

IBM Power Week 2025

#pw2025

18 - 19 - 20 novembre 2025

IBM Innovation Studio Paris

S49 – VS Code for i : focus sur le débogage et les tests unitaires

19 novembre 14:45 - 15:45

Philippe Bourgeois
IBM France
pbourgeois@fr.ibm.com



Plan de la présentation

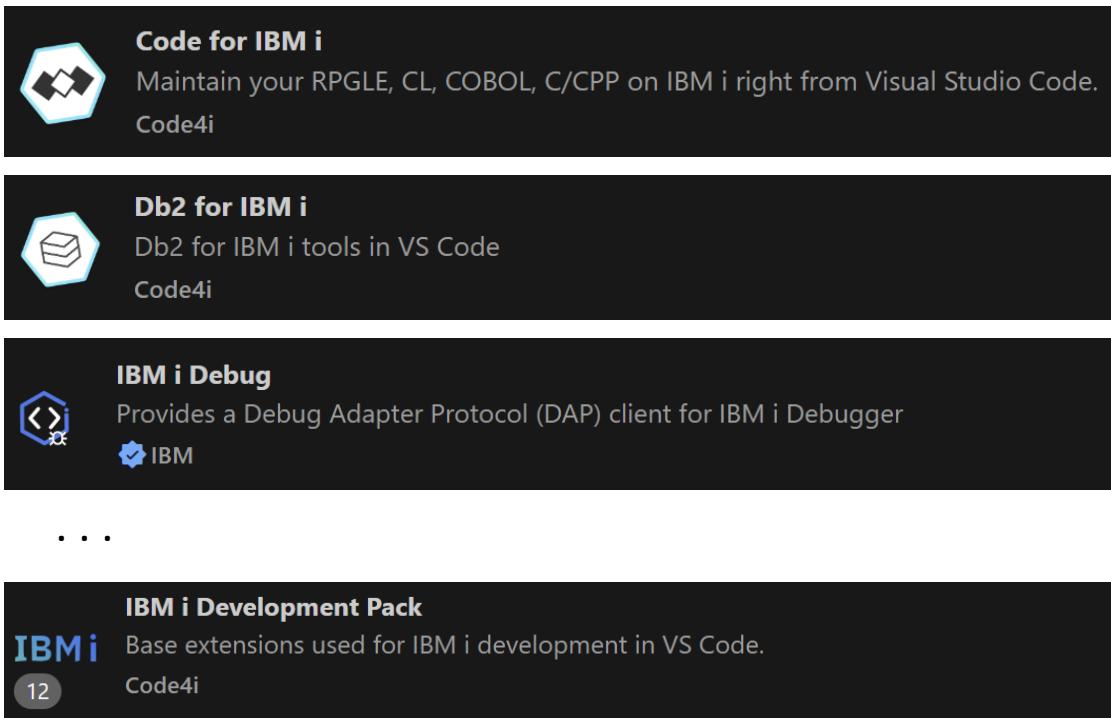
- 1. L'essentiel de VS Code for IBM i
- 2. Le débogage des programmes IBM i avec VS Code
- 3. Les tests unitaires IBM i avec VS Code

1. L'essentiel de VS Code for IBM i



VS Code for IBM i

- VS Code for IBM i = VS Code + des **extensions** IBM i
- Extensions IBM i :
 - Code for i
 - IBM i Languages
 - RPGLE
 - CL
 - COBOL
 - DB2 for i
 - IBM i Debug
 - IBM i Testing
 - IBM i Project Explorer
 - Source Orbit
 - ...
 - IBM i Development Pack

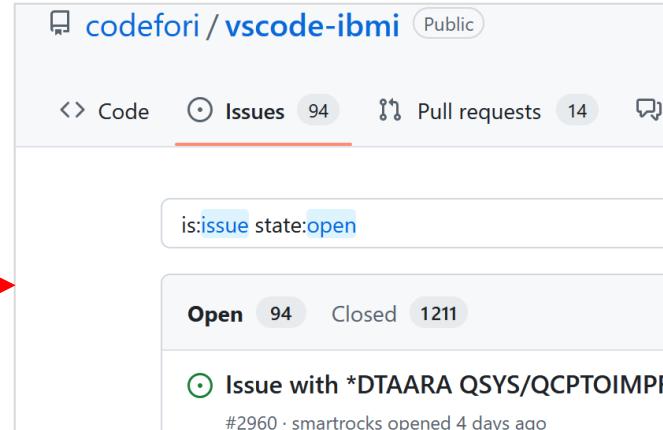
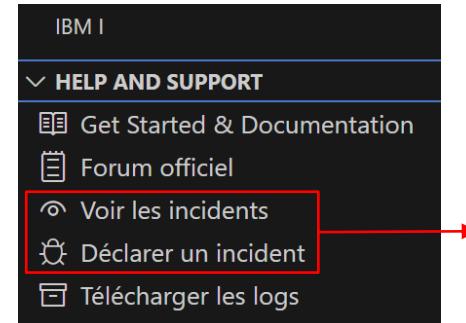


