

RPG Unit Testing Made Easy in VS Code

Alan Seiden · Principal, Seiden Group



IBM i

- Presentation based on material from Sanjula Ganepola (IBM)
 - Software Developer – IBM i App Dev & AI
Sanjula.Ganepola@ibm.com
- Additional content from Patrick Behr

Agenda

- Benefits of Unit Testing
- RPGUnit Testing Tool
- VS Code "IBM i Testing" Extension
- Command line tool (CLI) for IBM i Testing
 - How to run tests outside VS Code?
 - How to run tests in a CI/CD pipeline?
- Can IBM Bob help with unit testing?

Benefits of Unit Testing

- Small, concrete steps
 - You know when you're done
 - Simpler program design
 - All code is tested
-
- **INSPIRES CONFIDENCE AND COURAGE**

Test Phases

1. Environment Setup
2. Call the programs/functions being tested
3. Verify results
4. Cleanup



xUnit

Test automation frameworks

xUNIT

Test automation frameworks

SUnit - Smalltalk

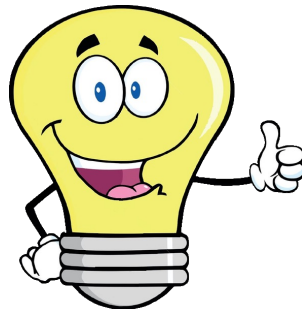
JUnit - Java

PHPUnit – PHP

PyUnit - Python

NUnit - .NET

RPGUnit - RPG



~~RPGUnit~~

iRPGUnit

ILEUnit

IBMiUnit

1. Environment setup

- `setupSuite()`

Runs once before the start of any tests

- `setup()`

Runs once before each of your tests

2. Call the program/function being tested

This can be the tricky part...

Getting the program/procedure into a state where it can be easily tested

3. Verify results

- Verify return values
- Verify spooled files
- Verify data

Tests Use **Assertions** with boolean pass/fail results
(more information to come)

4. Teardown (clean everything up)

- `teardown()`
Runs once after each of your tests
- `teardownSuite()`
Runs once after all tests have run

Assertions using assert()

```
actualValue = AddTwoNumbers(2 : 2);  
expectedValue = 4;  
  
assertEqual(expectedValue : actualValue : 'Error message');  
  
if expectedValue <> actualValue;  
    sendFailedMessage('Error message');  
    failedTests += 1;  
else;  
    passedTests += 1;  
endif;
```



xUnit

Example Assertions with assert()

Syntax

```
assert( booleanExpression [ : 'message' ] );
```

Example

```
//  Pass  
assert( 1=1 : 'Impossible to fail' );  
  
//  Fail  
assert( 1=0 : 'Obviously wrong' );
```

xUnit

Example Assertions with assertEquals()

Syntax

```
assertEquals( 'expected' : 'actual' [: 'message' ] );
```

Example

```
// ✅ Pass
assertEquals( 'Hello' : 'Hello' );
assertEquals( 123 : 123 );

// ❌ Fail
assertEquals( 123.45 : 123.46 );
assertEquals( D'2024-09-16' : D'2023-01-11' );
assertEquals( T'11.22.33' : T'22.33.44' );
assertEquals( Z'2024-09-16-11.22.33.123456' : Z'2024-01-11-11.22.33.123456' );
```

xUnit

Assertions are Helpful in Test Cases

- A **test case** contains testing code, including assertions
 - Example coming up
- A **test suite** contains multiple test cases



Test Cases

Test Case Using Assertions

```
testAddTwoNumbers {  
  
    actualValue = AddTwoNumbers(2 : 2);  
    expectedValue = 4;  
  
    assertEquals(expectedValue : actualValue : 'Error message');  
    assertEquals(expectedOther : actualOther : 'Error message');  
    assertEquals(...etc...);  
}
```



Test Suite

Test Suite with Multiple Test Cases

```
ArithmeticTests {
```

```
    setup{...}
```

```
    testAddTwoNumbers{...}
```

```
    testSubtract{...}
```

```
    testDivide{...}
```

```
    testMultiply{...}
```

```
    teardown{...}
```

```
}
```

environment setup

call test, verify results

call test, verify results

call test, verify results

call test, verify results

environment cleanup

RPGUnit / iRPGUnit

<https://sourceforge.net/projects/irpgunit/>

💡 Add the copybook: `/include
qinclude,TESTCASE`

Naming requirements

iRPGUnit supports both *SRVPGM and *PGM programs.

At this time, the IBM i Testing Extension supports only service programs.

A **test case** is an exported procedure

- starts with the name `test`

Examples:

`test_getEmployeeDetail`

`test_getDeptDetail`

`testIsPalindrome`

`testFactorial`

A **test suite** is a service program containing test case procedures

- Test suite program has no naming requirements

VS Code IBM i Testing Extension naming requirements:

- Local source files: must be suffixed with `.test.rpgle`, `.test.sqlrpgle`, `.test.cbllc`, or `.test.sqlcbllc`
- Source members: recommended: prefix or suffix with `T`, `_T`, `T_`, etc.

