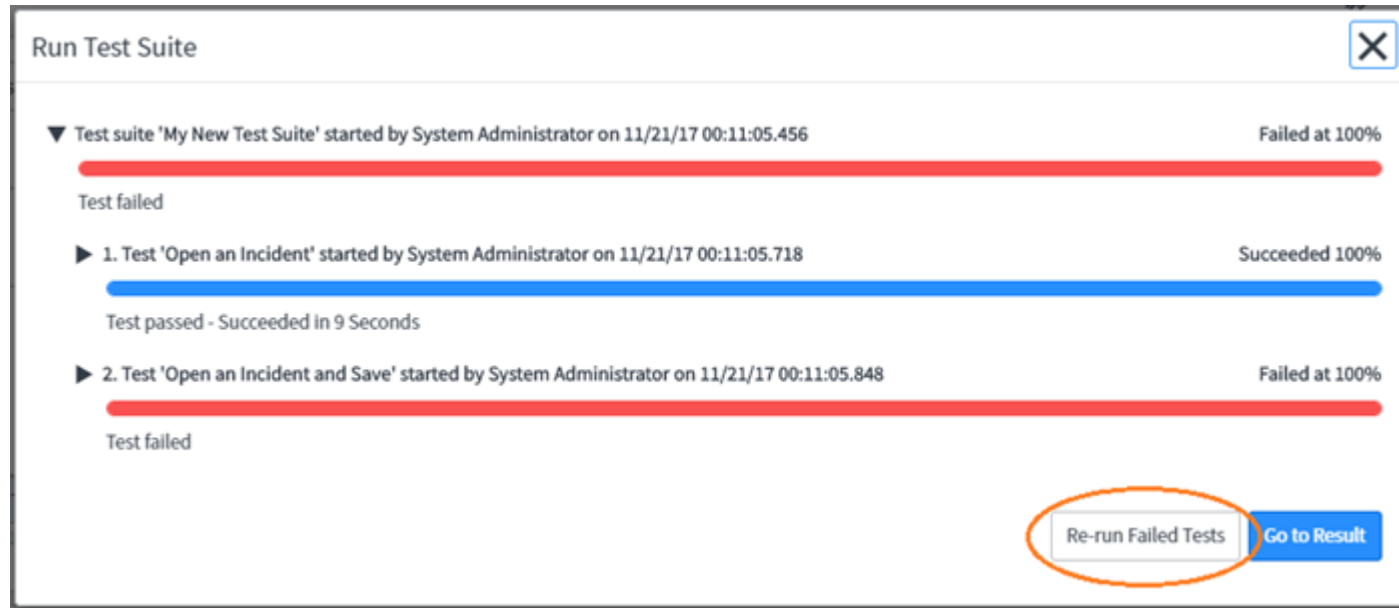
Run Test

- Only 1 Test can be executed at a time
- All Tests in a Suite are executed sequentially, not concurrently
- Only really an issue when you have multiple developers building different Tests simultaneously
- This is because of the rollback feature. If multiple Tests were running at one time, the rollback from one Test would break the execution of other Tests



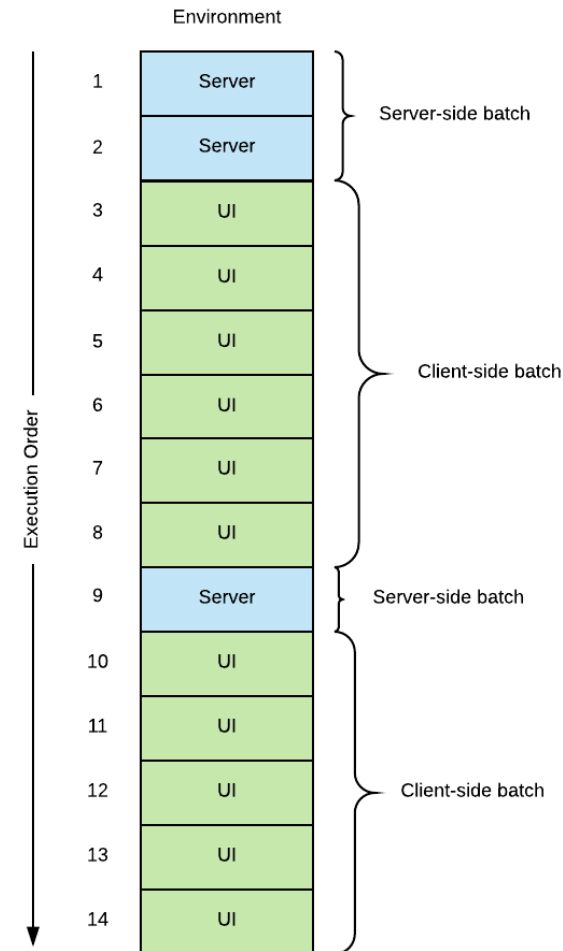
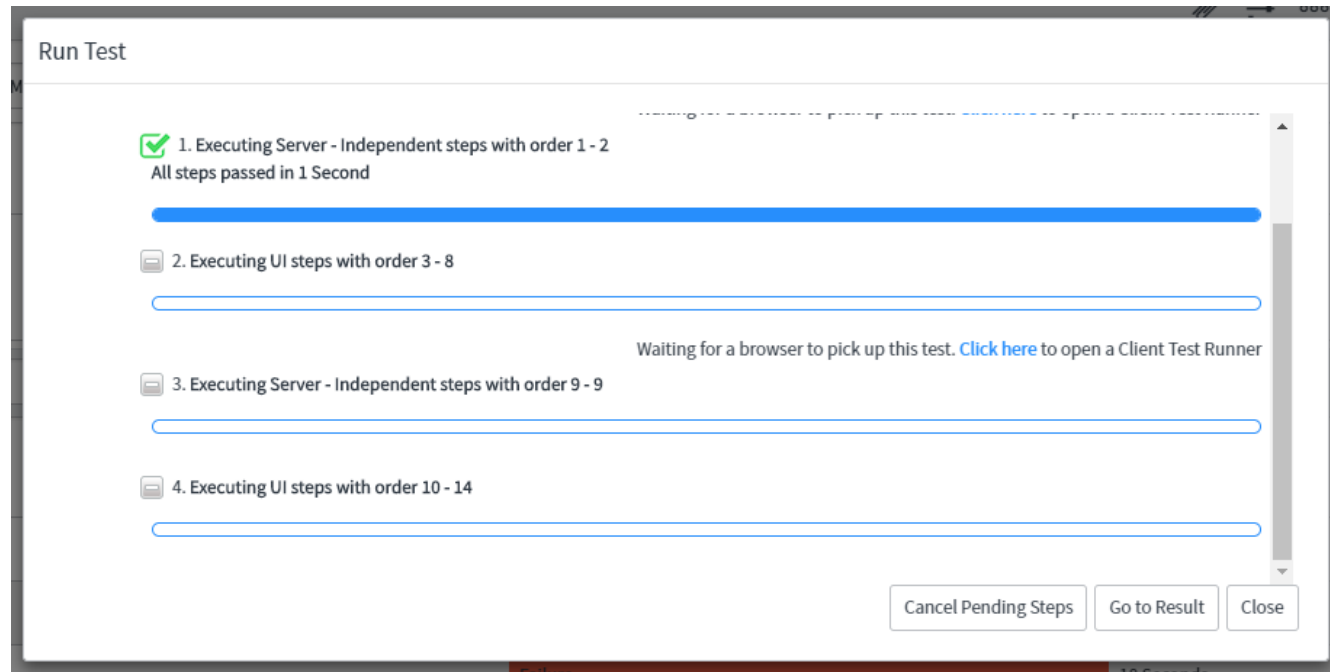
Re-run Failed Tests (Kingston+)

- Allows you to re-run only the Tests that failed, and ignore the Tests that passed
- Assumes that you have addressed the cause of the failure



Batching

- Server and UI Test Steps are batched (grouped) according to **Execution order**
- **Client Test Runner** executes a batch of UI Test Steps
- Server steps are run in the background (you only see a progress bar)
- Multiple Client Test Runners can mean different batches are run on different PCs
- You will need to load a form at the start of each UI batch



Client Test Runner

- All UI (client-side) Test Steps are executed in the Client Test Runner
- Must have the **atf_test_designer** role to use it
- Should be the active tab or in a separate browser instance
 - Browsers deprioritise background tabs
 - takes longer to run
- All open Client Test Runners poll the server for UI test batches
- Significant performance difference between browsers
- **Pre-London** - the Navigation bar, Banner and Related Lists are not visible, so we can't access them in our Tests

UI Test Runner

Performing other activities while tests are being executed can result in those activities being performed by the currently executing user and could be rolled back.