The screenshot shows the Visual Studio Code Marketplace interface with a dark theme. It displays several extensions related to IBM i development:

- Code for IBM i** by Code4i: Maintains RPGLE, CL, COBOL, C/CPP on IBM i right from Visual Studio Code.
- Db2 for IBM i** by Code4i: Db2 for IBM i tools in VS Code.
- IBM i Debug** by IBM: Provides a Debug Adapter Protocol (DAP) client for IBM i Debugger.
- ...
- IBM i Development Pack** by IBMi: Base extensions used for IBM i development in VS Code. (12 reviews)

VS Code for IBM i

- Gratuit
- Support communautaire
OU
- Support IBM



codefori / vscode-ibmi Public

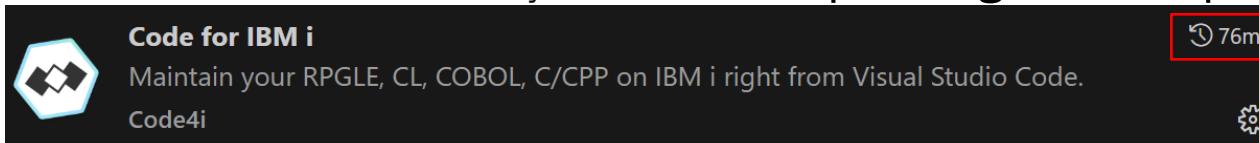
Code Issues 94 Pull requests 14

is:issue state:open

Open 94 Closed 1211

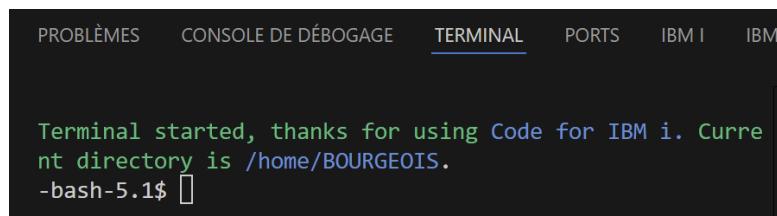
Issue with *DTAARA QSYS/QCPTOIMPF #2960 · smartrocks opened 4 days ago

- "IBM i extensions for VS Code Enterprise Support"
 - Disponible depuis le 31/10/2025
 - Pour toutes les extensions du "IBM i Development Pack"
 - Facturable en plus du support IBM i
 - <https://www.ibm.com/docs/en/announcements/i-extensions-vs-code-enterprise-support>
-
- Installation facile. Mises à jour automatiques, légères et rapides



VS Code for IBM i – Gestion des objets, gestion de l'IFS

- Connexions sécurisées en SSH
- Gestion de la LIBL
- Création de filtres de bibliothèques, objets, membres
- Possibilité de filtres en lecture seule
- Création de raccourcis vers les répertoires de l'IFS
- Infobulles sur les bibliothèques, objets et repertoires IFS
- Création de différents environnement de travail (profils)
- Fonction de recherche IBM i (fichiers source et IFS)
- Terminal 5250
- Terminal PASE
- Download / Upload de fichiers