Example Includes for RPGUnit

```
**free  
  
ctl-opt nomain;  
  
/include rpgunit/qinclude,TESTCASE  
  
/include SRVMATH_H
```

Test Case Procedure in RPGUnit

```
// -----  
// Add two numbers  
// -----  
dcl-proc test_AddTwoNumbers export;  
  
  dcl-s actualValue packed(7);  
  dcl-s expectedValue packed(7);  
  
  actualValue = AddTwoNumbers(2 : 2);  
  expectedValue = 4;  
  
  iEqual( expectedValue : actualValue);  
  
end-proc test_AddTwoNumbers;
```

Test case (full code)

```
**free
ctl-opt nomain;
/include rpgunit/qinclude,TESTCASE
/include SRVMATH_H

dcl-proc test_AddTwoNumbers export;
  dcl-s actualValue packed(7);
  dcl-s expectedValue packed(7);

  actualValue = AddTwoNumbers(2 : 2);
  expectedValue = 4;

  iEqual(expectedValue : actualValue);

end-proc test_AddTwoNumbers;
```

Assertion type "assert" in RPGUnit

```
dcl-proc assert;  
  condition    ind const;  
  messageFalse varchar(16384) const;  
end-proc assert;  
  
assert( a > b : 'A is not greater than B' );  
  
assert( %found(FILE) : 'Missing record in file FILE' );
```

Assertion type "aEqual" in RPGUnit

```
dcl-proc aEqual;  
  expectedValue char(32565) const;  
  actualValue   char(32565) const;  
  fieldname     varchar(64) const options(*nopass: *omit);  
end-proc aEqual;
```

```
aEqual('a' : 'b');  
// Expected 'a', but was 'b'.
```

```
aEqual('a' : 'b' : 'Switch');  
// Switch: Expected 'a', but was 'b'.
```

Assertion type "iEqual" in RPGUnit

```
dcl-proc iEqual;  
  expectedValue zoned(31) const;  
  actualValue   zoned(31) const;  
  fieldname     varchar(64) const options(*nopass: *omit); end-proc iEqual;  
end-proc iEqual;
```

```
iEqual(5 : 6);  
// Expected 5, but was 6.
```

```
iEqual(5 : 6 : 'Widget');  
// Widget: Expected 5, but was 6.
```

Assertion type "nEqual" in RPGUnit

```
dcl-proc nEqual;  
  expectedValue ind const;  
  actualValue   ind const;  
  fieldName     varchar(64) const;  
end-proc nEqual;
```

```
nEqual(*ON : *OFF);  
// Expected '1', but was '0'.
```

```
nEqual (*ON : *OFF : 'Alarm');  
// Alarm: Expected '1', but was '0'.
```

Utility to get joblog errors

```
monitor;  
  a = 10;  
  b = 0;  
  c = a / b;  
  
  // ❌ Fail: Only runs if exception was not caught  
  fail('Division by zero did not raise an error.');
```

on-error;

```
  msgInfo = getMonitoredMessage(*ON);  
endmon;
```



```
// ✅ Pass  
aEqual( 'MCH1211' : msgInfo.Id );
```

💡 Add the copybook: `/include
qinclude,TESTCASE`

Compile and execution commands

Compile Command:

```
RPGUNIT/RUCRTRPG TSTPGM(MYUSER/TEMPDET) SRCSTMF('/home/MYUSER/builds/  
ibmi-company_system/qtestsrc/empdet.test.sqlrpgle') TGTCCSID(*JOB)  
DBGVIEW(*SOURCE) COPTION(*EVENTF) RPGPPOPT(*LVL2) INCDIR(''/home/  
MYUSER/builds/ibmi-company_system'')
```

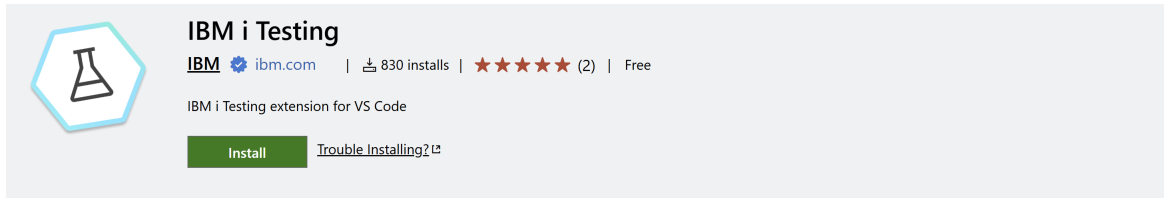
Execution Command:


```
RPGUNIT/RUCALLTST TSTPGM(MYUSER/TEMPDET) ORDER(*REVERSE) DETAIL(*ALL)  
OUTPUT(*ALLWAYS) LIBL(*CURRENT) JOBD(*DFT) RCLRSC(*NO) XMLSTMF('/tmp/  
testing/vscode-ibmi-testing/RPGUNIT/TEMPDET_1751646914682.xml')
```



IBM i Testing Extension for VS Code

Installable via the Marketplace



IBM i Testing
IBM  | 830 installs | ★★★★★ (2) | Free
IBM i Testing extension for VS Code
[Install](#) [Trouble Installing?](#)

[Overview](#) [Version History](#) [Q & A](#) [Rating & Review](#)

IBM i Testing

version rate limited by upstream service
vs marketplace rate limited by upstream service

The [IBM i Testing](#) VS Code extension empowers developers to run unit tests and generate code coverage results for RPG and COBOL programs on IBM i with ease. Under the covers, this extension leverages the [RPGUnit](#) testing framework. It not only simplifies compiling and running tests but also accelerates test creation with built-in stub generation, making it easy to get started. For automation, the companion [IBM i Testing CLI](#) lets you run tests in your terminal on your local PC or in PASE on IBM i, enabling you to script the execution of tests in a CI/CD pipeline.



- 👤 **Run Tests:** Visualize and run tests suites out of local files or source members.
- 🔥 **Generate Stubs:** Generate test stubs for individual test cases or an entire test suite.