Running step 2 of 5 for Incident - MIM AU

Your connection status is [Connected]
Currently executing as [Service Desk (AU)]
UI Batches Executed [2]

Taking screenshot and sending it to server

Execution Frame Debug Info

Incident - INC0011189

Reference Number: INC0011189

* Impacted colleague: ESSTest1 SYD

Full address: Change Manager (NZ)

Contact Number: Danielle Hunter, Danielle.Hunter@iag.com.au

Alternate contact No.: ITIL User (AU), ITILUser_Aus_Dev@iag.com.au

Asset number: ITIL User (NZ), ITILUser_NZ_Dev@iag.co.nz

* Impact: 1 - High

* Urgency: 2 - Medium

Priority: 2 - High

MIM Managed: MIM AU

* Short description: Donec eget nunc at magna volutpat tristique eu in lacus

Description: Aenean turpis neque, pharetra aliquam metus ut, porta ullamcorper risus. Aliquam placerat metus at orci suscipit, vel pretium augue interdum. Quisque quis euismod dui, blandit vestibulum risus. Cras eget lacinia tortor. Sed at elit nibh. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Vivamus fermentum vestibulum nisl non ultricies. Nam semper nisi massa, nec tristique risus gravida eget. Aliquam erat volutpat.

State: Open

Contact type: Phone

* Service Item: ADICS

Configuration item:

* Category: Incident

* Subcategory: Unavailable

* Assignment group: SERVICE_DESK_AU

Assigned to:

Vendor Ref No.:

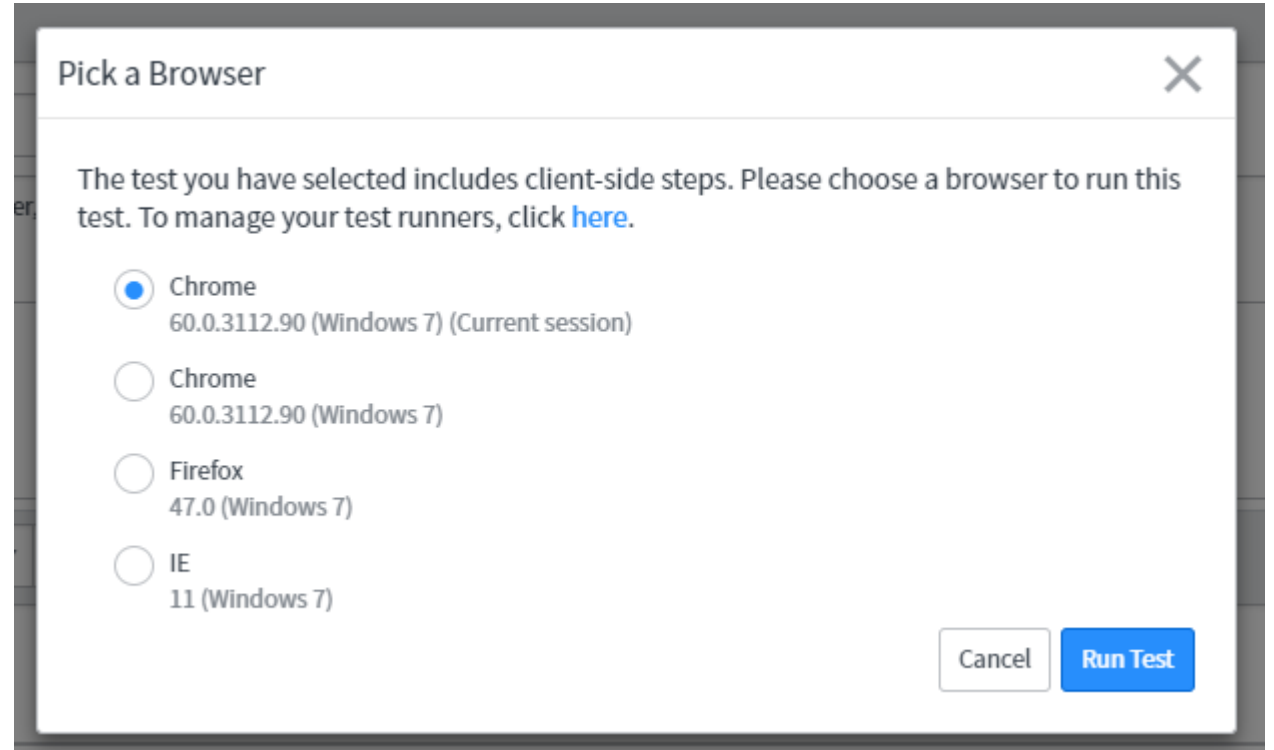
FOFC: ☐

Related Search Results

No matching results found for Donec eget nunc at magna volutpat tristique eu in lacus

Pick a Browser (Jakarta+)

- Client Test Runners are visible on the **Test Runner** [sys_atf_agent] table
- You must choose a Test Runner when you Run a Test
- **Istanbul** – You can't choose a browser. Pot luck between all Client Test Runners currently polling queue on the server



The screenshot shows a 'Pick a Browser' dialog box with a close button (X) in the top right corner. The main text reads: 'The test you have selected includes client-side steps. Please choose a browser to run this test. To manage your test runners, click [here](#).' Below this text are four radio button options: 'Chrome 60.0.3112.90 (Windows 7) (Current session)' (selected), 'Chrome 60.0.3112.90 (Windows 7)', 'Firefox 47.0 (Windows 7)', and 'IE 11 (Windows 7)'. At the bottom right are two buttons: 'Cancel' and 'Run Test'.

Pick a Browser

The test you have selected includes client-side steps. Please choose a browser to run this test. To manage your test runners, click [here](#).

- ☒ Chrome
60.0.3112.90 (Windows 7) (Current session)
- ☐ Chrome
60.0.3112.90 (Windows 7)
- ☐ Firefox
47.0 (Windows 7)
- ☐ IE
11 (Windows 7)

Cancel Run Test

Impersonate



- Specifies a User for executing subsequent steps in the Test
- Works for both server-side and client-side steps
- Stays in effect until changed with another Impersonate step or until the Test ends
- Impersonated User affects what records you can read and updated
 - Steps might fail due to ACLs
- Don't Impersonate (manually) before running Client Test Runner
- **Best Practice** - always use Impersonate as your first Test Step
- If you cancel a Test before it finishes, the last Impersonation will still be active

Example:

- Impersonate an ITIL User to create a Change Request, then submit for approval
- Impersonate each Approver to update the Approvals and progress through the workflow

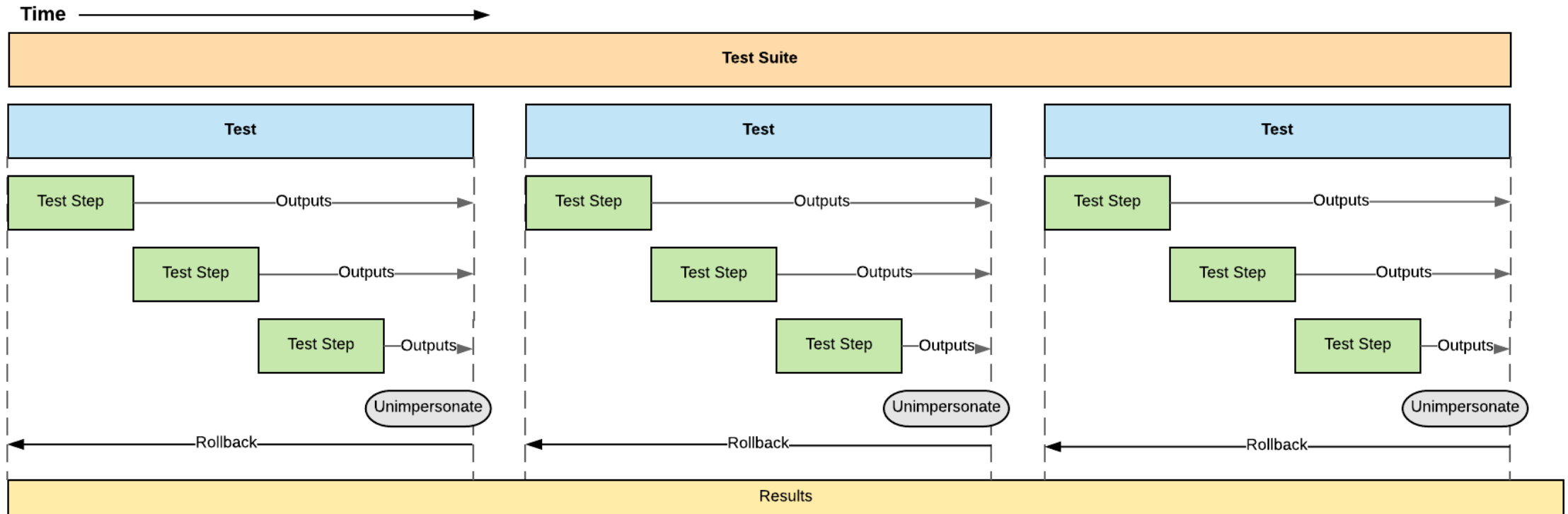
Impersonate - ACLs

- Impersonation affects Access Controls in some Steps
- **Enforce security** – checked by default
- Uncheck to ignore ACLs
- Whether or not to Enforce security depends on your Test Case
 - Leave it checked if the Test User should be able to perform the action
 - Uncheck it if you are performing an action to set up a Test – like creating an Incident to test ESS user updates
- Is the cause of many unexpected Test failures

