**Workflow Testing Tool**

By Adam Castle ([adam.castle@k2.com](mailto:adam.castle@k2.com)) Modified and extended by Lee Adams ([lee@k2.com](mailto:lee@k2.com))

Purpose: The purpose of this testing tool it to have the ability to test you workflow and its different permutations in one simple click. Features of the tool allow you to

* Action activities in current process
* Configure multiple process tests per test file
* Action activities in IPC
* Set Process data fields at each activity
* Pause for X amount of seconds before trying to action an activity
* Retry X amount of times to action an activity before giving up.
* Call custom code before and after you action an activity. Custom code can be a method that you have to instantiate or a static method
* Pass our Process and Activity context into our custom methods
* Save test results
* Monitor your tests using the viewflow
* Check blackpearl’s error log show new errors
* Check the status of any activity (e.g. Active, Expired, Completed, not taken) i.e that paths are taken/not taken.

Table of Contents

[How it works 1](#_Toc284190753)

[References Assemblies 2](#_Toc284190754)

[References Projects 2](#_Toc284190755)

[Possible uses 2](#_Toc284190756)

[XML testing file explained 2](#_Toc284190757)

[1st example of an XML Test file – Starting and auctioning a basic process 3](#_Toc284190758)

[2nd example of an XML Test – Actioning an IPC activity 4](#_Toc284190759)

[3rd example of an XML Test – With a PreMethod call. 4](#_Toc284190760)

[Important K2 Configurations 7](#_Toc284190761)

[Initial setup for the testing tool 7](#_Toc284190762)

[Using testing tool 8](#_Toc284190763)

# How it works

The tests are all configured within an XML file. The root element of this file is <Processes> and this allows for multiple process tests to occur. Each Process test is enclosed in the <Process> element.

The testing engine isn’t built within the workflow testing UI, instead a separate assembly was create that would do the testing for us. We load the XML file and subscribe to the results event of the test assembly. That’s why I have create a UI project for testing as well as a unit testing project

A few example projects have been created that will be explained within the documentation.

## References Assemblies

Project : K2 Code Helper

Assembly Name : K2Field.Helpers.Core

Project URL : [TBD](http://www.k2underground.com/groups/k2_code_helper/default.aspx)

## References Projects

Referenced project are included with this project except for the assembly SourceCode.Field.Core.Helper.

Project : SourceCode.Field.Testing.SampleData

* Contains sample data for the test workflows

Project : SourceCode.Field.Testing.SampleWorkflow

* Contains sample workflows used by the testing tool

Project : SourceCode.Field.Workflow.Testing

* The engine of the workflow testing tool

Project : SourceCode.Field.Workflow.TestingUI

* UI for the testing tool

Project : SourceCode.Field.Workflow.UnitTesting

* A sample project using the testing engine within Visual Studio unit testing

# Possible uses

Don’t just think of this testing tool as a way to validate changes of a workflow but it can also be used as a way to test all smartobjects that your workflows may use.

# XML testing file explained

There are 2 xml formats Adam’s original format and a newer format which allows for more functionality. If you are upgrading from an old version of this tester then your existing xml files should still work.

|  |  |  |
| --- | --- | --- |
| **XML Node (old)** | **(new)** | **Description** |
| <Processes> | <Processes rootDir="\..\.."> | The root element of the XML test file. The new format optionally point to the root directory so that files can be linked even if the tests start in different directories |
| n/a | <Processes fileName="Approval.xml"> | A link to processes in another file |
| <Process> | <Process type="cleanup" uniqueID="remove2011subsites" description="Removes all 2011 sub sites" processName="Underwriting\TestProcesses\DeleteYearOfAccountSites" TestEnabled="True" | Opening tag of the process test. You can have multiple process test per XML file.  Type: cleanup, setup and assert (default if blank) – This allows you to choose which type of tests to run. E.g. You may need your tests to have some setup and tests may not run perfectly unless they are cleaned up. N.B. All setup and cleanup tests must be coded or configured by you. This will not clear up process instances…yet!  uniqueID is used to stop multiple test files from running the same test over and over again. E.g. some tests may require a sharepoint site collection to be created first. You may have a setup process that creates this site collection, but when running multiple tests you do not want this setup process to be run multiple times.  TestEnabled is used to turn some tests off. E.g. you run a bunch of tests and the 5th test failed. You fixed the test and want to continue from where you left off, so set TestEnabled="False" on the first four tests. |
| <Activities> | <Activity name="Quote NTU" exceptionExpected="True"  > | Opening tag of the activity event  exceptionExpected – Allows the k2 server to put the process into an error state without failing the test. E.g. The test is supposed to raise and error. |
| n/a | <Activities fileName= "ApprovalActivities.xml" /> | A link to activities in another file |
| n/a | <Activity type="StartProcess" > | Useful when you need to leave an activity active, then start a new process before finishing the activity. |
| <ProcessName>SourceCode.Field.Testing.SampleWorkflow\Travel</ProcessName> | processName= "Travel" | Used to specify the process name to start, or to action an IPC event |
| <Activity>**Start**</Activity> | N/A | If ‘start’ then a new process will be started. Not required in new format |
| <Activity>**Management Approval**</Activity> | name="Management Approval" |  |
| <DataFields> | <DataFields> | Opening tag of the datafields node |
| </DataFields> | </DataFields> | Closing tags of the datafields node. |
| <Action>**Accept**</Action> | <Action>**Accept**</Action> | The action to execute at the specified activity |
| n/a | <Action**>[Completed]**</Action> | Anything in [] is matched up against the Status column of the out of the box Activity\_Instance SmartObject. [NotTaken] is a special token which passes if the SmartObject is empty. |
| <RetryInSeconds>**10**</RetryInSeconds> | <RetryInSeconds>**10**</RetryInSeconds> | The amount of time to pause between trying to actions activities. |
| <RetryCount>1**5**</RetryCount> | <RetryCount>1**5**</RetryCount> | The number of times to keep trying before failing the test |
| <PreMethodCall Assembly="C:\SourceCode.Field\SourceCode.Field.Testing.SampleData\bin\Debug\SourceCode.Field.Testing.SampleData.dll" Class="Travel" Method="CreateTravelObject"> | <PreMethodCall Assembly="C:\SourceCode.Field\SourceCode.Field.Testing.SampleData\bin\Debug\SourceCode.Field.Testing.SampleData.dll" Class="Travel" Method="CreateTravelObject"> | Start tag of the premethod call that requires a method to be called. |
| <Parameter>[Process]</Parameter> | <Parameter>[Process]</Parameter> | Parameters used by the PreMethod call. |
| </PreMethodCall> | </PreMethodCall> | End |
| <PostMethodCall /> | <PostMethodCall /> | Node of the PostMethod call. This is empty otherwise can be setup same as the PreMethodCall |
| </Activities> | N/a | End tag of the Activities node |
| </Process> |  | End tag of the Process node |
| </Processes> |  | End tag of the Processes node (end of XML test file) |