Categories

Testing

Tags

as400 ibmi irpgunit iseries json test

Works with

Universal

Resources

[Issues](#)
[Repository](#)
[Homepage](#)
[License](#)
[Changelog](#)

<https://marketplace.visualstudio.com/items?itemName=IBM.vscod-ibmi-testing>

Additional Extensions

Extensions from VS Code Marketplace or Open VSX Registry:

[IBM i Testing](#) ***(Required)***

[Code for IBM i](#) ***(Required - 2.16.0 minimum)***

[RPGLE](#) ***(Required - 0.31.0 minimum)***

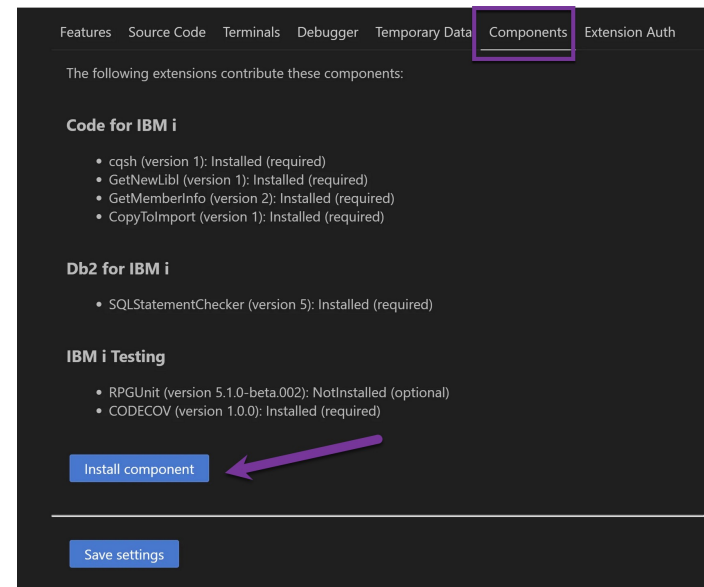
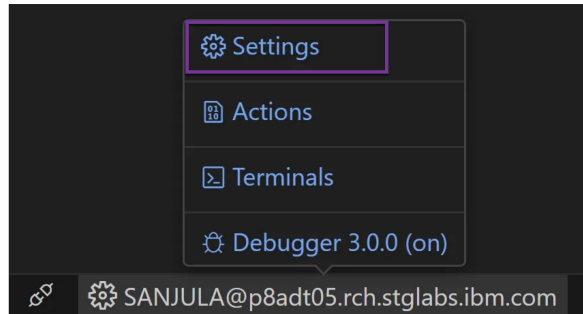
[COBOL](#) ***(Optional - Install if writing COBOL unit tests)***

*Once you have these, you can install iRPGUnit easily from VS Code
(Instructions coming up)*

<https://codefori.github.io/docs/developing/testing/overview/>

Install RPGUnit library and CODECOV

Hover over Settings, then click "Components" →



<https://codefori.github.io/docs/developing/testing/overview/>

Extension lets you generate test cases

```
empdet.sqlrpgle ●
qrplesrc > empdet.sqlrpgle > getEmployeeDetail
7 dcl-proc getEmployeeDetail export;
8 dcl-pi *n like(employee_detail_t);
9 empno char(6) const,
10
11
12 Generate test case for 'getEmployeeDetail' (employee_detail_t);
13 Generate test suite for 'empdet.sqlrpgle'
14
15 select
16     rtrim(firstname) || ' ' || rtrim(midinit) || ' '
17     salary + bonus + comm
18 into
19     :employee_detail.name,
20     :employee_detail.netincome
21 from
22     employee
```

Result

```
1  **free
2  ⚡
3  // ctl-opt nomain ccsidcvt(*excp) ccsid(*char : *jobrun);
4
5  //include qinclude,TESTCASE
6  //include 'qrpgleref/empdet.rpgleinc'
7
8  // dcl-proc test_getEmployeeDetail export;
9  //   dcl-pi *n extproc(*dclcase) end-pi;
10
11 //   dcl-s empno char(6);
12 //   dcl-ds actual likeDs(employee_detail_t) inz;
13 //   dcl-ds expected likeDs(employee_detail_t) inz;
14
15 //   empno = '..';
16
17 //   actual = getEmployeeDetail(empno);
18
19 //   expected.found = *off;
20 //   expected.name = '..';
21 //   expected.netincome = 0.0;
22
23 //   nEqual(expected.found : actual.found : 'found');
24 //   assert(expected.name = actual.name : 'name');
25 //   assert(expected.netincome = actual.netincome : 'netincome');
26 // end-proc;
```

Now edit your test case

```
qtestsrc > ≡ empdet.test.sqlrpgle > ...
1 1 **free
2 2
3 - ctl-opt nomain ccsidcvt(*excp) ccsid(*char : *jobrun);
3+ ctl-opt nomain ccsidcvt(*excp) ccsid(*char : *jobrun) bnmdir('APP');
4 4
5 5 /include qinclude,TESTCASE
6 6 /include 'qrppleref/empdet.rpgleinc'
7 7
8 > 8 dcl-proc test_getEmployeeDetail export;
9 9 | dcl-pi *n extproc(*dclcase) end-pi;
10 10
11 11 | dcl-s empno char(6);
12 12 | dcl-ds actual likeDs(employee_detail_t) inz;
13 13 | dcl-ds expected likeDs(employee_detail_t) inz;
14 14
15 - | empno = '';
15+ | empno = '000010';
16 16
17 17 | actual = getEmployeeDetail(empno);
18 18
19 - | expected.found = *off;
20 - | expected.name = '';
21 - | expected.netincome = 0.0;
19+ | expected.found = *on;
20+ | expected.name = 'CHRISTINE I HAAS';
21+ | expected.netincome = 57970;
22 22
23 23 | nEqual(expected.found : actual.found : 'found');
24 24 | assert(expected.name = actual.name : 'name');
25 25 | assert(expected.netincome = actual.netincome : 'netincome');
26 26 | end-proc;
```

Generate more test cases (showing preview mode)

The screenshot displays the IBM i Testing Extension interface. The top section shows a code editor with the following code:

```
8+dcl-pr getDeptDetail like(department_detail_t) extproc('GETDEPTDETAIL');
9+ deptno char(3) const;
10+end-pr;
7
11
8 12 dcl-proc test_getEmployeeDetail export;
9 13 dcl-pi *n extproc(*dclcase) end-pi;
```