The screenshot shows the 'Add Test Step: Record Update' dialog. The 'Enforce security' checkbox is checked and highlighted with a red box. Other settings include Execution order 5, Active checked, Application Global, Test Basic UI Test, and Step config Record Update. The Table is set to Incident [Incident], and the Record is Step 4: Submit a Form > Record. Field values are Assignment group and Database.

Test Scope

- Tests are self-contained
 - Outputs are forgotten
 - Unimpersonation occurs
 - Database changes are rolled back
- Cannot pass variables between Tests
- Only Results are persistent



Rollback

- All changes that occur during the Test are rolled back when the test is completed
 - New records are deleted
 - Deleted records are restored
 - Updated records are reverted to previous values
- Tables can be excluded from rollback by adding an **Attribute** to the **Collection** record in the Dictionary:
excludeFromRollback=true
 - Example: Exclude Emails from rollback to check their content and styling
- Appears to be asynchronous
 - Not always completed before the next Test starts
 - Do not rely on rollback being completed before the next Test in the Suite is run (eg, Record Query)

Scheduling (Jakarta+)

- Suite Schedule
 - When to Run
 - List of Test Suites
 - Extends Scheduled Script Execution
- Scheduled Suite Run
 - m2m for Schedules and Test Suites
 - Specify Browser preferences
- Client Test Runners need to be open on a computer with sleep and screensaver disabled (use a VM)
- You can run Suites multiple times with different browsers
- Results can be emailed

The screenshot shows the 'Suite Schedule' configuration page for a 'Weekly Check' suite. The page is divided into two main sections: configuration and a list of scheduled runs.

Configuration Section:

- Name:** Weekly Check
- Description:** This schedule will perform the following actions:
 - Run suite 'Child A' on a system with this configuration:
 - Chrome browser on any version
 - Any operating system on any version
 - Run suite 'Child A' on a system with this configuration:
 - IE browser on any version
 - Any operating system on any version
 - Run suite 'Child B' on a system with this configuration:
 - Chrome browser on any version
 - Any operating system on any version
 - Run suite 'Child B' on a system with this configuration:
 - IE browser on any version
 - Any operating system on any version
- Active:** ☒
- Run:** Weekly
- Day:** Monday
- Time:** Hours 06, 00, 00
- Application:** Global
- Run as tz:** System (US/Pacific-New)
- Conditional:** ☐

Scheduled Suites Section:

Update Delete

Scheduled Suites New Go to Execution order Search

Scheduled Suite Runs

	Test suite	Execution order	Browser name	Browser version starts with	OS name	OS version starts with	Active
<input type="checkbox"/>	Child A	1	Chrome		Any		true
<input type="checkbox"/>	Child A	2	IE		Any		true
<input type="checkbox"/>	Child B	3	Chrome		Any		true
<input type="checkbox"/>	Child B	4	IE		Any		true

Date/Time Inputs

- Populating Date and Date/Time inputs on the client-side can be tricky (surprise!)
- Issues arise when the User you are impersonating has a different Date or Time format than the default system formats
- To avoid these issues, either
 - Ensure your test User is using the system formats before you run your Tests
 - This is easier when you are using a test account
 - Create a custom Step Config that sets the logged in User's Date and Time formats to the system defaults
 - Only needed before populating any dates on the client-side
 - This update will be reverted by the rollback feature
 - Use this when impersonating real Users who might have changed their Date or Time formats
- Any Test Designer building Tests also needs to set their Date and Time formats to system defaults, otherwise it can affect the Test Inputs

Date/Time Inputs

- If you create a custom Step Configuration that outputs a Date or Date/Time value, make sure you create 2 Output Variables; one for use in Server Steps, and one for use in UI Steps

```
1 (function executeStep(inputs, outputs, stepResult) {  
2  
3     var gdt = new GlideDateTime();  
4     outputs.u_now_ui = gdt.getDisplayValue();  
5     outputs.u_now.setValue(gdt.getValue());  
6  
7     stepResult.setOutputMessage('The current time is: ' + gdt.getDisplayValue());  
8     stepResult.setSuccess();  
9  
10 })(inputs, outputs, stepResult);  
11
```

Input Variables		Output Variables (2)		
		Output Variables	New	Go to Label Search
		Model = Generate Current Date/Time		
		Column name	Label ▲	Type
		<u>u_now</u>	Now (Server)	<u>Date/Time</u>
		<u>u_now_ui</u>	Now (UI)	<u>String</u>

Whitelist Client Errors (London)

- Many UI Test failures are the result of errors logged in the console
- There are often very difficult to diagnose and fix
- They are the source of many Known Errors (eg, the Approval form)
- In London, you can whitelist console log errors
- Choose whether they are totally ignored, or if they show a warning

< ☰ Whitelisted Client Error
New record

Report level: Warning - Step & Test will report Success with Warning(s)

Application: Global

Active: ☒

* Error message

Description

Created from test

Submit



What is possible & what is not

What is possible

Server-side Functionality

- Anything and everything – thanks to the “Run Server Side Script” Step Config, and the ability to create custom Step Configs for Server categories.
- Some things may be possible, but not practical – such as validating the HTML or CSS content of an email.
- Some challenges exist when customers want to build Test Cases that run over an extended period of time.
 - For example, SLAs. Creative ways of simulating the passing of time are needed.

Client-side Functionality

- Uses ATF versions of g_form functions:
 - getValue
 - setValue
 - isReadOnly
 - isMandatory
 - isVisible
 - gsftSubmit (UI Actions)
 - save
- Click Modal Button
- If you can't perform an action in a Client Script using one of the above functions, then you probably can't test it with ATF
- You cannot create custom Step Configurations for UI categories

The screenshot displays the 'UI Test Runner' interface for a ServiceNow incident. At the top, a status bar indicates 'Running step 2 of 5 for Incident - MIM AU' and shows connection details: 'Your connection status is [Connected]', 'Currently executing as [Service Desk (AU)]', and 'UI Batches Executed [2]'. A warning message states: 'Performing other activities while tests are being executed can result in those activities being performed by the currently executing user and could be rolled back.' Below this, a progress bar shows 'Taking screenshot and sending it to server'. The main form is titled 'Incident - INC0011189' and includes tabs for 'Execution Frame' and 'Debug Info'. The form fields are organized into two columns. The left column contains: 'Reference Number' (INC0011189), 'Impacted colleague' (ESSTest1 SYS), 'Full address' (Change Manager (NZ)), 'Contact Number' (Daniel Hunter, Danielle.Hunter@iag.com.au), 'Alternate contact No.' (ITIL User (AU), ITILUser_Aus_Dev@iag.com.au; ITIL User (NZ), ITILUser_NZ_Dev@iag.com.au; Joe Employee (AU), Joe_Employee_Aus@iag.com.au; Joe Employee (NZ), Joe_Employee_NZ@iag.com.au; Service Desk (AU), IAG_ServiceDesk_Aus_UAT@iag.com.au), 'Asset number', 'Impact' (1 - High), 'Urgency' (2 - Medium), 'Priority' (2 - High), 'MIM Managed' (MIM AU), 'Short description' (Donec eget nunc at magna volutpat tristique eu in lacus), and 'Description' (Aenean turpis neque, pharetra aliquam metus ut, porta ullamcorper risus. Aliquam placerat metus at orci suscipit, vel pretium augue interdum. Quisque quis euismod dui, blandit vestibulum risus. Cras eget lacinia tortor. Sed at elit nibh. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Vivamus fermentum vestibulum nisl non ultrices. Nam semper nisi massa, nec tristique risus gravida eget. Aliquam erat volutpat.). The right column contains: 'State' (Open), 'Contact type' (Phone), 'Service Item' (ADICS), 'Configuration Item', 'Category' (Incident), 'Subcategory' (Unavailable), 'Assignment group' (SERVICE_DESK_AU), 'Assigned to', 'Vendor Ref No.', and 'FOFC' (checkbox). At the bottom, a search bar shows 'Related Search Results' and a message: 'No matching results found for Donec eget nunc at magna volutpat tristique eu in lacus'.