DataField Attributes

|  |  |  |
| --- | --- | --- |
| Name="**Amount**">**67**</DataField> | name="**Amount**" >**67**</DataField> | Used when setting a process data field. The name attribute contains the name of the process datafield and the value is in between the elements. |
| n/a | type="Activity|Process" | Is it an Activity or Process level datafield? |
| n/a | action="check:??" | Checks the value of the datafield.  ne=checks not equal to  eq=checks equal to |
| n/a | action="store:  action="setToVariable" | Stores the value of the datafield in a global variable that can be used (by setToVariable) in a later test (useful in setup workflows) |

Below is an example

## 1st example of an XML Test file – Starting and auctioning a basic process

Below is an old style example of;

* Starting a process call SourceCode.Field.Testing.SampleWorkflow\Travel
* Also start this process setting the datafield Amount with the value of 67
* Find the activity ‘Needs Approval’ and action this activity with ‘Accept’

|  |
| --- |
| <?xml version="1.0" encoding="utf-8" ?>  <Processes>  <Process Description="Travel - Gets approved at start">  <Activities>  <ProcessName>SourceCode.Field.Testing.SampleWorkflow\Travel</ProcessName>  <Activity>Start</Activity>  <DataFields>  <DataField Name="Amount">67</DataField>  </DataFields>  <Action />  <RetryInSeconds>5</RetryInSeconds>  <RetryCount>5</RetryCount>  <PreMethodCall />  <PostMethodCall />  </Activities>  <Activities>  <ProcessName />  <Activity>Needs Approval</Activity>  <DataFields />  <Action>Accept</Action>  <RetryInSeconds>5</RetryInSeconds>  <RetryCount>5</RetryCount>  <PreMethodCall />  <PostMethodCall />  </Activities>  </Process> |

## 2nd example of an XML Test – Actioning an IPC activity

Below is an new style example of;

* Start a process called SourceCode.Field.Testing.SampleWorkflow\Travel
* With datafields;
  + **Amount** with a value of **67**
  + **AdvApprovalRequired** with a value of **True**
* Find the activity ‘**Needs Approval’** and action with ‘**Accept**’
* Find the activity ‘Upper Management Approval’ in the IPC ‘SourceCode.Field.Testing.SampleWorkflow\AdvApproval’ and action with ‘Accept’

|  |
| --- |
| <Processes>  <Process Description="Travel - Adv Approval Required" processName="K2Field.Utilities.Testing.SampleWorkflow\Travel">  <DataFields>  <DataField Name="Amount">67</DataField>  <DataField Name="AdvApprovalRequired">True</DataField>  </DataFields>  <PreMethodCall />  <PostMethodCall />  <Activity name="Needs Approval">  <ProcessName>K2Field.Utilities.Testing.SampleWorkflow\Travel</ProcessName>  <DataFields />  <Action>Accept</Action>  <RetryInSeconds>5</RetryInSeconds>  <RetryCount>5</RetryCount>  <PreMethodCall />  <PostMethodCall />  </Activity>  <Activity name="Upper Management Approval">  <ProcessName>K2Field.Utilities.Testing.SampleWorkflow\AdvApproval</ProcessName>  <Activity></Activity>  <DataFields />  <Action>Accept</Action>  <RetryInSeconds>5</RetryInSeconds>  <RetryCount>10</RetryCount>  <PreMethodCall />  <PostMethodCall />  </Activity>  <Activity name="Book Flight">  <ProcessName>K2Field.Utilities.Testing.SampleWorkflow\Travel</ProcessName>  <DataFields />  <Action>Task Completed</Action>  <RetryInSeconds>5</RetryInSeconds>  <RetryCount>5</RetryCount>  <PreMethodCall />  <PostMethodCall />  </Activity>  <Activity name="Book Hotel">  <ProcessName>K2Field.Utilities.Testing.SampleWorkflow\Travel</ProcessName>  <DataFields />  <Action>Task Completed</Action>  <RetryInSeconds>5</RetryInSeconds>  <RetryCount>5</RetryCount>  <PreMethodCall />  <PostMethodCall />  </Activity>  </Process> |

## 3rd example of an XML Test – With a PreMethod call.

**N.B The below has not changed since the initial version and needs to be updated to match the new namespaces**

Below is an example of;

* Starting a process called SourceCode.Field.Testing.SampleWorkflow\SimpleTravel
* We also have a premethod call where we calls a method with the following details
  + Assembly : **C:\SourceCode.Field\SourceCode.Field.Testing.SampleData\bin\Debug\SourceCode.Field.Testing.SampleData.dll**
  + Class : **Travel**
  + Method : **CreateTravelObject**
  + Parameter 1 : **[Process]** – this inserts the process context
  + Parameter 2 : **65**
* Actions the client activity ‘**DefaultActivity**’ with action ‘**TaskCompleted**’.

*Code for the PreMethod node is located after the following XML.*

|  |
| --- |
| <?xml version="1.0" encoding="utf-8" ?>  <Processes>  <Process Description="Simple Travel - Calls a premethod function then starts the workflow with data. ">  <Activities>  <ProcessName>SourceCode.Field.Testing.SampleWorkflow\SimpleTravel</ProcessName>  <Activity>Start</Activity>  <DataFields />  <Action />  <RetryInSeconds>5</RetryInSeconds>  <RetryCount>5</RetryCount>  <PreMethodCall Assembly="C:\SourceCode.Field\SourceCode.Field.Testing.SampleData\bin\Debug\SourceCode.Field.Testing.SampleData.dll" Class="Travel" Method="CreateTravelObject">  <Parameter>[Process]</Parameter>  <Parameter>65</Parameter>  </PreMethodCall>  <PostMethodCall />  </Activities>  <Activities>  <ProcessName>SourceCode.Field.Testing.SampleWorkflow\SimpleTravel</ProcessName>  <Activity>DefaultActivity</Activity>  <DataFields />  <Action>Task Completed</Action>  <RetryInSeconds>5</RetryInSeconds>  <RetryCount>5</RetryCount>  <PreMethodCall />  <PostMethodCall />  </Activities>  </Process>  </Processes> |

Code for :

* Assembly : **SourceCode.Field.Testing.SampleData.dll**
* Class : **Travel**
* Method : **CreateTravelObject**

As can be seen from the code below, using our helper **SourceCode.Field.Core.Helper** we create a smartobject record. When creating a record we get back the ID of the newly created record. When then add a new activity process data field to our process context.

p.Activities[0] is infact the **start** activity for the XML above. Therefore it will start the process with this new data field.