Below the code editor, there is a section labeled "14 hidden lines" which contains the following code:

```
24 28 assert(expected.name = actual.name : 'name');
25 29 assert(expected.netincome = actual.netincome : 'netincome');
26 30 end-proc;
31+
32+dcl-proc test_getDeptDetail export;
33+ dcl-pi *n extproc(*dclcase) end-pi;
34+
35+ dcl-s deptno char(3);
36+ dcl-ds actual likeDs(department_detail_t) inz;
37+ dcl-ds expected likeDs(department_detail_t) inz;
38+
39+ deptno = '';
```

The bottom section of the interface is a "REFACTOR PREVIEW" panel, which is highlighted with a red border. It contains the following options:

- PROBLEMS
- DEBUG CONSOLE
- OUTPUT
- TERMINAL
- IBM I JOB LOG
- REFACTOR PREVIEW

The "Add prototype(s)" section is expanded, showing a checked checkbox and the following code snippet:

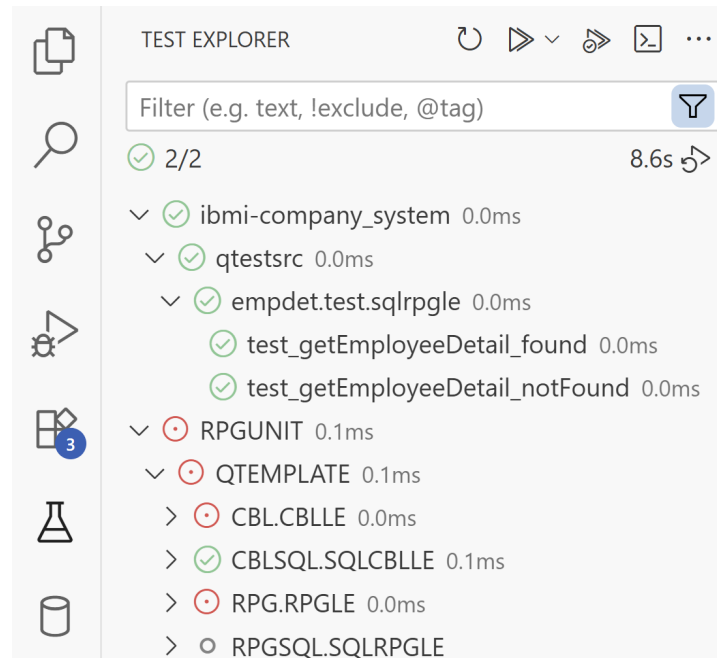
```
dcl-pr getDeptDetail like(department_detail_t) extproc('GETDEPTDETAIL'); deptno char(3) const; end-
```

The "Add test case(s)" section is also expanded, showing a checked checkbox and the following code snippet:

```
end-proc; dcl-dcl-proc test_getDeptDetail export; dcl-pi *n extproc(*dclcase) end-pi; dcl-s deptno
```

At the bottom of the "REFACTOR PREVIEW" panel, there are two buttons: "Apply" and "Discard".

See tests in Text Explorer



See tests in code editor

```
empdet.test.sqlrpgle M X
qttestsrc > empdet.test.sqlrpgle > test_getEmployeeDetail_notFound
⊗ 61 dcl-proc test_getEmployeeDetail_notFound export;
62   dcl-pi *n extproc(*dclcase) end-pi;
63
64   dcl-s empno char(6);
65   dcl-ds actual likeDs(employee_detail_t) inz;
66   dcl-ds expected likeDs(employee_detail_t) inz;
67
68   empno = '111111';
69   actual = getEmployeeDetail(empno);
70
71   expected.found = *on;
72
73   nEqual(expected.found : actual.found : 'found');
74 end-proc;
75
⊙ 76 dcl-proc test_getDeptDetail_found export;
77   dcl-pi *n extproc(*dclcase) end-pi;
```

Configure tests

```
testing.json 1 ● [Icons]
```

```
qtestsrc > {} testing.json > {} rpgunit > {} rucrtrpg
```

```
1 {
2   "rpgunit": {
3     "rucrtrpg": {
4       "tgtCcsid": "*JOB",
5       "dbgView": "*SOURCE",
6       "rpgPpOpt": "*LVL2",
7       "cOption": [
8         "*EVENTF"
9       ]
10    }
11  },
12 },
13 "codeco
14   "mo
15   compileOpt
16   define
17   dltSplf
18   }
19 }
```

- actGrp Activation group
- bndDir
- bndSrvPgm Refer to the ACTGRP parameter in CRTSRVPGM command help.
- bOption
- compileOpt
- define
- dltSplf
- incDir
- module

Run tests and view results

The screenshot displays the IBM i Testing Extension interface. On the left, the TEST EXPLORER shows a tree view of test files under the project 'ibmi-company_system'. The main editor shows the source code for 'test_getEmployeeDetail_found', which includes a dcl-proc and dcl-ds definitions. The bottom panel displays the test execution results, including a summary of successful deployments and compilations, and a detailed results section showing 1 passed test file, 4 passed test cases, 9 assertions, and a duration of 0.004s. A green 'PASS' indicator is visible at the bottom of the results section.

```
empdet.test.sqllrpgle x
qttestsrc > empdet.test.sqllrpgle > test_getEmployeeDetail_found
42 dcl-proc test_getEmployeeDetail_found export;
43   dcl-pi *n extproc(*dclcase) end-pi;
44
45   dcl-s empno char(6);
46   dcl-ds actual likeDs(employee_detail_t) inz;
47   dcl-ds expected likeDs(employee_detail_t) inz;
48
```