What is NOT possible

Server-side Limitations

- None that our PS have encountered
- Limited only by your
 - imagination
 - coding ability
 - time



Timed events (such as SLA breaches) can be challenging, but are possible through simulating the passing of time. You can shorten the durations for the Test, or update the `sys_updated_on` field with `gr.setWorkflow(false)` and `gr.autoSysFields(false)`



Some customers encounter "limitations" when they fail to understand the "Automated" in Automated Test Framework.

For example, they try to replicate the functionality of a 'Wait for Condition' Workflow Activity and build Tests that require manual interaction.

Client-side Limitations

Reminder

- There are many limitations on the client-side.
- Remember what ATF is intended for: **functional testing of business logics**
- You should be aiming to test what is **unique to your instance**
- Consider if you really need to test core UI functionality that is the same across every single instance of ServiceNow.
- Think about what actually breaks after an Upgrade. Is it ever the core UI controls like the magnifying glass icon?

Client-side Limitations

Click Actions

- Reference Icons (i)
- Reference field magnifying glass
- Reference popup windows
- URL and List field padlock icons
- Hyperlinks (Labels and URL fields)
- Interceptors
- Context Menus
- Banner features
 - Search, Settings menu, User menu
- Header bar features
 - Back
 - Additional actions (3-bar icon)
 - Attachments
 - Activity stream
 - Personalize form
 - More Options (3-dot menu)
 - Toggle Template bar
 - Tags
 - Next/Previous Record
- Toggle or collapse Sections
- Template Bar
- List Edit

Client-side Limitations

Visual Elements

- Process Flows
- Field Styles
- Field Messages
- Form layouts
- Activity Log contents
- Journal Entries
- Info/Error Messages
- Presence of Related Lists
- Mandatory asterisk
- Date picker
- Slush-buckets
- Email Styling
- Dashboards
- Homepages
- BSM map
- Visual Task Boards
- CAB Workbench
- Gantt Chart
- Performance Analytics
- Reports

Client-side Limitations

Other

- List Views & Related Lists
- Available Options (Select and Reference)
- Login Page
- UI Macros
- UI Pages
- Chat
- Contextual Help
- Guided Tour
- Browser features
 - Alert, Confirm, Prompt
 - Does not render in Client Test Runner
 - Replace with GlideModal for better UX
 - Location (URL)
 - Number incrementors
- Server-side Aborts count as successful client-side form submission

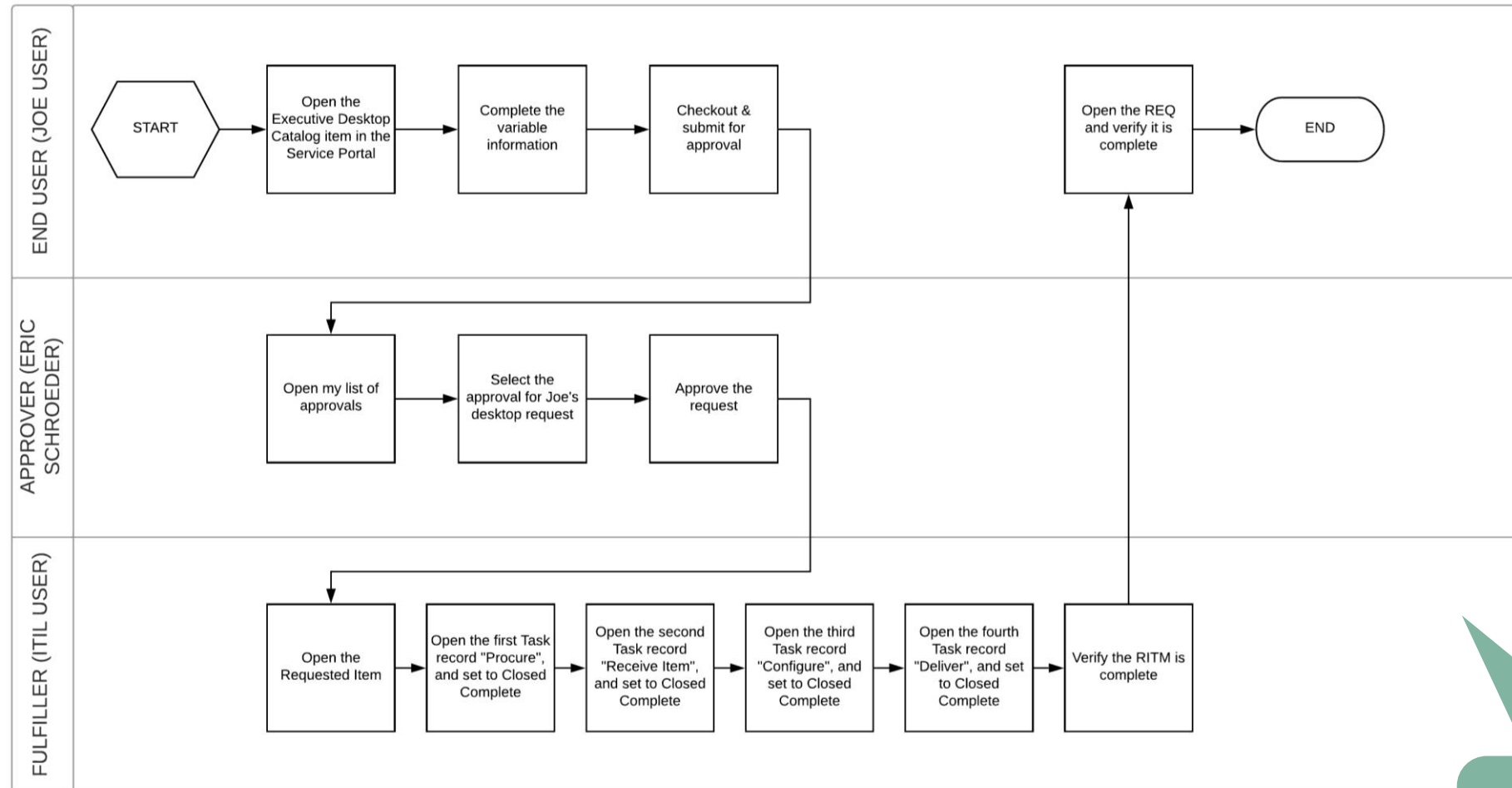
Example

Create a Test for a simple Catalogue Item

Example: Catalogue Item

1. Impersonate: Self-service User
2. Open a Catalog Item
3. Variable State Validation: Check which fields are visible, mandatory or read-only
4. Set Variable Value: Populate the form
5. Validate Variable Values: Check field values are as expected
6. Order Catalogue Item: Submit the form
7. Record Query: Look up the RITM
8. Record Validation: Check the contents of the RITM
9. Impersonate: Approver
10. Record Query: Look up the Approval request
11. Open an Existing Record: Open the Approval request
12. Click a UI Action: Approve the Approval request
13. Impersonate: Fulfiller User
14. Record Validation: Check the state of the RITM
15. Record Query: Look up the Task SLA
16. Record Validation: Check that the correct SLA was attached
17. Record Query: Look up the Catalogue Task
18. Record Validation: Check the contents of the Catalogue Task
19. Open an Existing Record: Open the Catalogue Task
20. Set Field Values: Populate the Assigned to User
21. Click a UI Action: Close Task
22. Record Validation: Check that the RITM was closed

Or a process view



You will build this in one of our labs!

Example: Catalogue Item

< Client Test Runner

Performing other activities while tests are being executed can result in those activities being performed by the currently executing user and could be rolled back.

Running step 1 of 3 for (NF18) Service Catalog via SP Lab 7

Your connection status is [Connected]

Currently executing as [Joe Employee]

UI Batches Executed [5]

Opening Catalog Item

Execution Frame

servicenow

Knowledge


Service Catalog

Requests 11

System Status

Cart

Tours


 Joe Employee

Home > Service Catalog > Hardware > Desktops > Executive Desktop

Search

Executive Desktop

Dell Precision 690



Ideal for corporate customers seeking next-generation design, advanced video card options, dual monitor support and a wide selection of form factors.