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using SourceCode.Field.Core.Helper;  using SourceCode.Field.Workflow.Testing;  namespace SourceCode.Field.Testing.SampleData  {  public class Travel  {  public void CreateTravelObject(Process p, string num)  {  using (SourceCode.Field.Core.Helper.K2Helper helper = new K2Helper("dlx"))  {  Dictionary<string, object> Data = new Dictionary<string, object>();  Data.Add("PersonName", "Adam Castle");  Data.Add("Amount", num);  string TravelRequestsID = helper.SmartObjectClient().SmartObjectCreate(Data, "TravelRequestsID", "TravelRequests");  //Creating a new data field, so when our process starts it will start with the new travel request ID  p.Activities[0].DataFields.Add("TravelRequestsID", TravelRequestsID);  }  }  }  } |

Possible activity parameters are listed below

<Parameter>[Process]</Parameter> will resolve to the current process context

<Parameter>[Activity]</Parameter> will resolve to the current activity context

<Parameter>67</Parameter> will resolve to a string.

Details of the Process and Activity classes are below.

|  |
| --- |
| Process  public string Folio { get; set; }  public string Description { get; set; }  public string ProcessName { get; set; }  public int ProcessInstanceID { get; set; }  public string ActivityExecutionError { get; set; }  public string ProcessError { get; set; }  public string ProcessStatus { get; set; }  public bool ProcessHasErrors { get; set; }  public DateTime TestStartDate { get; set; }  public DateTime TestEndDate { get; set; }  public List<Activity> Activities { get; set; }  Activity  public string Name { get; set; }  public string ProcessName { get; set; }  public int ProcessInstanceID { get; set; } //Used when actioning an IPC activity  public string ActivityExecutionError { get; set; }  public string TestStatus { get; set; }  public string Action { get; set; }  public MethodCalls PreMethodCall { get; set; }  public MethodCalls PostMethodCall { get; set; }  public int RetryInSeconds { get; set; }  public int MaxRetryCount { get; set; }  public int CountOfRetries { get; set; }  public Dictionary<string, object> DataFields { get; set; }  MethodCalls  public string Assembly { get; set; }  public string Class { get; set; }  public string Method { get; set; }  public List<string> Parameters { get; set; }  public bool NeedToInvoke { get; set; } |

# Important K2 Configurations

This tool is designed to imitate how a workflow would be tested within your organization. Therefore it is important to;

1 - Set the users account that will be running the test to have admin permissions on the workflows to be tested.

2 - Set Impersonate rights on the K2 Server for this users account.

# Initial setup for the testing tool

I have created two sample projects that can be used to further understand how tests can be created.

Please make sure you deploy these solutions before you use the testing tool

Project : SourceCode.Field.Testing.SampleData

* Contains sample data for the test workflows in static methods.

Project : K2Field.Utilities.Testing.SampleWorkflow

* Workflows
  + TravelRequest
    - Travel request sample
  + AdvApproval
    - Advanced approval for the travel request
  + SimpleTravel
    - A simple travel request workflow, the configured test calls a static method in an external assembly to insert data.
* SmartObjects
* TravelRequests

Project : K2Field.Utilities.Testing.TestFilesDB

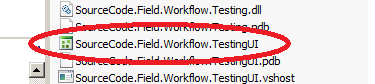
* + Tables to store the xml file structure in

# 

# Using testing tool

The actual testing tool is located with the project ‘SourceCode.Field.Workflow.TestingUI’

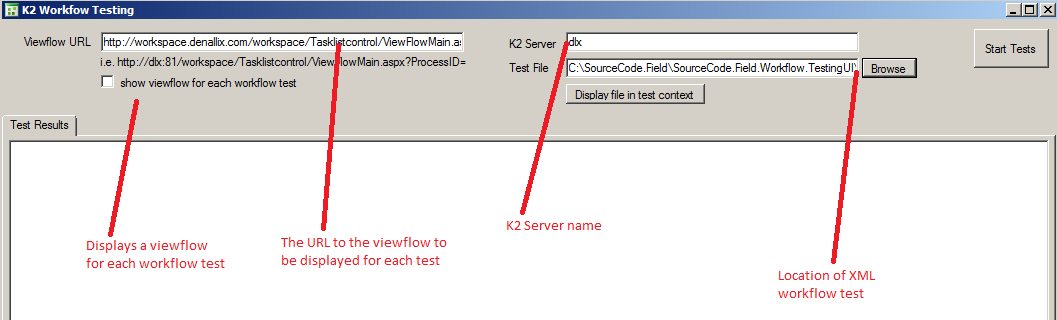
Located this project and the file ‘**SourceCode.Field.Workflow.TestingUI.exe**’ in SourceCode.Field.Workflow.TestingUI\bin\Debug.



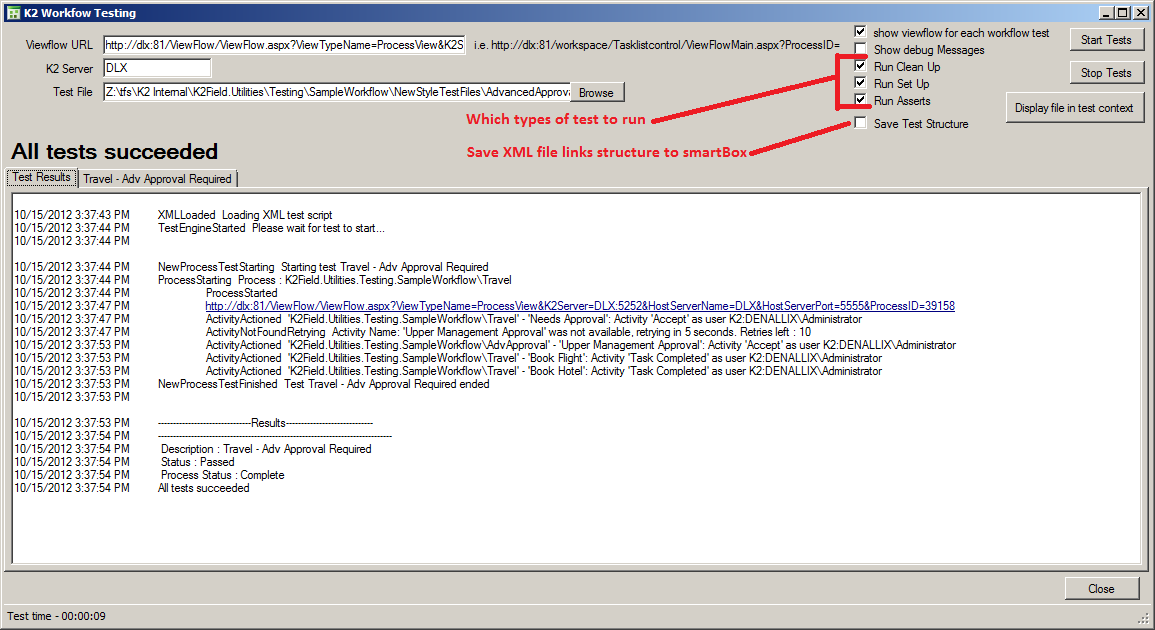
Run this tool as the user that has permissions to impersonate on the K2 server and also has admin rights to the workflows that are to be tested.  
The two configured test are located in **SourceCode.Field.Workflow.TestingUI\Tests\Travel**