WORKSPACE ibmi-company_system (1) [Deployment Successful]
> empdet.test.sqllrpgle → TEMPDET.SRVPGM (4) [Compilation Successful]
✓ test_getDeptDetail_found 0.003s
✓ test_getDeptDetail_notFound 0s
✓ test_getEmployeeDetail_found 0.001s
✓ test_getEmployeeDetail_notFound 0s

EXECUTION
Deployments: 1 successful | 0 failed | 0 skipped (1)
Compilations: 1 successful | 0 failed | 0 skipped (1)

RESULTS
Test Files: 1 passed | 0 failed | 0 errored (1)
Test Cases: 4 passed | 0 failed | 0 errored (4)
Assertions: 9
Duration: 0.004s

PASS

View failures and errors

The screenshot displays the IBM i Testing Extension interface. On the left, a tree view shows test results for 'RPGUTILS', including 'TEST_FALSE', 'TEST_FAIL0', 'TEST_FAIL1', and 'rpgstmf.test.rpgle'. The 'rpgstmf.test.rpgle' test is expanded, showing 'testWhatever_1' and 'testWhatever_2' as failed tests. The main area shows the source code for 'rpgstmf.test.rpgle' with a red error message at line 231: 'Name of the test suite should be 'TEMPLATE''. Below the code, the 'PROBLEMS' tab is active, displaying a list of system messages (CPFA0A9, CPI2126, CPI2121, LNC0899, LNC2903, LNC9011, LNC9001, CPF9897) and test results for 'rpgstmf.test.rpgle'.

```
230 // Run
231 assert(sds.pgmName = 'TEMPLATE': 'Name of the test suite should be 'TEMPLATE'');
```

Name of the test suite should be 'TEMPLATE' testWhatever_1
Name of the test suite should be 'TEMPLATE'

232

PROBLEMS 834 DB2 FOR I DEBUG CONSOLE OUTPUT TERMINAL IBM I JOB LOG TEST RESULTS

CPFA0A9: Object not found. Object is EN_US.UTF-8.
CPI2126: AUT parameter ignored.
CPI2121: Replaced object TEARDWNSTE type *MODULE was moved to QRPL0BJ.
LNC0899: Module TEARDWNSTE created in library SANJULA on 10/12/25 at 04:24:24.
LNC2903: Module TEST_TRUE not created in library SANJULA because of source statement errors.
LNC9011: Unable to create debug view.
LNC9001: Compile failed. TEST_FALSE not created.
CPF9897: Unable to create test TCBL.
△ TEST_TRUE
△ TEST_FALSE
△ TEST_FAIL0
△ TEST_FAIL1

> rpgstmf.test.rpgle → TRPGSTMF.SRVPGM (2) [Compilation Successful]
X testWhatever_1 0s
Failure (line 231): Name of the test suite should be 'TEMPLATE'

Run tests with code coverage

TEST EXPLORER

▼ TEST EXPLORER

Filter (e.g. text, !exclude, @tag)

3/8 20.4s

- > ○ RPGUNIT
- ▼ ○ rpgunit-stmf 0.0ms
 - > ○ cbl.test.cblle
 - > ● rpgsqlstmf.test.sqlrpgle 0.0ms
 - > ○ rpgstmf.test.rpgle 0.0ms
- > ○ RPGUTILS

▼ TEST COVERAGE

- ▼ rpgunit-stmf 67.36%
- rpgsqlstmf.test.sqlrpgle 74.32%
- rpgstmf.test.rpgle 60.00%