1

Add to Cart

Price: \$1,875

Delivery Time: 2 Days

Order Now

now

74

© 2018 ServiceNow, Inc. All Rights Reserved. Confidential.



Planning for ATF

Challenges and Decisions

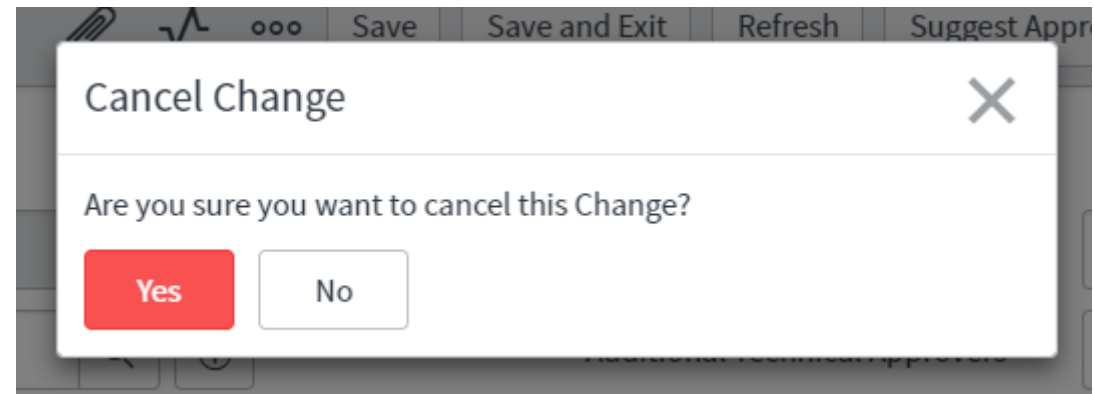
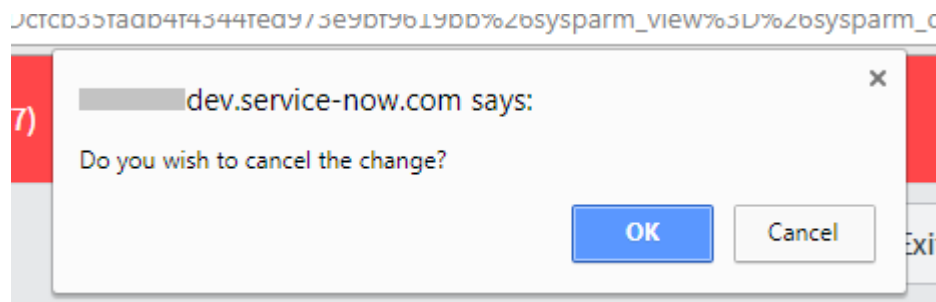
- The Test Designer's challenge is in building Tests that work within the limitations of the current functionality
 - Remember that ATF's goal is for 40-60% coverage
- There are a few questions that need to be considered before you start (detailed next):
 - How much detail should we test? More detailed testing provides more comprehensive testing, but creates large maintenance overheads.
 - Should we change existing functionality to make a Test work?
 - Do we need to test every possible variation in data-driven functionality? For example, a Priority Matrix.
 - Can we start building Tests before Production Go-Live?
 - Should we re-test common functionality across different processes?

Detailed Testing vs ATF Maintenance

- It is possible to create extremely detailed Tests in ATF.
Consider a single field on the Incident form. We can test:
 - Is it visible, read-only and/or mandatory
 - What is its current value
 - Can we set a new value
 - Now repeat this for every other field on the Incident form, then double it for negative testing.
- The issue with highly detailed testing is that when you make a change to the field, you also need to update any and all ATF Tests that look at that field.
 - Do you want to have to find and update ATF Tests each time you change a UI Policy?
- Need to consider how Tests will be maintained
 - Who will build or update the Tests, and when?
 - What is your organisation's Change Management like?
 - How frequently do you update existing functionality?
- You need to find the right balance between detailed testing and ATF maintenance for your organisation

Changing Existing Functionality

- While building new Tests, you may find that you need to reconfigure existing functionality to build a Test
 - An alert() message cannot be tested, but a GlideModal can be tested.
Should you replace the alert() message with a GlideModal?
 - When an update is rejected by a server-side setAbortAction(), it still counts as a successful form submission by the browser.
 - Should you redesign this behaviour to have the update rejected on the client-side?
- Consider if this is worth while (the above examples create a better user experience)



Changing Existing Functionality

- Catalogue Items that work in the Service Portal may not work in the CMS view
- Some specific behaviours are treated differently between the two systems
- Example: Variables that are Mandatory and Hidden
 - The Service Portal hides it
 - The CMS displays a mandatory field
- What do you do in these situations?
 - Revoke the dev's computer licence?
 - Fix the "working" UI Policies?
 - Wait until you upgrade to London?
- List fields can't be populated
 - If you have a mandatory List field, you basically can't test the Item

Data-Driven Features

- Avoid testing data in data-driven features (like the Priority matrix)
 - Use 1 or 2 tests to check the functionality (the value changes when it's supposed to)
 - Any more and you are only testing the data, not the functionality
 - It is very time-consuming to build and run Tests that check every possible variation

Impact

Urgency

Priority

Priority Data Lookups					Go to		Order	Search	1 to 9 of 9	
All										
Data lookup type					Order		Impact	Urgency	Priority	
Search					Search		Search	Search	Search	
<input type="checkbox"/>	i	Priority Data Lookup	100	1 - High			1 - High	1 - Critical		
<input type="checkbox"/>	i	Priority Data Lookup	200	1 - High			2 - Medium	2 - High		
<input type="checkbox"/>	i	Priority Data Lookup	300	1 - High			3 - Low	3 - Moderate		
<input type="checkbox"/>	i	Priority Data Lookup	400	2 - Medium			1 - High	2 - High		
<input type="checkbox"/>	i	Priority Data Lookup	500	2 - Medium			2 - Medium	3 - Moderate		
<input type="checkbox"/>	i	Priority Data Lookup	600	2 - Medium			3 - Low	4 - Low		
<input type="checkbox"/>	i	Priority Data Lookup	700	3 - Low			1 - High	3 - Moderate		
<input type="checkbox"/>	i	Priority Data Lookup	800	3 - Low			2 - Medium	4 - Low		
<input type="checkbox"/>	i	Priority Data Lookup	900	3 - Low			3 - Low	4 - Low		

Migrating ATF Tests

- When building Tests you will need to enter Users and Groups (and possibly many other records) as Inputs to your Test Steps.
 - For example: Impersonating a User, or setting an Assignment group on an Incident.
- Make sure that these records have the same sys_id between all of your instances.
- If the sys_ids don't match, your Tests will break when you migrate to a new instance and all of your Test Steps will need to be updated
- Usually an issue before Production Go-Live if data has been loaded separately into each instance
- One option is to use a defined and consistent set of test Users and Groups for all of your inputs
- Alternatively, you could:
 - Create a custom Step Config that creates a User with the right roles to test (it will be deleted on rollback)
 - Dynamically look up a random existing user that matches your criteria

Common Functionality

- Say you need to build Tests for 20 Catalogue Items that all use a common Variable Set
- Should you create 1 Test for each Item and (re)test the common Variables
 - Each form is tested independently and requires no knowledge of its workings (black box testing)
 - A change to the Variable Set causes all 20 Tests to fail
- Or, should you create 1 Test for the common Variable Set, then 1 Test for the unique functionality on each Item?
 - Requires understanding of Variable Sets (grey box testing)
 - A change to the Variable Set causes 1 Test to fail
- After you submit each Catalogue Item, should you (re)test the RITM State life-cycle?
- Answer depends on your organisation's Test Strategy/Philosophy

Retaining Test Results

- ATF can only be run in non-Production instances
 - Consider the chaos that would ensue if manager approvals were rolled-back during a test run.
- How are you going to maintain your Test Results that are being generated in a non-Prod instance?
 - Import them into Prod
 - Create Data Preserves
 - Do you even need/want to store them long-term?