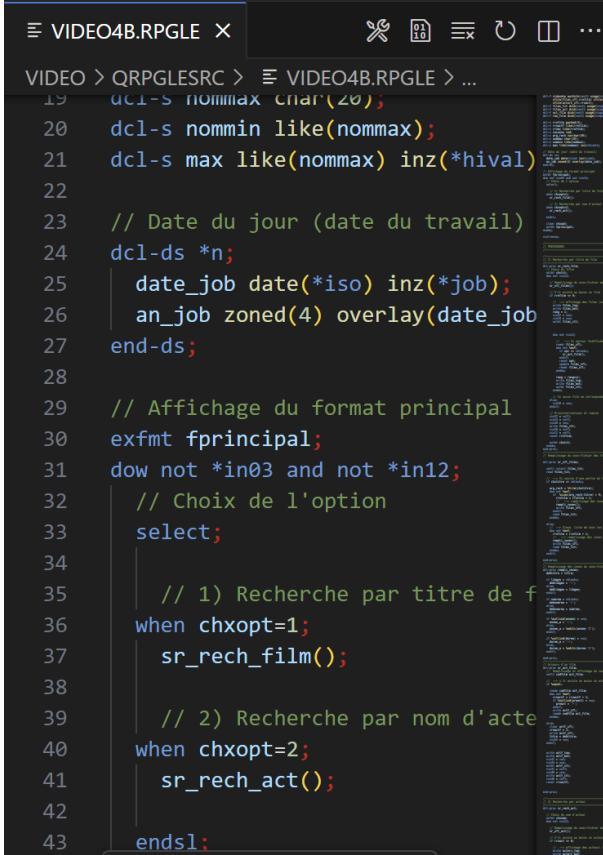


The screenshot shows the VS Code interface for IBM i. At the top, there are tabs for PROBLÈMES, CONSOLE DE DÉBOGAGE, TERMINAL, PORTS, IBM I, and IBM I. The TERMINAL tab is active, showing a terminal window with the following text:
Terminal started, thanks for using Code for IBM i. Current directory is /home/BOURGEOIS.
-bash-5.1\$

- ▽ PROFILES
 - ↻ Reset to Default
 - ❖ Services Web (SOADEMO)
 - ❖ PB Active
 - ❖ RPG4COR
 - ❖ VIDEO
 - ❖ PBTEST
 - ❖ PBOL39
 - ☒ SOASETUPLIBL
- ▽ USER LIBRARY LIST
 - PB (bibliothèque en cours) Philippe Bourgeois
 - QGPL General Purpose Library (*PROD)
 - QTEMP (*TEST)
- ▽ OBJECT BROWSER
 - > Fichiers source de PB PB/*/*.* (*SRCPF)
 - > Δ QGPL QGPL/*/*.* (*)
 - > Programmes de PB PB/*/*.* (*PGM)
 - > Fichiers de PB PB/*/*.* (*FILE)
- ▽ IFS BROWSER
 - > 📂 /pb
 - > 📂 /home/bourgeois

VS Code for IBM i – Edition des sources

- Colorisation des sources
- Vérification de syntaxe
- Complétiion de code
- Ouvertures multiples d'un même source en édition
- Minimap
- Fil d'Ariane
- Mise en commentaires rapide
- Affichage / masquage des blocs de programmation structurée
- Affichage des imbrications de blocs
- Renommage de chaînes de caractères avec contrôle avant validation
- Vue Structure
- Infobulles sur les variables, formats et fichiers
- Formatage de code
- Aperçu des, et accès aux, définitions, références et implémentations
- Refactorisation de code
- Modèles de code
- Prévisualisation des formats des DSPF
- Contrôle de la qualité du code
- Conversion RPG fixe vers RPG Full Free

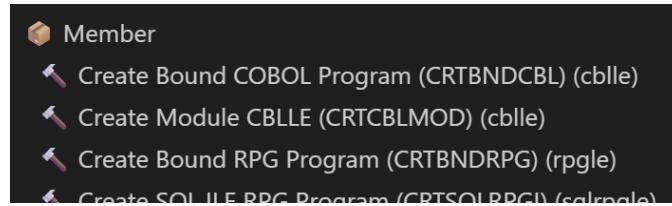
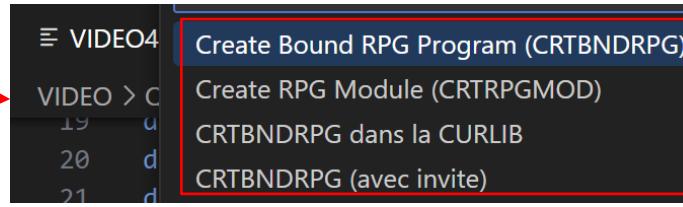


