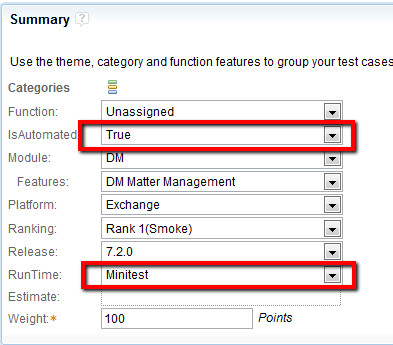
Ruby Minitest Support of Galaxy

This document is to introduce how Galaxy supports the Ruby Minitest.

## To support the ruby minitest, the user needs to do below actions:

1. Update the test case properties in RQM

* Specify the test case’s runtime as Minitest:



* Set IsAutomated as True.

1. Automation the test case using Ruby Minitest:

* Reference the saber\_base.rb in folder of SourceOne/Common/saber\_base which contains the utilities to interact with S1, such as injest test data into it.
* Reference the minitest\_saber\_base.rb which is used to generate the result file name as the web id of the test case in RQM
* Define your class by inherit from the SaberTestBase class which is defined in the file of minitest\_saber\_base.rb
* In your setup of class, you can call the ingest\_test\_data\_to\_sourceone(path) (path parameter is optional) to ingest test data into S1, the function will:
  + ingest all the test data in the folder of path to Exchange(\*.pst and \*.tsv files)
  + Create the archive connection
  + Configure the workers, native archive server and so on
  + Create the archive folder and mapped folder with specific permissions
  + Create a policy
  + Create a Historical Archive activity in S1
  + Wait the complete of archiving and indexing of this activity.
* Create your test whose name should be ended with “\_webid\_1234” and 1234 is the web id of the manual test case in RQM.

Below is the sample test case:

*require\_relative '..\..\..\..\Common\minitest\_base\minitest\_saber\_base.rb'*

*require\_relative '..\..\..\..\Common\saber\_base\saber\_base.rb'*

*class TestSuite < SaberTestBase*

*include SaberBase*

*def setup*

*ingest\_test\_data\_to\_souceone(‘C:\testdata’)*

*end*

*def teardown*

*end*

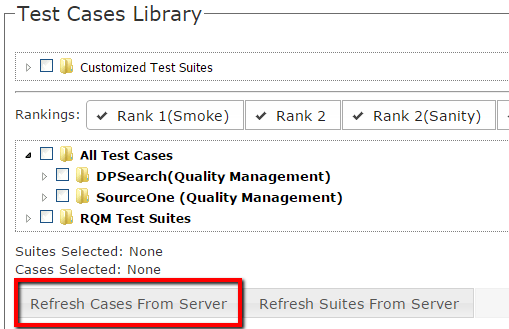
*def test\_workflow\_webid\_1787*

*……*

*End*

*end*

1. Submit your test scripts into the Source Control system
2. Sync the latest test cases manually if it’s not available in the test case library in Galaxy.



1. Select your test cases to submit a task.

## To support the Minitest, the system has done:

1. Install Ruby and required gem into the template
2. For Prepare test data and Interact with S1

* Download the source code of Saber solution into the test agent
* Build the solution on the test agent, the solution contains the utilities the user needs to interact with S1
* The support functions in saber\_base.rb will call the utilities built from the Saber solution to interact with S1

1. For test case execution

* Get the result file which contains the test function which contains a postfix of \_webid\_1234(1234 is the web id of the manual test case) by string pattern matching
* Call the test case by running command

*ruby.exe test\_file.rb -n /\_webid\_1234/*

1. For result parser and send back the test result

We defined the naming pattern of the test case, using \_webid\_1234 as the result file name, where 1234 is the web id of the test case. So the result parser can know where the result file for the test case with specific web id is and how to write back the result.