Best Practice

Best Practice

- Make a plan
 - How detailed will your Tests be?
 - How will your Tests be maintained?
 - How will you decide what functionality to change to make a Test work?
 - How will you track the features that you can't build a Test for in ATF?
 - How will you test common functionality across multiple processes?
- Always add the **Impersonate** Test Step first
 - Without Impersonate, all Tests are executed as the logged in User
 - You do not need to manually impersonate before running a Test
- Avoid testing data in data-driven features. Just test the functionality.
- Test execution time
 - Avoid repeating Test Steps in the UI - Server is always faster
 - Test features in the UI once, then repeat on the Server in other Tests if needed

Best Practice

- Validate existing Tests before an upgrade
 - If new development has been completed, and the ATF Tests have not been updated, then your Tests may not work and will be useless when you need them
- Use a consistent set of Test Users and Groups to avoid migration issues
- Avoid creating Tests with too many Test Steps
 - When a Test fails, all following Test Steps are skipped. If your Test has multiple errors, it will fail at the first error and you won't find the second error until the first error is rectified and you rerun the Test.
 - More granular (shorter) Tests will allow you to identify more issues in a shorter amount of time.
- Create custom Step Configurations where possible
 - Create new types of Tests
 - Simplify and speed up future Test development
 - Reduce and reuse server-side code
 - Only use **Run Server Side Script** for something unique and one-off

Best Practice

- Use **Record Validation** after **Record Update**
 - Record Update appears to always pass, even if the record was not updated (for example, the update was rejected by a Data Policy)
 - This can lead to failures in subsequent Test Steps, and mislead you into troubleshooting the wrong Test Step
- Make sure the Test Designer and impersonated User are using the system default date and time formats
- Use a Record Query to create a Variable when you need to populate a value into multiple Steps
 - Allows you to update a value in one place instead of many
- When populating fields on the client-side, make sure that field updates which trigger AJAX calls are in separate Steps
 - Creates issues when running in a Suite
- Set Screenshots mode to **Enable for failed steps**
 - Speeds up client-side Tests

Best Practice – Dynamic Selection

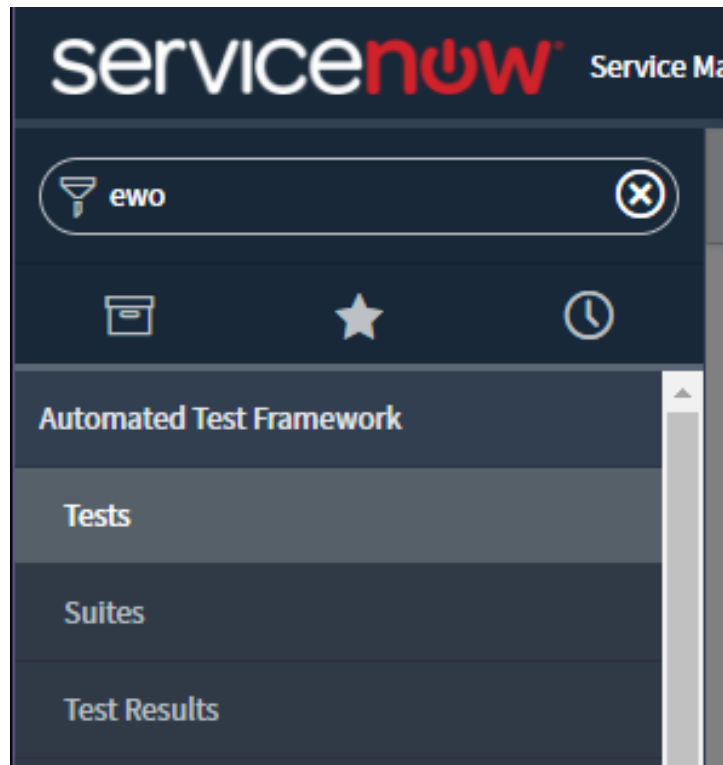
- From PS experience, the most effective way to select a Record to use as a Test input is to use dynamic selection at run-time.
 - For example, if you want to create an Incident, randomly select any active User with the itil role and impersonate them.
- Avoids issues with Test records being changed or deleted.
- Avoids issues with migrating ATF development (and breaking references)
- Adds an element of randomisation to your Tests which tend to result in finding more issues.
 - For example, an Approval workflow that fails when the User's manager is not define



Development in ATF

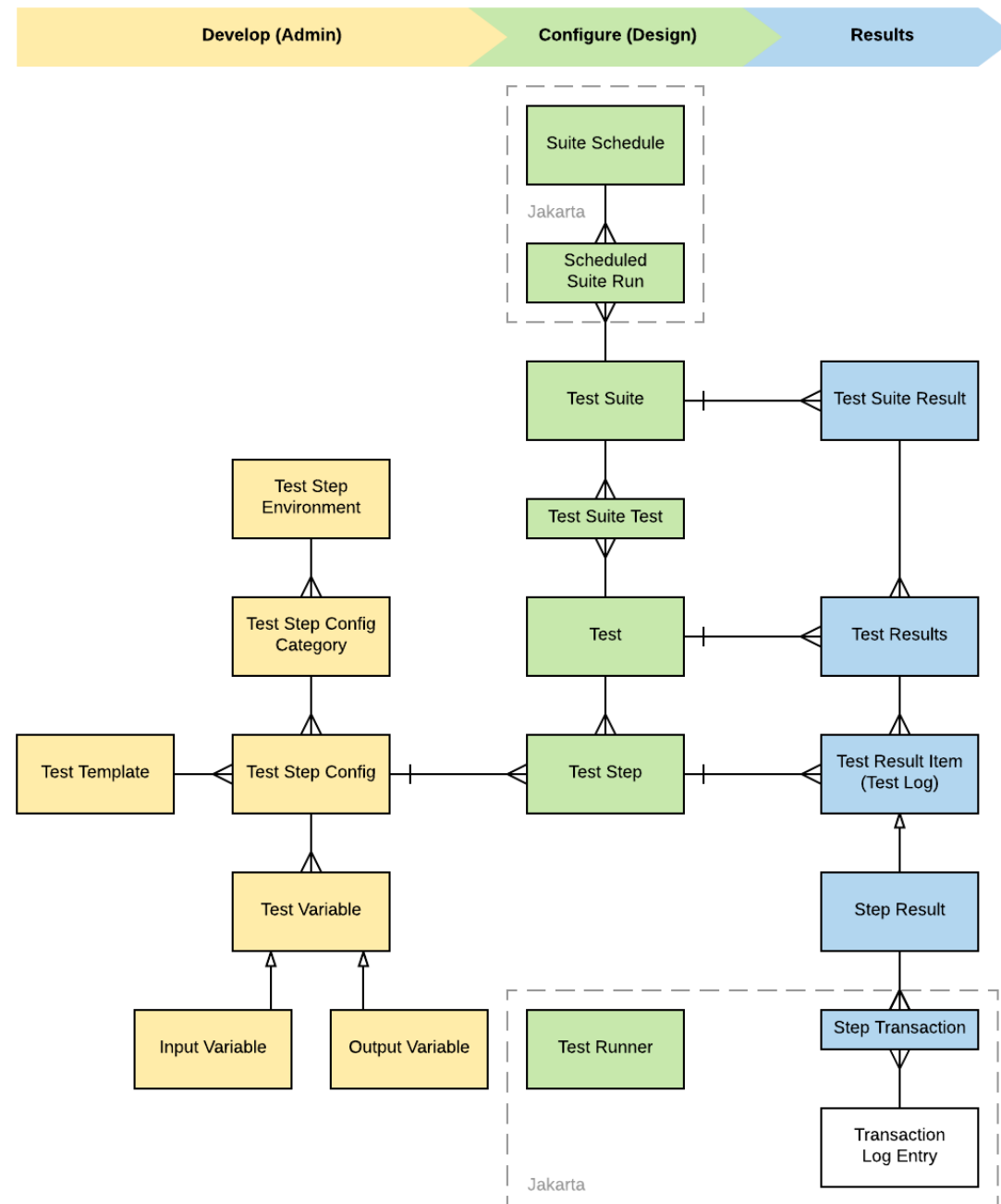
Navigation Tip

- Type **ewo** into the Navigation Filter to bring ATF to the top
 - Smallest number of unique characters