* SimpleTravelTest
  + Test shows how to use methods in external assemblies as part of your tests.
* TravelTestdo joe cools
  + This show how to action different activities during your test as well as executing activities in IPC calls.

Old Screenshot



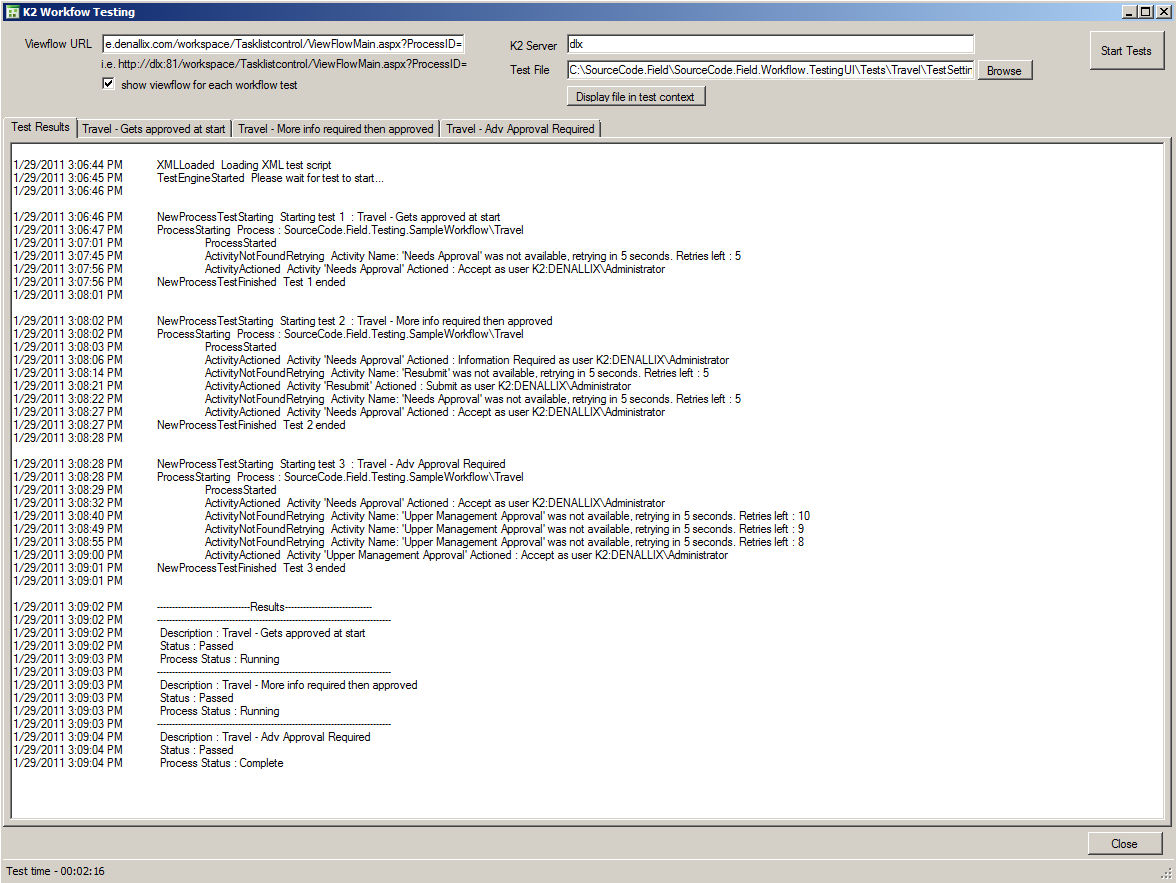
New Screenshot



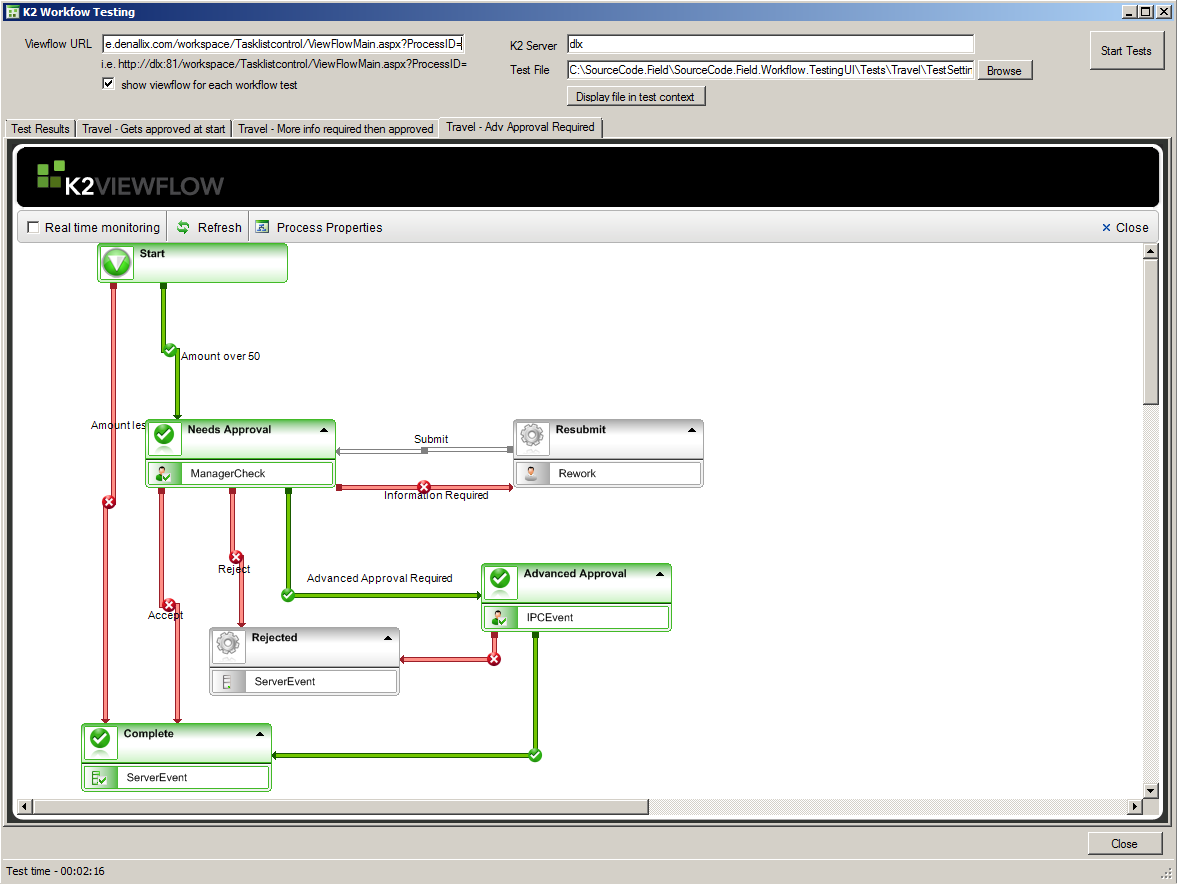
Load a test and click **Start Tests**.