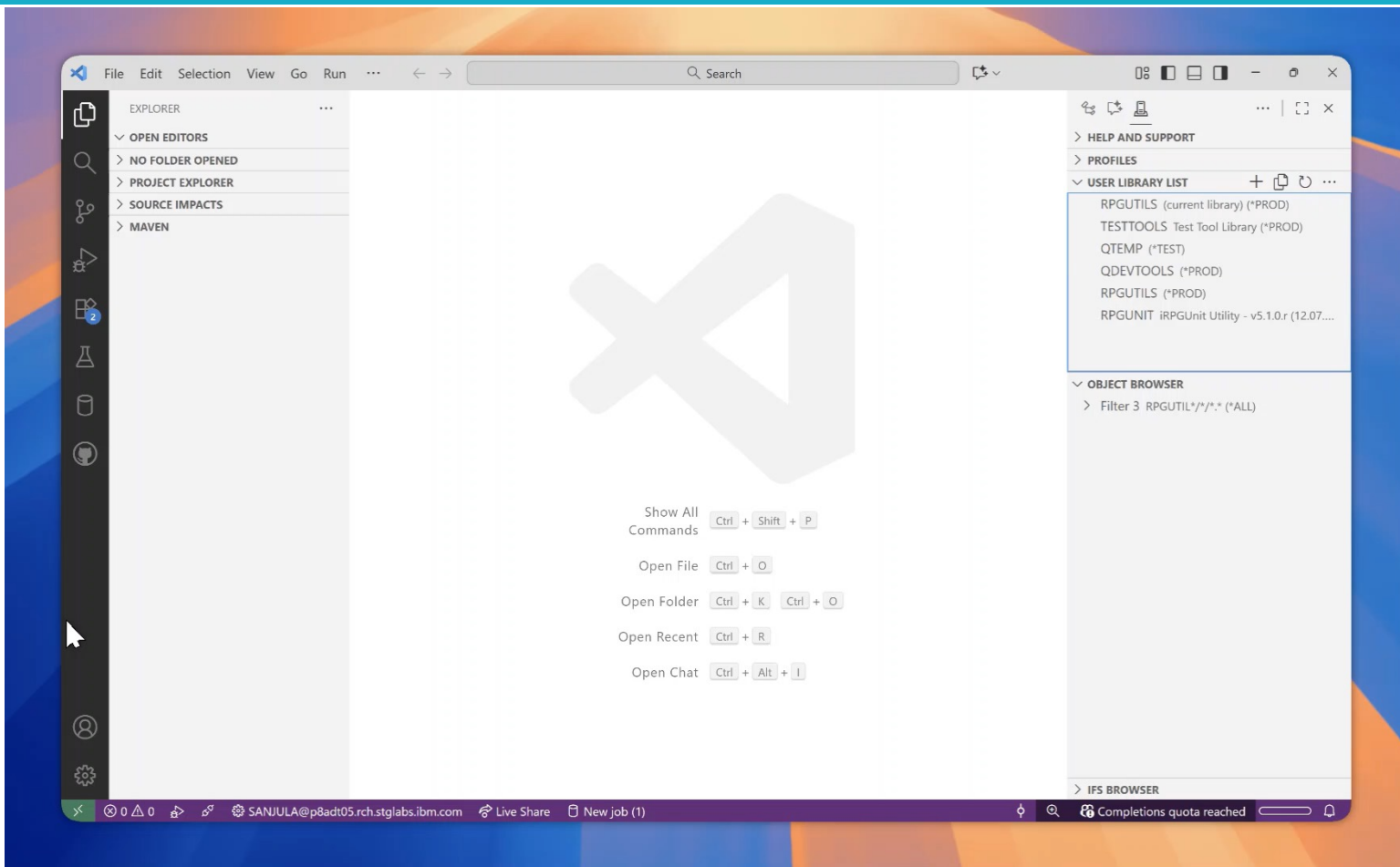
rpgsqlstmf.test.sqlrpgle ×

rpgsqlstmf.test.sqlrpgle > ...

Hide Inline Coverage Rerun

```

127 dcl-proc isSQLError;
139     sqlState = i_state;
140     sql = g_status.sql;
141
142     reset g_status.sql;
143
144     select;
145     // SQL code 00: Unqualified Successful Completion
146     when (sqlState = '00000');
147         // Execution of the operation was successful and did not
148         // result in any type of warning or exception condition.
149         return *off;
150
151     // SQL code 01: Warning
152     when (sqlState.class = '01');
153         // Valid warning SQLSTATES returned by an SQL routine.
154         // Also used for RAISE_ERROR and SIGNAL.
155         if (sql.ignSQLWarn);
156             return *off;
157         else;
158             return *on;
159         endif;
                
```



Documentation for IBM i Testing Extension

Q Search Ctrl K Light

Developing ▼
Editing and compiling
Source Dates
Actions >
Testing **NEW** ▼
 Overview
 Writing Tests
 Running Tests
 Configuring Tests
 CLI & Automated Tests
 Troubleshooting
Debugging **Updated** >
ILEDocs
Local Development >
Browsers ▼
 IBM i Browsers
 Go To File
 IFS Browser
 Object Browser
 User Library List
Languages ▼
 C
 COBOL
 FORTRAN
 RPG
 SQL

Overview

version rate limited by upstream service **installs 71.3**

The [IBM i Testing](#) extension allows developers to run unit tests and generate code coverage results for RPG and COBOL programs on IBM i. Under the covers, this extension leverages the [RPGUnit](#) testing framework.

- **Run Tests:** Visualize and run tests suites out of local files or source members.
- **Generate Stubs:** Generate test stubs for individual test cases or an entire test suite.
- **Configure Tests:** Configure parameters to compile (`RUCRTRPG` / `RUCRTCBL`) and run (`RUCALLTST`) tests.
- **View Test Results:** View detailed test results along with inline failure/error messages.
- **Generate Code Coverage:** View line and procedure level code coverage results as an overlay in the editor.

Installation

1. Extensions

<https://codefori.github.io/docs/developing/testing/overview/>



IBM i Testing CLI

Installable via NPM (Node package manager)



Search packages

Search

Sign Up

Sign In

@ibm/itest TS

1.2.3 • Public • Published a month ago

Readme

Code Beta

5 Dependencies

0 Dependents

2 Versions

IBM i Testing CLI

version **v1.2.3** downloads **70/month**

The **IBM i Testing CLI** (`itest`) is a companion to the **IBM i Testing** VS Code extension, which allows you to run unit tests and generate code coverage results for RPG and COBOL programs on IBM i. With this CLI, you can run tests in your terminal on your local PC or in PASE on IBM i. This enables you to even script the execution of tests in a CI/CD pipeline.