The screenshot shows the VS Code interface with a tab titled "VIDEO4B.RPGLE X". The code editor displays an RPGLE program with syntax highlighting. The code includes various RPG statements like DCL-S, DCL-DS, and EXFMT, as well as comments and logic for date handling and file search. The interface shows the code in a dark theme with color-coded tokens.

```
VIDEO > QRPGLESRC > VIDEO4B.RPGLE > ...
15  dcl-s nommax char(20);
20  dcl-s nommin like(nommax);
21  dcl-s max like(nommax) inz(*hival)
22
23 // Date du jour (date du travail)
24 dcl-ds *n;
25   date_job date(*iso) inz(*job);
26   an_job zoned(4) overlay(date_job
27 end-ds;
28
29 // Affichage du format principal
30 exfmt fprincipal;
31 dow not *in03 and not *in12;
32   // Choix de l'option
33   select;
34
35   // 1) Recherche par titre de f
36   when chxopt=1;
37     sr_rech_film();
38
39   // 2) Recherche par nom d'acte
40   when chxopt=2;
41     sr_rech_act();
42
43 endl;
```

VS Code for IBM i – Compilation des sources

- Lancement par CTRL-E
- Commandes de compilation fournies
- Possibilité de se créer ses propres commandes et variables
- Affichage de la liste des erreurs :
 - Dans le source
 - Vue dédiée ("Problèmes")
- Affichage de la liste de compilation



```
19  dcl-s nomma char(20);    Le nom ou l'indicateur NOMMA n'est pas référencé. (0)
20  dcl-s nommin like(nommax);  Le nom ou l'indicateur NOMMAX n'est pas défini. (30)
21  dcl-s max like(nommax) inz(*hival);  Le nom ou l'indicateur MAX n'est pas défini.
22
23  // Date du jour (date du travail)
24  dcl-ds *n;
25      date_job date(*iso) inz(*job);
26      an job zoned(4) overlaid(date job);  Le nom ou l'indicateur AN JOB n'est pas référencé.
```

PROBLÈMES 19 SORTIE CONSOLE DE DÉBOGAGE TERMINAL PORTS IBM I Filtre (exemple : texte, **/*.ts, !**/no

VIDEO4.RPGLE \VIDEO\QRPGLESRC 19

Arrêt de la compilation/Erreurs de gravité 30 détectées dans le programme. (50) (RNS9308) [Ln 1, Col 1]

Le nom ou l'indicateur NOMMIN n'est pas défini. (30) (RNF7030) [Ln 20, Col 7]

VS Code for IBM i – Gestion de DB2 for i

- **SQL Job Manager**
 - Gestion des jobs SQL et de leurs propriétés
 - Activation de SELF (SQL Error Logging Facility)
- **Gestion des objets DB2 for i (Schema Browser)**
 - Liste des objets DB2 d'un schéma
 - Génération du DDL
 - Affichage du contenu des tables
 - Liste des objets dépendants
 - Liste des index, MTI, index conseillés, droits, verrouillages sur une table et verrouillages sur les enregistrements
- **Exécution de requêtes SQL**
 - Exécution d'une ou plusieurs requêtes
 - Vérification de la syntaxe SQL
 - Affichage / modification des données
 - Sorties JSON, CSV, RPG...
 - Historique des requêtes
 - Exemples de requêtes
 - Complétion de code
 - Liage de variables
 - Exécution de commandes CL
 - Visual Explain
 - Notebooks
 - Sauvegarde des requêtes
- **DB2 for i Code Assistant**
 - Intégration avec des assistants d'IA pour interroger DB2 for i

VS Code for IBM i – Gestion de DB2 for i

The screenshot illustrates the VS Code for IBM i interface, specifically the DB2 FOR I extension. It shows the following components:

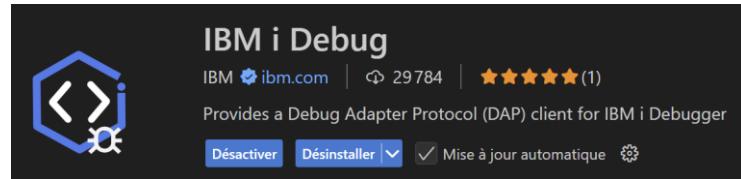
- Schema Browser:** On the left, under the DB2 FOR I section, the **SCHEMA BROWSER** is expanded, showing the **AS425F** schema. Other options like **Aliases**, **Logicals**, **Functions**, **Global Variables**, **Indexes**, **Procedures**, **Tables**, **Triggers**, **Types**, and **Views** are also listed.
- Context Menu:** A context menu is open over the **employees** table, with the **Generate SQL** option highlighted in blue. Other options include **Clear...**, **Copy Data...**, **Delete...**, **Rename...**, **View permissions**, **Advised Indexes**, **Clear All Advised Indexes...**, **Get Related Objects**, **Get Indexes**, **Get MTIs**, **Get Authorities**, **Get Object Locks**, and **Get Record Locks**.
- Code Editor:** The main editor window shows a query: `1 SELECT * FROM AS425F.EMPLOYES;`. Below it, the **RESULTS** tab displays the following data:

MAT	NOM	SRV	POST
10	CHRISTIAN	A00	DIR.
110	VINCENT	B00	DIR.
120	JEAN	E00	COM.