Below shows you the basic output of a test.

* When an activity is actions, or cannot be found and the actioner
* It also displays the test results. These are also stored in XML format that can be for later analysis.



The next image show that a viewflow can be displayed for each test. This allows us to monitor the test in real time.



## Advanced Options

Consider the following scenarios:

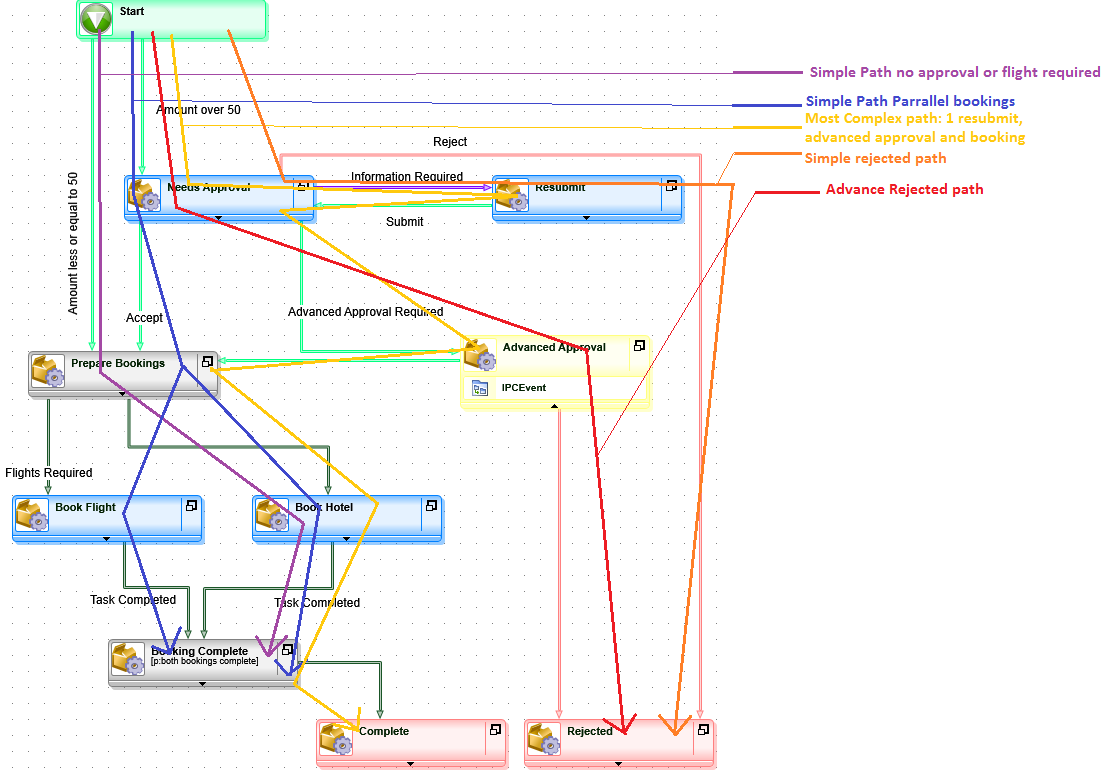
1. An IPC that can also be started independently
2. Running all your tests at once
3. A lot of tests which cover the same actions to test all paths
4. Need to make sure paths (line runes) are not taken (i.e. make sure the process does not go parallel when it should be sequential)

In the example project we have an Advanced Approval process which is normally started by the Travel Process. It could also be started independently and possibly in the future be started by a separate process (e.g. an expense claim process)

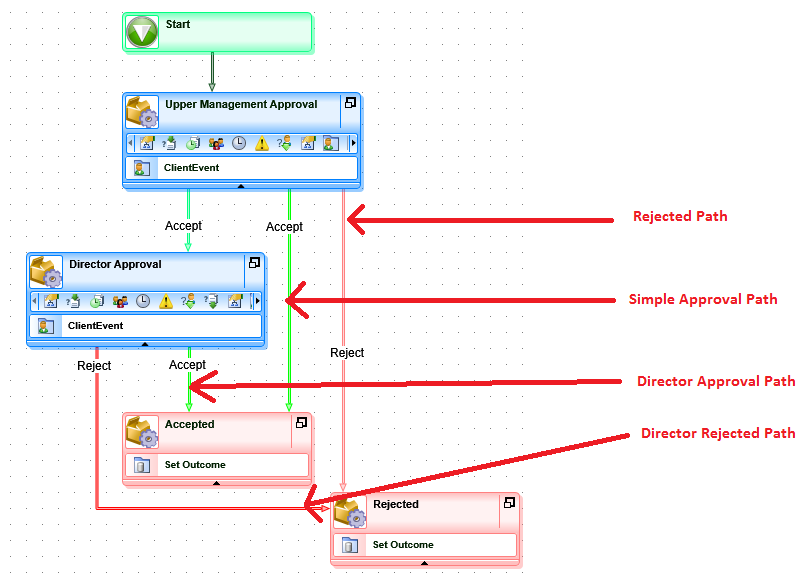
Also in the travel process there are the booking activities which need to be completed to finish at least 3 tests once approval has taken place.