Install

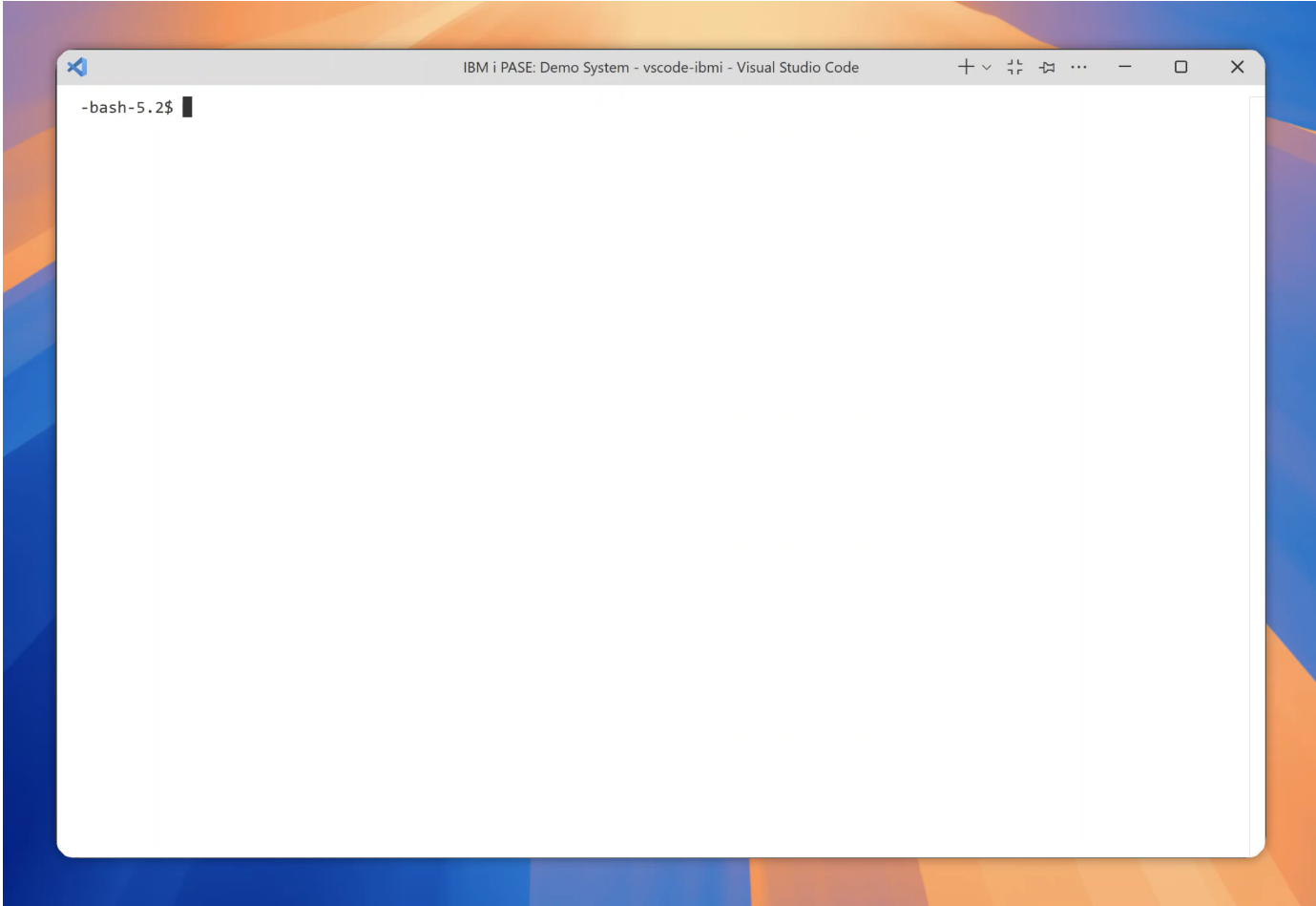
```
> npm i @ibm/itest
```

Repository

[github.com/IBM/vscode-ibmi-testing/b...](https://github.com/IBM/vscode-ibmi-testing/blob/master/README.md)

Homepage

[codefori.github.io/docs/developing/te...](https://codefori.github.io/docs/developing/testing/)



Sample GitHub Action workflow step

```
71 - name: Run Unit Tests
72   id: test
73   continue-on-error: true
74   run: |
75     itest \
76       --ld . \
77       --id /home/$IBMI_USER/builds/ics_${GITHUB_HEAD_REF} \
78       --ll $(so -bl ${GITHUB_HEAD_REF}) CMPSYS RPGUNIT QDEVTOOLS \
79       --cl $(so -bl ${GITHUB_HEAD_REF}) \
80       --cc --sr --tr --to --co
81   env:
82     IBMI_HOST: ${ secrets.IBMI_HOST }
83     IBMI_USER: ${ secrets.IBMI_USER }
84     IBMI_PASSWORD: ${ secrets.IBMI_PASSWORD }
85     IBMI_SSH_PORT: ${ secrets.IBMI_SSH_PORT }
```

https://github.com/IBM/ibmi-company_system/blob/main/.github/workflows/pr.yaml

Pull request triggers impact analysis, build, and tests

The screenshot displays the IBM i Testing CLI interface. On the left, a sidebar lists job steps, with 'Run Unit Tests' highlighted. A red arrow points from this step to the main log area. The main area shows the job 'ibmi' succeeded on Aug 5 in 53s. The 'Run Unit Tests' step is expanded, showing a log of test execution. A dashed box highlights the 'EXECUTION' and 'RESULTS' sections of the log.

```
ibmi
succeeded 4 days ago in 47s

> ✓ Set up job
> ✓ Checkout
> ✓ Setup Node.js 18
> ✓ Install CLI Tools
> ✓ Generate Impact Analysis
> ✓ Adding Impact Analysis to Markdown
> ✓ Post Impact Analysis Comment
> ✓ Generate makefile
> ✓ Run Build
> ✓ Run Unit Tests
> ✓ Add Summary Report to Markdown
> ✓ Post Unit Test Comment
> ✓ Upload Test and Coverage Logs
  ○ Check Test Result
> ✓ Post Setup Node.js 18
> ✓ Post Checkout
> ✓ Complete job
```

```
ibmi
succeeded on Aug 5 in 53s
Search logs

Run Unit Tests 17s
228 IFS example-test-run (1)
229 > empdet.test.sqlrpgle → TEMPDET.SRVPGM (2) [ Compilation Successful ]
230     ✓ test_getEmployeeDetail_found 0.026s
231     ✓ test_getEmployeeDetail_notFound 0.004s
232
233
234 EXECUTION
235 Deployments: 0 successful | 0 failed (0)
236 Compilations: 1 successful | 0 failed | 0 skipped (1)
237
238 RESULTS
239 Test Files: 1 passed | 0 failed | 0 errored (1)
240 Test Cases: 2 passed | 0 failed | 0 errored (2)
241 Assertions: 4
242 Duration: 0.03s
243
244 PASS
245
```