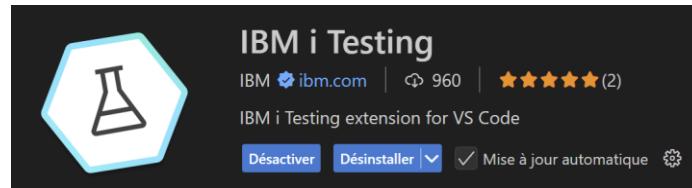
- Chart:** A bar chart on the right shows salary data. The x-axis represents service levels (SRV) and the y-axis represents salary. The chart compares **SALMOYEN** (blue bars) and **SALMAX** (red bars). The data points are: E00 (SALMOYEN: 3871, SALMAX: 6871), B00 (SALMOYEN: 4500, SALMAX: 4500), A00 (SALMOYEN: 4000, SALMAX: 4000), and C00 (SALMOYEN: 2500, SALMAX: 4000).
- Terminal:** A terminal window at the bottom shows a message from a user named **bourgeoisp**: `@db2i Obtenir la moyenne des salaires par service à partir de la table AS425F.EMPLOYES`. A red arrow points from this message to a tooltip in the bottom right corner.
- Tooltip:** A tooltip displays the SQL query: `✓ Accepter les modifications | X Abandonner | Show Changes` followed by the query: `SELECT ·SRV, ·AVG(SALARIS) AS ·Moyenne_Salaire FROM ·AS425F.EMPLOYES GROUP ·BY ·SRV;`

VS Code for IBM i – Autres fonctionnalités / extensions

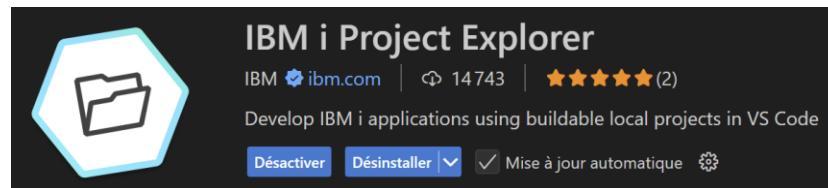
- IBM i Debug
 - Débogage de programmes IBM i



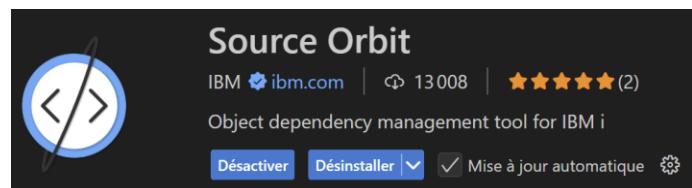
- IBM i Testing
 - Tests unitaires IBM i



- IBM i Project Explorer
 - Développement IBM i avec des projets locaux



- Source Orbit
 - Gestion des dépendances IBM i applicatives

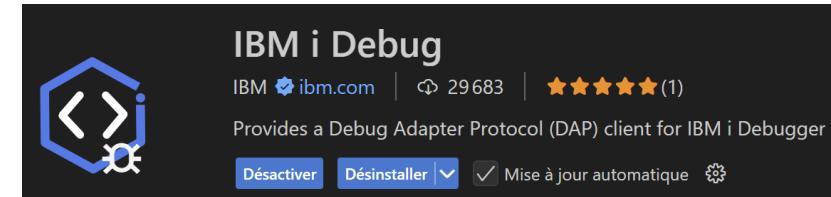


2. Le débogage des programmes IBM i avec VS Code



Débogueur IBM i de VS Code – Fonctionnalités

- Débogage de programmes RPG (OPM et ILE), COBOL, C/C++, CL
- Débogage pas à pas dans le source
 - STEP OVER, STEP INTO, STEP OUT
- Points d'arrêt simples ou conditionnés
 - Peuvent être ajoutés avant ou pendant le débogage
- Affichage et modification de variables
- Monitoring de variables
- Affichage de la pile d'appel des programmes
- Possibilité de mettre à jour les "fichiers de production" (~ UPDPRD(*YES))
- Points d'entrée de service 
 - Permet de mettre le programme en "attente de débogage"
 - Quand le programme est exécuté (quel que soit le job), la session de débogage est lancée dans VS Code
 - Avec possibilité de choisir le profil utilisateur, une condition d'exécution et de rafraîchir le point d'entrée de service après recompilation