My test plan to test all paths for both processes looks like this:

* Travel – 5 tests



Advanced Approval – 4 tests



I have a total of 9 tests that should test every line rule successfully. In addition to of my Travel tests will also need to test the advanced approval paths.

I structure my test files like this:

* **All.xml** 🡪 Points to the following 2 root test files

|  |
| --- |
| <Processes>  <Processes fileName="AllTravel.xml" description="run test" TestEnabled="True" />  <Processes fileName="AllAA.xml" description="run test" TestEnabled="True" />  </Processes> |

* **AllAA.xml** 🡪 points to all the 4 Advanced Approval tests

|  |
| --- |
| <Processes fileName="Tests\AA\1.NoDirector-Approved.xml" description="run aa1" />  <Processes fileName="Tests\AA\2.NoDirector-Rejected.xml" description="run aa2" />  <Processes fileName="Tests\AA\3.Director-Approved.xml" description="run aa3" />  <Processes fileName="Tests\AA\4.Director-Rejected.xml" description="run aa4" /> |

* **AllTravel.xml** 🡪 points to all the 5 travel tests
* **\Tests\Travel\1.Simple.xml**

|  |
| --- |
| <Processes rootDir="\..\..">  <Process Description="Travel - No Approval" processName="K2Field.Utilities.Testing.SampleWorkflow\Travel">  <DataFields>  <DataField Name="Data.Amount">50</DataField>  <DataField Name="Data.AdvApprovalRequired">False</DataField>  <DataField Name="Data.FlightsRequired">False</DataField>  </DataFields>  <PreMethodCall />  <PostMethodCall />  <Activities fileName="~\Activities\Travel\BookHotelOnly.xml" />  <Activity name="Needs Approval">  <ProcessName>K2Field.Utilities.Testing.SampleWorkflow\Travel</ProcessName>  <DataFields />  <Action>[NotTaken]</Action>  <RetryInSeconds>5</RetryInSeconds>  <RetryCount>5</RetryCount>  <PreMethodCall />  <PostMethodCall />  </Activity>  </Process>  </Processes> |

* + **\Tests\Travel\2.SimpleWithFlight.xml**
  + **\Tests\Travel\3.Complex.xml**
  + **\Tests\Travel\4.SimpleReject.xml**
  + **\Tests\Travel\5.AdvancedReject.xml**
  + **\Tests\AA\1.NoDirector-Approved.xml**
  + **\Tests\AA\2.NoDirector-Rejected.xml**
  + **\Tests\AA\3.Director-Approved.xml**
  + **\Tests\AA\4.Director-Rejected.xml**

In addition I can see that the book Hotel path and the advanced approval routes will be reused twice.

I make these paths their own xml files Structured like this:

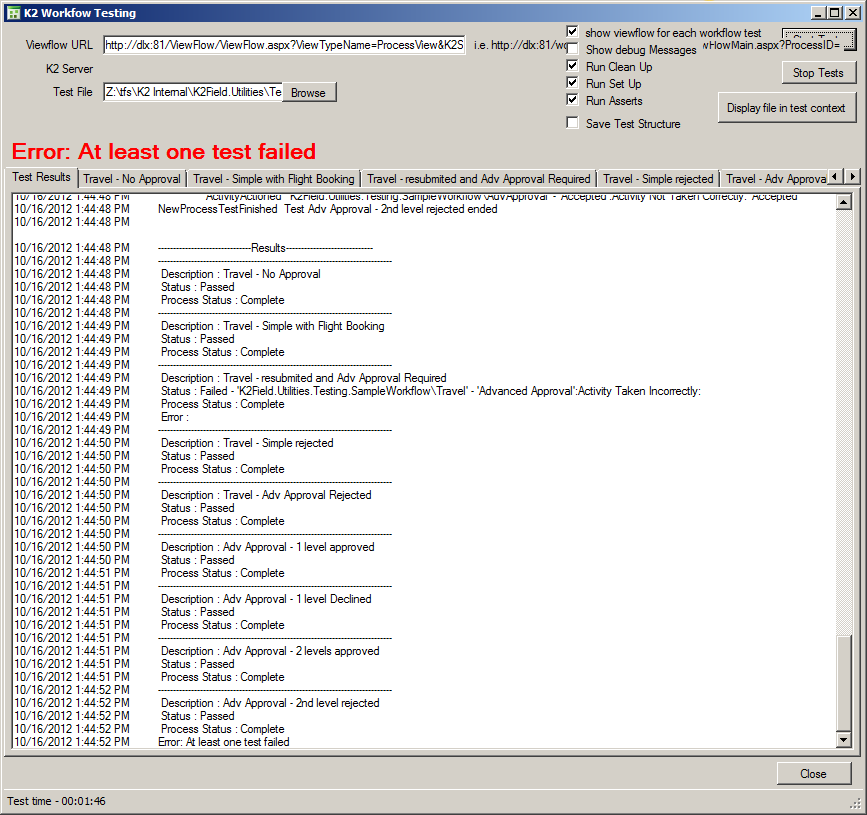
* **\Activities\Travel\BookHotelOnly.xml** (used by Travel tests 1 and 3)