Pull request reviews made easy

ibmi
succeeded 4 days ago in 47s

- > Set up job
- > Checkout
- > Setup Node.js 18
- > Install CLI Tools
- > Generate Impact Analysis
- > Adding Impact Analysis to Markdown
- > Post Impact Analysis Comment
- > Generate makefile
- > Run Build
- > Run Unit Tests
- > Add Summary Report to Markdown
- > Post Unit Test Comment
- > Upload Test and Coverage Logs
- Check Test Result
- > Post Setup Node.js 18
- > Post Checkout
- > Complete job



github-actions (bot) commented on Aug 5

Test and Code Coverage Report

test passing coverage 76%

Test Result

Total Tests	✓ Passed	✗ Failed	⚠ Errored	🎯 Assertions	🕒 Duration
1	1	0	0	4	0.03s

Code Coverage

File	Coverage	Uncovered Lines	Covered Lines	Executable Lines
empdet.sqlrpgle	53%	9	10	19
empdet.test.sqlrpgle	93%	2	25	27



Can Bob Help With Unit Testing?


Run RPGUnit tests against some CRUD
APIs and resolve error

File Edit Selection View Go Run Terminal Help

ibmi-hospital-system

EXPLORER

- IBMI-HOSPITAL-SYSTEM
 - .evfevent
 - .github
 - .logs
 - .vscode
 - qbndsrc
 - qddsrc
 - qrppleref
 - qrpplersrc
 - qsqsrc
 - qsrvsrc
 - qtestsrc
 - .env
 - .gitignore
 - iproj.json
 - LICENSE
 - README.md
 - Rules.mk
- OUTLINE
- TIMELINE
- PROJECT EXPLORER




Show All Commands `Ctrl + Shift + P`

Toggle Terminal `Ctrl + ``

Go to File `Ctrl + P`

Start Workflow Tasks + 74%



Hi, I'm Bob

Ask me questions or let me code for you.

Documentation

What's on your mind?

(@ to add context, / for commands, hold shift to drag in files/images)

IBM Developer ibmi-hospital-syst... Bob - Settings

Generate new RPGUnit tests
for existing CRUD APIs
to improve code coverage

File Edit Selection View Go Run Terminal Help


ibmi-hospital-system

TESTING

Filter (e.g. text, !exclude, @tag)

No test results yet.

- ibmi-hospital-system
 - qtestsrc
 - appt.test.sqlrpgle
 - doctor.test.sqlrpgle
 - test_createDoctor_success
 - test_getDoctor
 - test_updateDoctor_success
 - test_deleteDoctor_success
 - test_listDoctors_success
 - test_getDoctorsBySpecializ...
 - medrec.test.sqlrpgle
 - patient.test.sqlrpgle



Hi, I'm Bob

Ask me questions or let me code for you.

Documentation

RECENT TASKS

See all

One of my tests are failing in @qtestsrc\patient...
0.343 - 16 min ago

What's on your mind?

(@ to add context, / for commands, hold shift to drag in files/images)

IBM | Developer ibmi-hospital-syst... Bob - Settings

feature/patient SANJULA@common1.frankeni.com (Demo) Project: ibmi-hospital-system Deploy New job (1)

Possible future enhancements

- Improve test case generation support
 - Right now the extension now only supports generating stubs for RPG exported procedures, but Sanjula has the idea to add stubs for:
 - RPG programs
 - CL programs
 - SQL procedures / functions / table functions
- Add support for importing and viewing arbitrary CCZIP files (code coverage reports) in VS Code (similar feature exists in RDi)
- Generate PDF reports with test results and code coverage (in the IDE and via a pipeline) rather than just text/markdown file reports
- Add common test utilities as snippets that can be easily accessed by developers
 - Attendees can share any common utilities they would like to add to the extension and share with the community by posting on GitHub: <https://github.com/IBM/vscode-ibmi-testing>

RPGUnit and VS Code Links

IBM i Testing

VS Code Marketplace

<https://marketplace.visualstudio.com/items?itemName=IBM.vscode-ibmi-testing>

Documentation

<https://codefori.github.io/docs/developing/testing/overview/>

GitHub Repository

<https://github.com/IBM/vscode-ibmi-testing>

RPGUnit

Documentation

<https://irpgunit.sourceforge.io/help/>

GitHub Repository

<https://github.com/tools-400/irpgunit>

Sample Projects

Company System Demo

https://github.com/IBM/ibmi-company_system

RPGUTILS

<https://github.com/IBM/vscode-ibmi-testing/tree/main/examples/rpg-utils>

VS Code

Documentation

<https://code.visualstudio.com/docs/editor/testing>



Any Questions?

STAY CONNECTED

Free Resources from Seiden Group

Code for i Fridays



seidengroup.com/vs-code-for-i-fridays

- next session: 19 June at 20:30 CEST (recorded)

IBM i Strategy & Tips — monthly

seidengroup.com/tips

PHP Free Upgrade Assessment

seidengroup.com/free-php-upgrade-assessment

Node.js v20 and Other Updated Packages Require New IBM i Repositories

by Alan Seiden



The latest IBM i open source packages, such as Node.js v20, require new IBM i repositories. We've heard from several perplexed clients wondering why Yum tells them "No package nodejs20 available" during their node upgrade.

This article shows you how to install the latest IBM i repositories to enable your open source updates well into the future . . .

[Read More](#)

What IBM i Users Should Check when Learning of an Open Source Security Vulnerability

by Alan Seiden



Security continues to be a top concern in IT, especially among upper management.

This article shows how to check IBM i open source components to ensure they are at a patched level if a vulnerability is reported . . .

[Read More](#)

STAY CONNECTED

Contact / Stay in Touch

Alan Seiden

Seiden Group

Ho-Ho-Kus, NJ

alan@seidengroup.com

201-447-2437

Monthly Tips:

seidengroup.com/tips

Open Source Support and Seiden PHP+ Server

<https://www.seidengroup.com/seiden-php-for-ibm-i/>