Tables

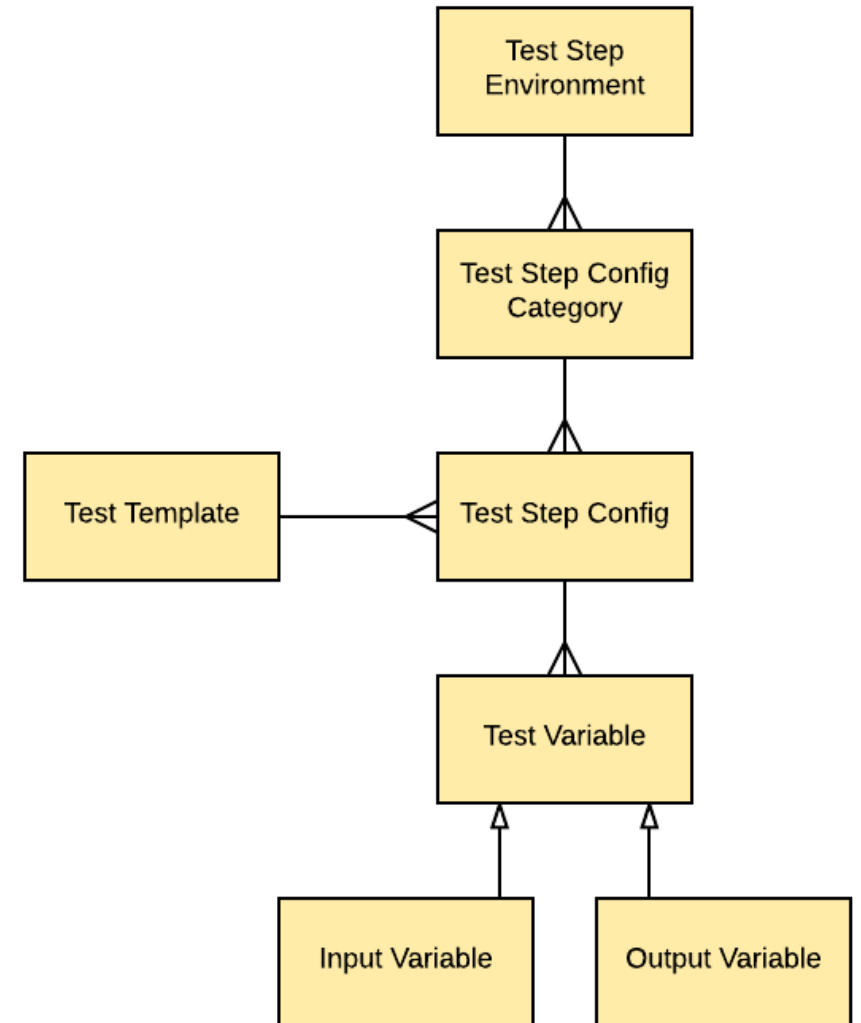
Overview



Test Admin

Developer

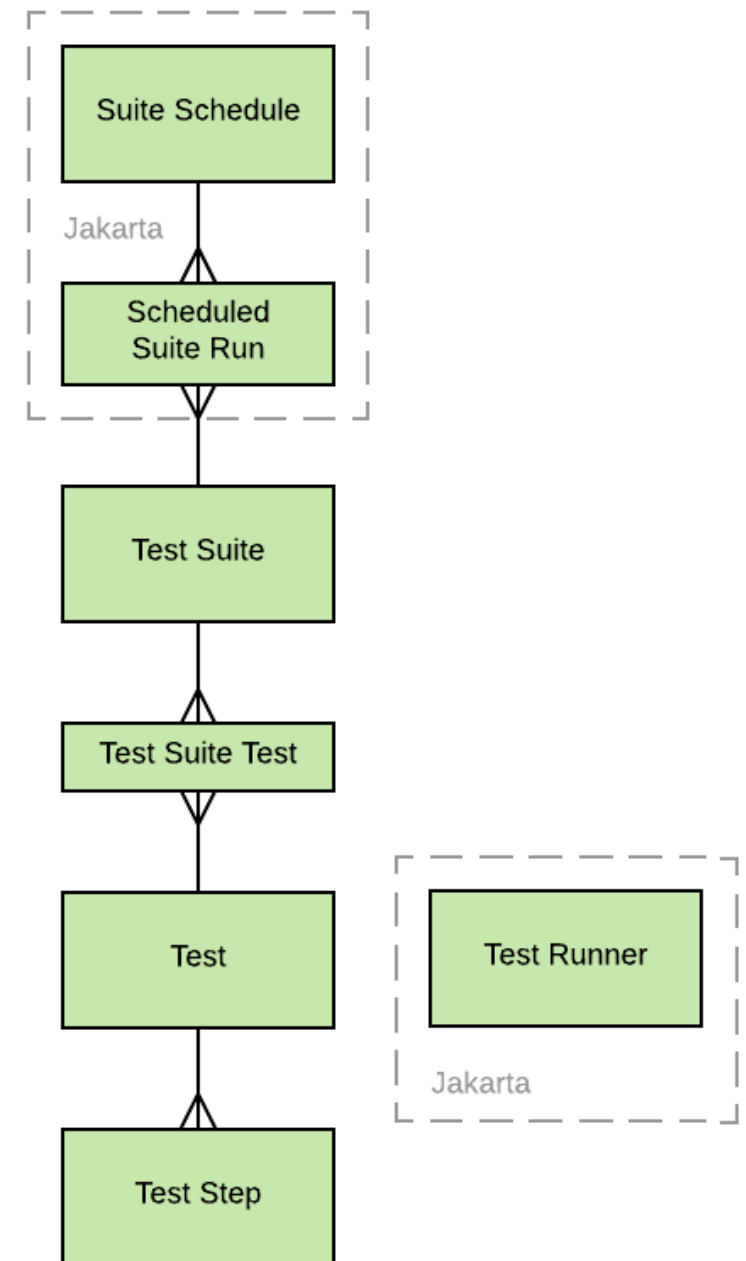
- Test Step Environment: Server, UI
- Test Step Config Category: Form, Server, REST, SC
- Test Step Config
 - Definition of a Test Step
 - Contains code for executing a Test Step
- Test Variable: Parent table for Inputs and Outputs
- Input Variables
 - Defines dynamic inputs for Test Step Configs
- Output Variables
 - Defines dynamic output for Test Step Configs
 - Used as Inputs for future Test Steps
- Test Template
 - A list of Test Step Configs
 - Only used to save a few clicks when adding Test Steps to a Test



Test Designer

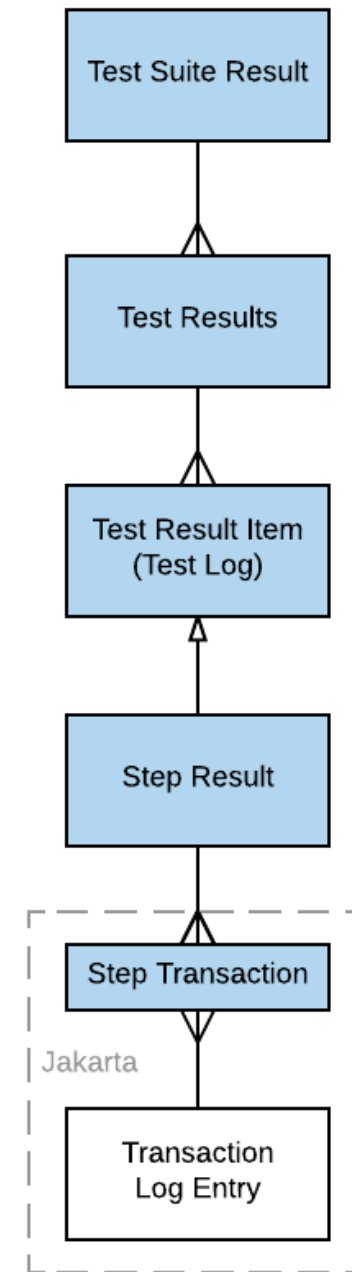
Test Analyst (non-developer)

- Test Suite
 - A set of Tests
 - Used to run many Tests with one click
- Test
 - Represents a Test Case
 - A set of Test Steps
 - Used to test end-to-end functionality of a process
- Test Step
 - A specific action to perform during a Test
 - Action is defined by the Test Step Config
- Suite Schedule
 - Extends Scheduled Script Execution
 - Scheduled job for 1 or more Test Suites
- Test Runner
 - An instance of a Client Test Runner



Results

- Test Suite Result
 - Result from testing an entire Test Suite
 - Successful only if all child Tests are successful
- Test Result
 - Result from testing a Test (Case)
 - Successful only if all child Tests Steps are successful
- Test Result Item / Test Log
 - Log Entries
- Step Results
 - Result from performing a single Test Step (action)
 - Successful if the Test Step was completed
 - Output contain specific details
- Step Transaction
 - m2m link to Transaction Log



Custom Step Configurations

- ATF admins can create new Step Configs
- Only possible for Server Categories (not UI)
- Use this instead of repeating code in multiple “Run Server Side Script” Test Steps
- Very useful in performing common functions, simplifying Test Steps, or creating Tests that otherwise aren’t possible
 - Example: Check for Email
 - Create an Input to select the Notification you want to check was sent
 - Use `gs.sleep()` and check that the Email has been created
- Can be used to generate dynamic values to be used as Inputs for other Test Steps
 - Example: Start and End dates for a Change window
 - Doesn’t actually test anything (always passes)
 - Input a lead time. Output 2 dates for Start and End.