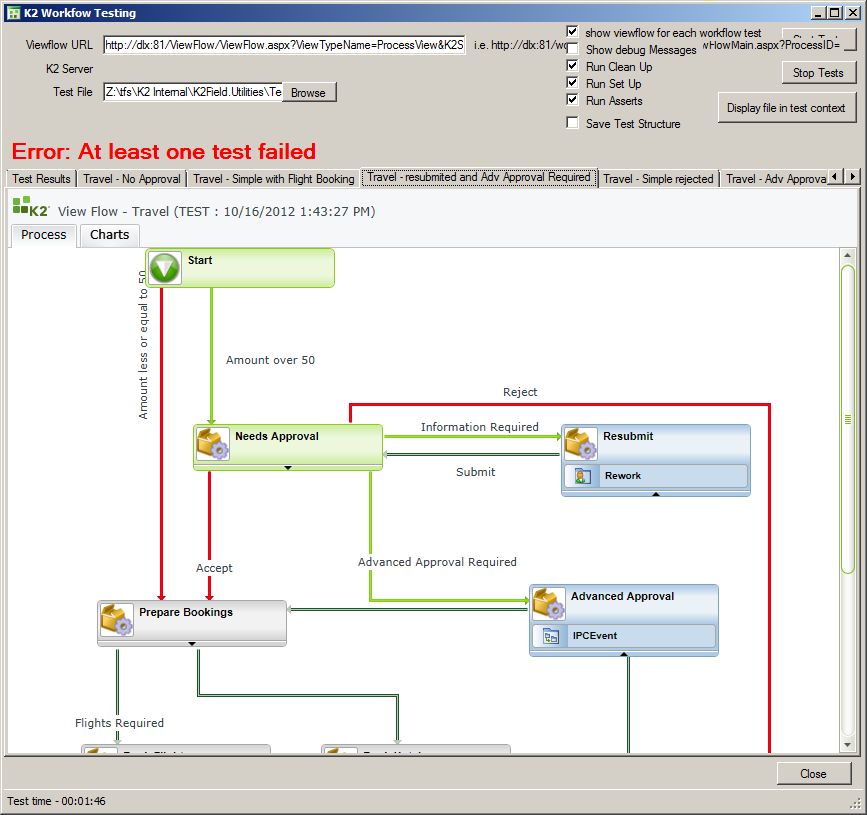
|  |
| --- |
| <Activities>  <Activity name="Book Hotel">  <ProcessName>K2Field.Utilities.Testing.SampleWorkflow\Travel</ProcessName>  <DataFields />  <Action>Task Completed</Action>  <RetryInSeconds>2</RetryInSeconds>  <RetryCount>15</RetryCount>  <PreMethodCall />  <PostMethodCall />  </Activity>  <Activity name="Booking Complete">  <ProcessName>K2Field.Utilities.Testing.SampleWorkflow\Travel</ProcessName>  <DataFields/>  <Action>[Completed]</Action>  <RetryInSeconds>2</RetryInSeconds>  <RetryCount>15</RetryCount>  <PreMethodCall>  </PreMethodCall>  <PostMethodCall></PostMethodCall>  </Activity>  <Activity name="Book Flight">  <ProcessName>K2Field.Utilities.Testing.SampleWorkflow\Travel</ProcessName>  <DataFields />  <Action>[NotTaken]</Action>  <RetryInSeconds>2</RetryInSeconds>  <RetryCount>15</RetryCount>  <PreMethodCall />  <PostMethodCall />  </Activity>  </Activities> |

* **\Activities\AA\NoDirector-Approved.**xml (used by Travel test 3 and AA test 1)
* **\Activities\AA\NoDirector-Rejected.xml** (used by Travel test 5 and AA test 2)

## Some Things to note

One test actually fails. I’m sure this was by design. I’ve left it in.





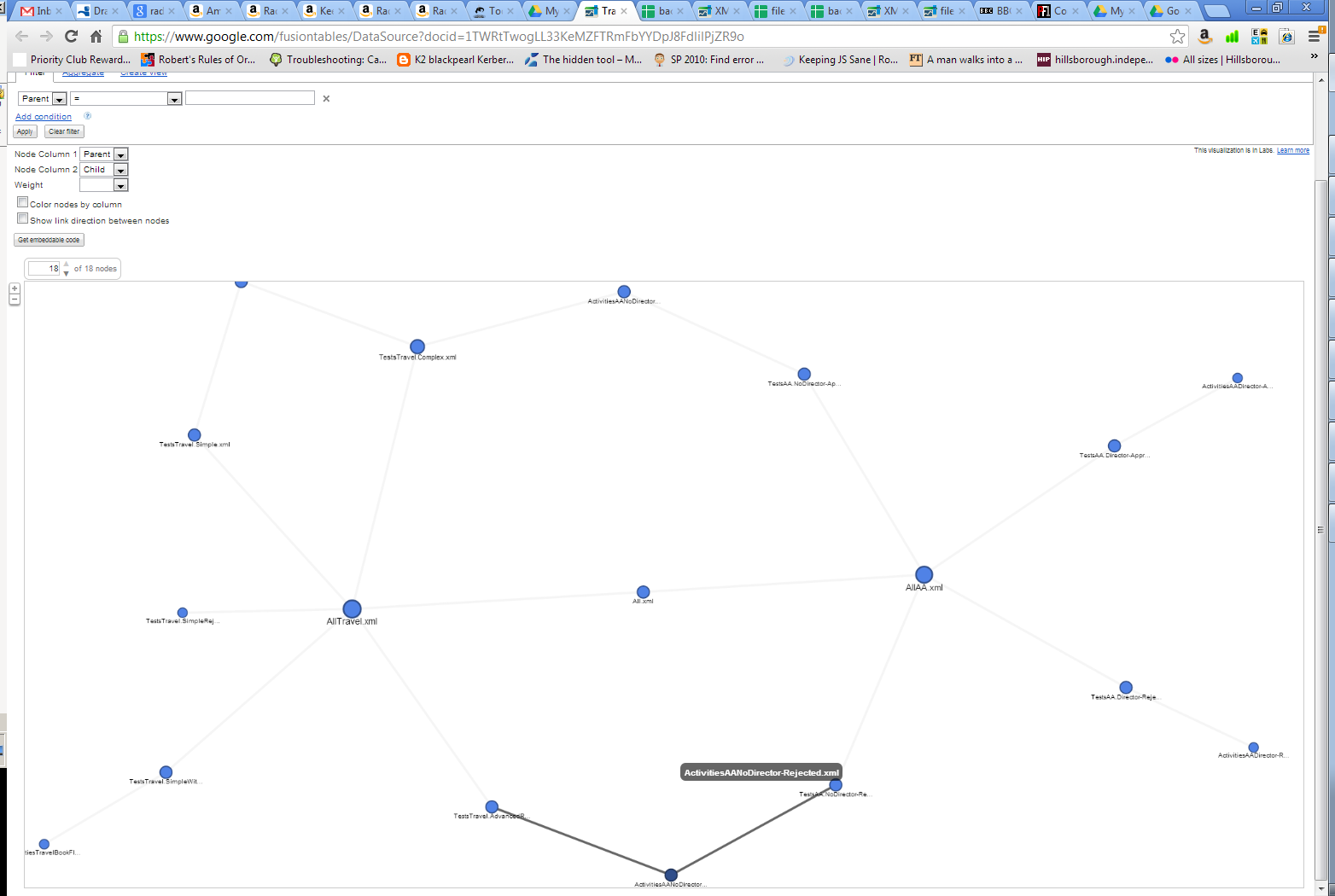
*(You can see above that the test went to parallel paths when it was not supposed to!)*

Tests not in the root directory have a ‘rootDir="\..\.."’ in them to let the tester UI know where to find the linked files. This let you run test files (that links to another file) both from the sub directory and also link to that file from a test file in the root directory.