Débogueur IBM i de VS Code – Versions

- Au 19/11/2025 la version du débogueur est la 3.0.1 :
 - <https://codefori.github.io/docs/developing/debug/overview/>
 - <https://codefori.github.io/docs/developing/debug/configure/>

Developing

Editing and compiling

Source Dates

Actions > Version 3.0.1 Version 3 Version 2.0.2 Version 2.0.0 Version 1

Testing NEW >

Debugging Updated >

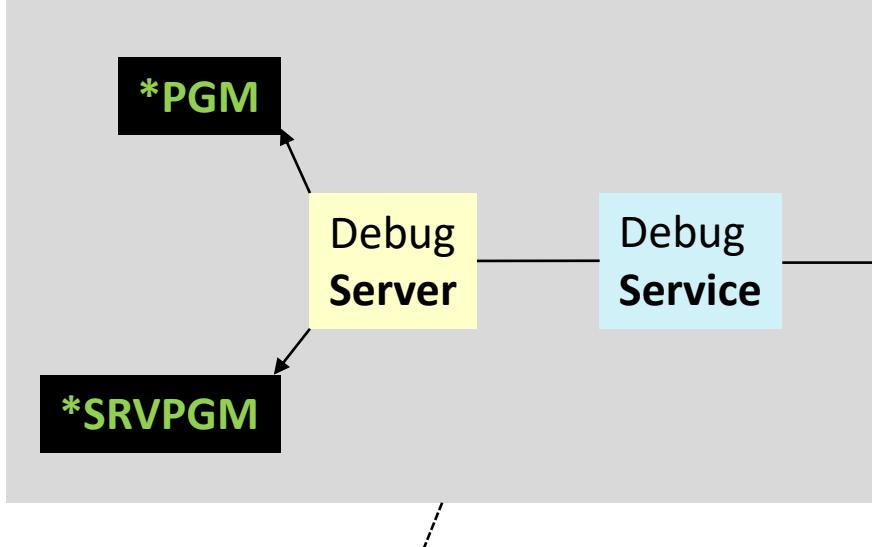
Overview

Configuring



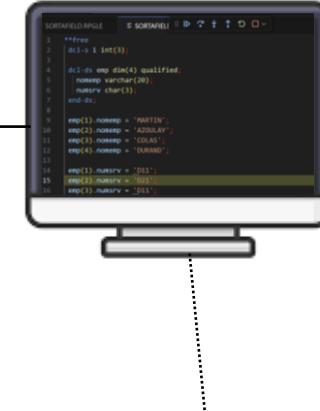
Débogueur IBM i – Composants et prérequis

IBM i



- 5770-JV1 Option 19 (Java 11) ou Option 20 (Java 17)
- 5770-WDS Option 60
- PTFs (<https://codefori.github.io/docs/developing/debug/configure/>)
- 1 certificat dans le magasin *SYSTEM de DCM et affecté au Debug Service
- Démarrer le Debug Server
- Démarrer le Debug Service