Custom Step Configurations

Examples

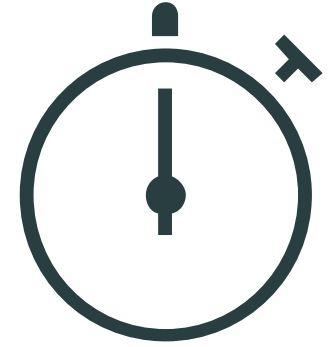
- Email Notification
 - Input a Record ID and an Email Notification. Check that an Email was created.
- Generate Change Window
 - Input a Lead time and Duration. Output Start and End dates.
- Get Approval
 - Input a Record ID. Output an Approval (in Requested State), and the Approver.
 - Use this to dynamically impersonate an Approver and update an Approval record.
- Approve all Approvals
 - Approval all existing Approvals on a given record. Use this to quickly push a Change Request through its workflow
- Get Random User
 - Define some inputs (like Company, Group, Role) and use these to create dynamic queries to create a list of Users that meet the criteria. Then select one at random. Great for picking a valid User for a scenario if you don't want to use test accounts.
- Get Current Date/Time
- Wait
 - Input a number of seconds to wait. Use this for asynchronous updates or for debugging.

gs.sleep()

- Server-side version of JavaScript setTimeout()
- Argument is milliseconds

```
gs.sleep(3000); // wait for 3 seconds
```

- Required when waiting for asynchronous updates
 - Event processing
 - Email Notifications
 - Workflow updates
- Needed to create asynchronous Test Steps, like “Verify Email Notification”.
- Useful for debugging
 - Add a 30 second wait between Steps to manually inspect the state of records before rollback

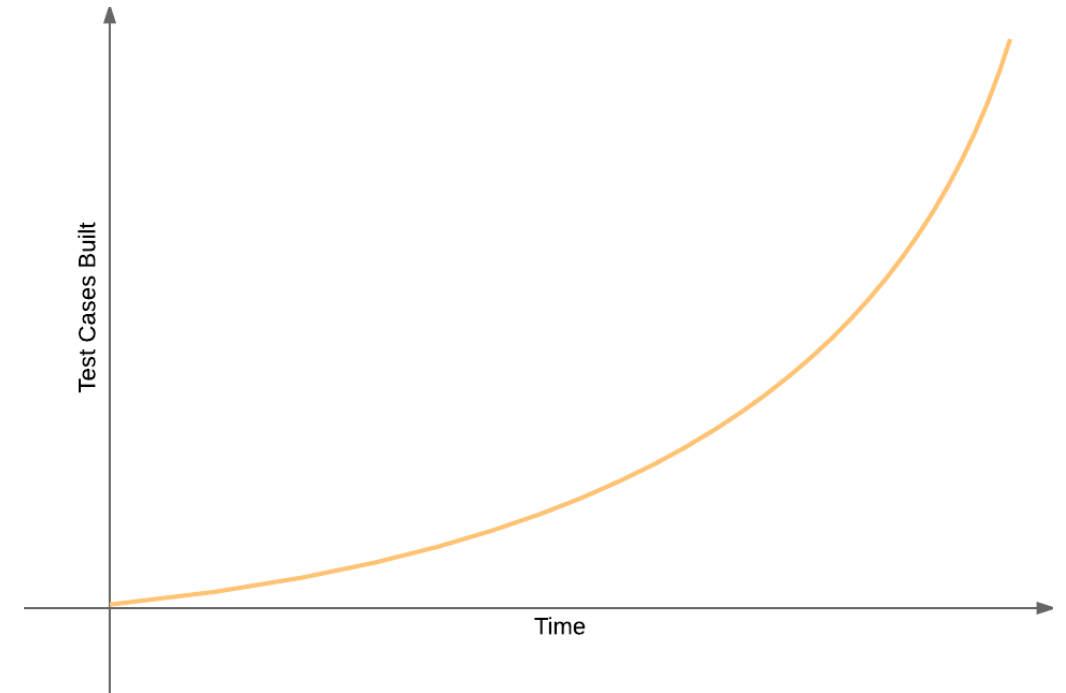


Executing a Test individually vs in a Suite

- Sometimes a Test will work when you execute it on its own, but will fail when executed as part of a Suite
- This is likely due to increased CPU utilisation, network traffic, and/or server workload
- Asynchronous updates take longer (Workflows, Events, Email Notifications)
 - Use asynchronous query methods, or a Wait
- Rendering UI components takes longer (GlideModal)
 - Create a delay in the UI with addition Field State Validations
 - Enable screenshots for all Steps
- GlideAJAX responses take longer
 - Separate triggering actions into different Steps
- Always Unit Test your ATF Tests in a Suite with other Tests
- Solving these problems requires excellent understanding of the Now Platform

Estimating Development Times

- Developing a single Test (Case) can take between a few minutes and a few days
- Writing new Step Configurations is what takes the most time
 - Once they have been written, you can reuse them
 - Consider how **unique** each Test Case/Step is and whether or not you will need to build new Step Configs (eg, Change workflows)
 - Can you “leverage” existing Step Configs off others
- Development starts off slow, and then speeds up as you build a bigger library of Step Configurations, and a bigger library of Tests that can be copied



Estimating Development Times

- Number of Test Steps does not have a significant impact
- Number of Test Cases is not usually a good indicator
 - Multiple Test Cases that test different inputs for the same process can be built very quickly
 - Tests can be copied
 - If every Test Case tests a completely different process or feature, then this will take a long time
 - Some Test Cases are entirely untestable in ATF (eg, using a Slush-bucket)
- Level of detail adds to time and complexity
 - Detailed client-side actions are mostly untestable in ATF (eg, testing the Magnifying glass icon)
- Client-side tests take longer to execute (and therefore Unit Test)
- The quality of existing development may have an impact if functionality needs to be redesigned to allow an ATF Test to be built
- Similar Tests can be copied with a single click

Estimating Development Times

- The best indicator of development time is the **uniqueness** of each Test
 - The more unique Tests you have, the longer they will take to develop
 - Every unique test must be carefully designed and built, one step at a time
 - Tests can be copied, but there is only value if they are similar
 - How many unique Test Cases do you have?
 - Are the steps different?
 - Are the expected outputs different?
 - Are the tables and forms different?
- Do you have a few Test Cases covering a lot of different functionality (high uniqueness), or a lot of Test Cases covering a small amount of functionality (low uniqueness)?

Date/Time Inputs

- Populating Date and Date/Time inputs on the client-side can be tricky (surprise!)
- Issues arise when the User you are impersonating has a different Date or Time format than the default system formats
- To avoid these issues, either
 - Ensure your test User is using the system formats before you run your Tests
 - This is easier when you are using a test account
 - Create a custom Step Config that sets the logged in User's Date and Time formats to the system defaults
 - Only needed before populating any dates on the client-side
 - This update will be reverted by the rollback feature
 - Use this when impersonating real Users who might have changed their Date or Time formats
- Any Test Designer building Tests also needs to set their Date and Time formats to system defaults, otherwise it can affect the Test Inputs

Date/Time Outputs

- If you create a custom Step Configuration that outputs a Date or Date/Time value, make sure you create 2 Output Variables; one for use in Server Steps, and one for use in UI Steps

```
1 (function executeStep(inputs, outputs, stepResult) {  
2  
3     var gdt = new GlideDateTime();  
4     outputs.u_now_ui = gdt.getDisplayValue();  
5     outputs.u_now.setValue(gdt.getValue());  
6  
7     stepResult.setOutputMessage('The current time is: ' + gdt.getDisplayValue());  
8     stepResult.setSuccess();  
9  
10 })(inputs, outputs, stepResult);  
11
```

Input Variables		Output Variables (2)		
		Output Variables	New	Go to Label Search
Model = Generate Current Date/Time				
		Column name	Label ▲	Type
		<u>u_now</u>	Now (Server)	<u>Date/Time</u>
		<u>u_now_ui</u>	Now (UI)	<u>String</u>

Hands on Labs



Thank you