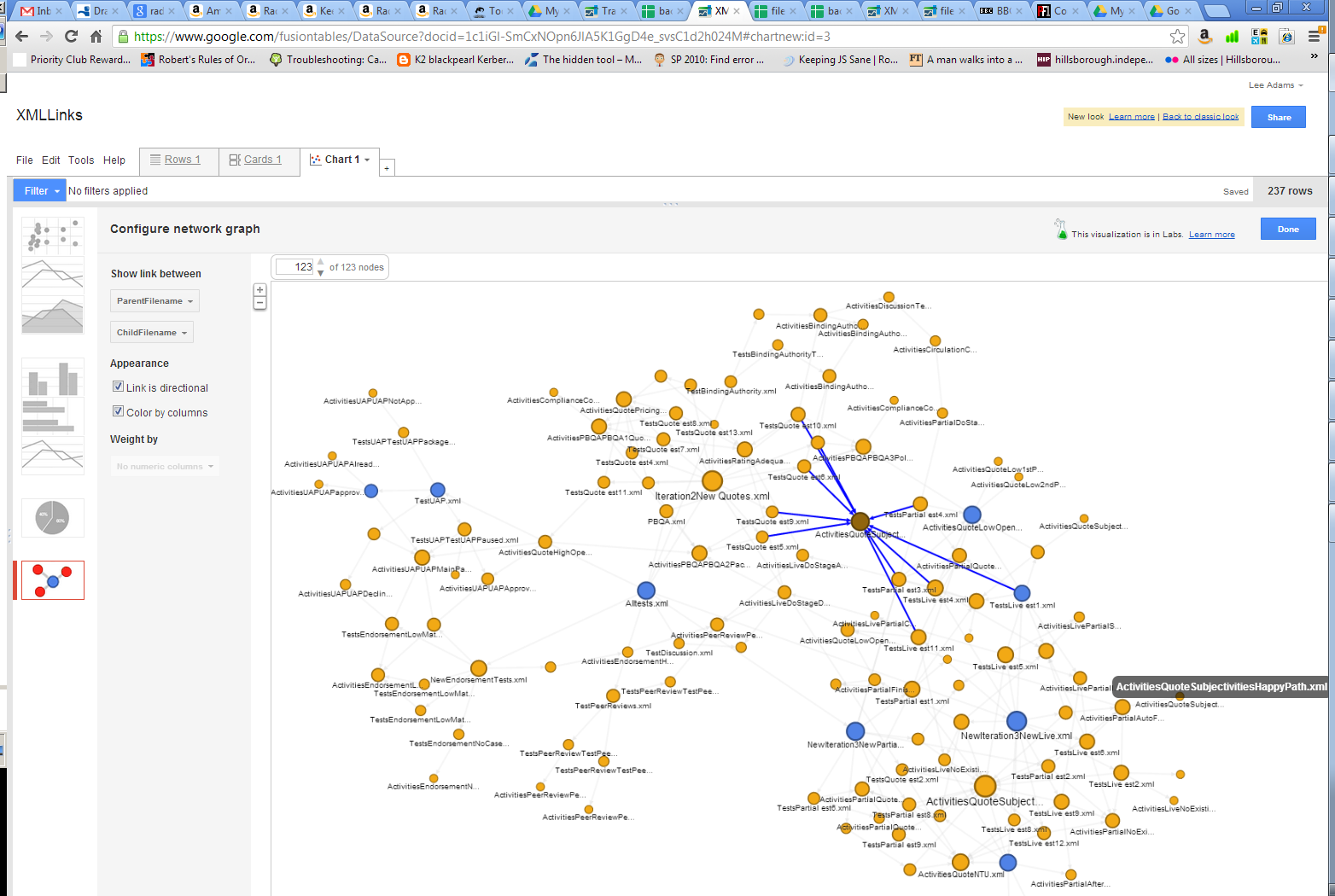
Links to xml files can get messy (see below) which is why I have designed a SmartBox database which can record the structure. Once deployed select your parent test file and select the Save Test Structure checkbox and run the tests. This should record the following rows in the FilenameRelationship smartbox:

|  |  |
| --- | --- |
| All.xml | AllTravel.xml |
| AllTravel.xml | Tests\Travel\1.Simple.xml |
| Tests\Travel\1.Simple.xml | Activities\Travel\BookHotelOnly.xml |
| AllTravel.xml | Tests\Travel\2.SimpleWithFlight.xml |
| Tests\Travel\2.SimpleWithFlight.xml | Activities\Travel\BookFlightsAndHotel.xml |
| AllTravel.xml | Tests\Travel\3.Complex.xml |
| Tests\Travel\3.Complex.xml | Activities\AA\NoDirector-Approved.xml |
| Tests\Travel\3.Complex.xml | Activities\Travel\BookHotelOnly.xml |
| AllTravel.xml | Tests\Travel\4.SimpleReject.xml |
| AllTravel.xml | Tests\Travel\5.AdvancedReject.xml |
| Tests\Travel\5.AdvancedReject.xml | Activities\AA\NoDirector-Rejected.xml |
| All.xml | AllAA.xml |
| AllAA.xml | Tests\AA\1.NoDirector-Approved.xml |
| Tests\AA\1.NoDirector-Approved.xml | Activities\AA\NoDirector-Approved.xml |
| AllAA.xml | Tests\AA\2.NoDirector-Rejected.xml |
| Tests\AA\2.NoDirector-Rejected.xml | Activities\AA\NoDirector-Rejected.xml |
| AllAA.xml | Tests\AA\3.Director-Approved.xml |
| Tests\AA\3.Director-Approved.xml | Activities\AA\Director-Approved.xml |
| AllAA.xml | Tests\AA\4.Director-Rejected.xml |
| Tests\AA\4.Director-Rejected.xml | Activities\AA\Director-Rejected.xml |

Which can be visualized (I used Google Fusion Tables) in a network map



A closer look at the above can be found [here](https://www.google.com/fusiontables/DataSource?docid=1TWRtTwogLL33KeMZFTRmFbYYDpJ8FdlilPjZR9o)



The above complex visualization involves 12 different processes (each with at least 2 client events and which could be started in any order, but all must be run once and only once) can be found [here](https://www.google.com/fusiontables/DataSource?docid=1c1iGl-SmCxNOpn6JlA5K1GgD4e_svsC1d2h024M)

If you have any problems or feature requests with this testing tool please email lee@k2.com

Regards,

Lee Adams.