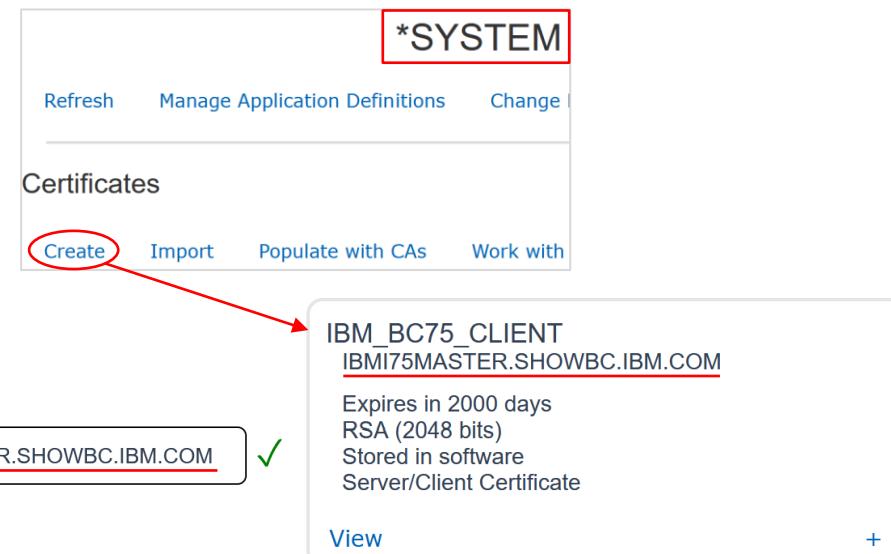
VS Code



- Extension "IBM i Debug"
- BASH comme shell par défaut
- 1 connexion IBM i avec un nom de hôte et non pas une adresse IP
- Importer le certificat créé sur le serveur

Débogueur IBM i – Mise en oeuvre

- Côté IBM i (1/2)
 - Le certificat créé dans DCM doit avoir comme CN (Common Name) le nom du hôte défini dans la connexion IBM i de VS Code



The screenshot shows the 'Certificates' section of the IBM i interface. A red box highlights the 'SYSTEM' label in the top right corner. Below it, a red circle highlights the 'Create' button in the top navigation bar. A red arrow points from this 'Create' button to a detailed view of a newly created certificate. The certificate details are as follows:

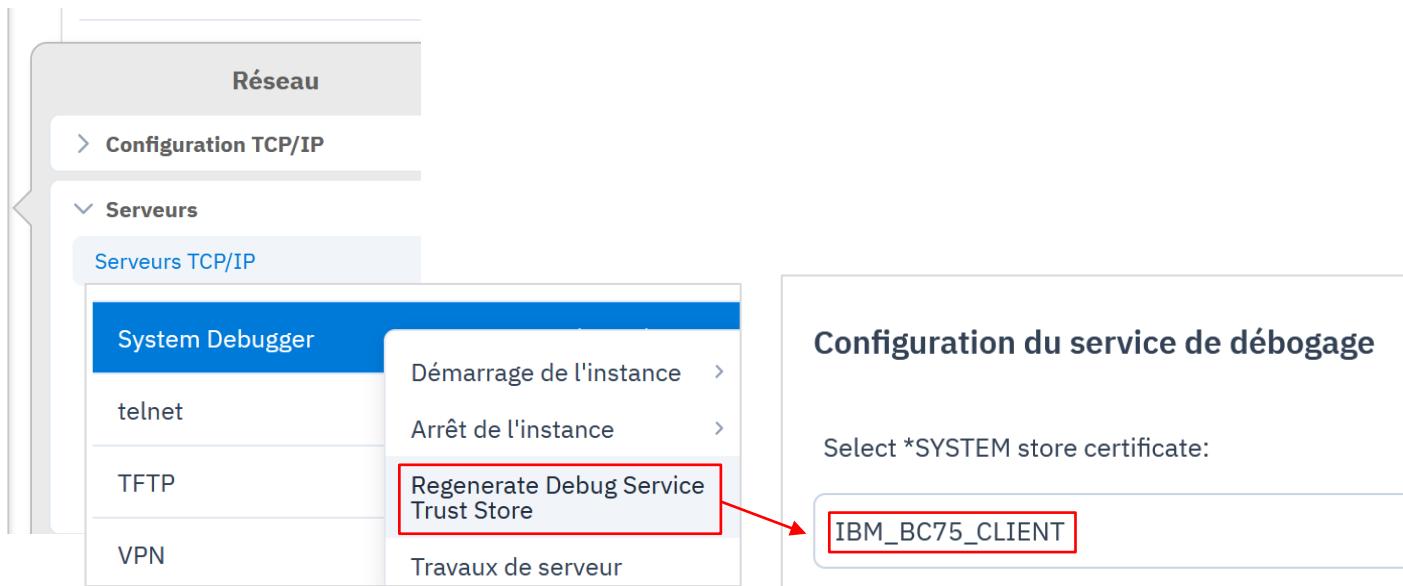
- Subject: IBM_BC75_CLIENT
- Common Name: IBMI75MASTER.SHOWBC.IBM.COM
- Expires in 2000 days
- RSA (2048 bits)
- Stored in software
- Server/Client Certificate

A green checkmark is shown next to the common name field. A 'View' button is at the bottom right of the certificate detail box.

Débogueur IBM i – Mise en oeuvre

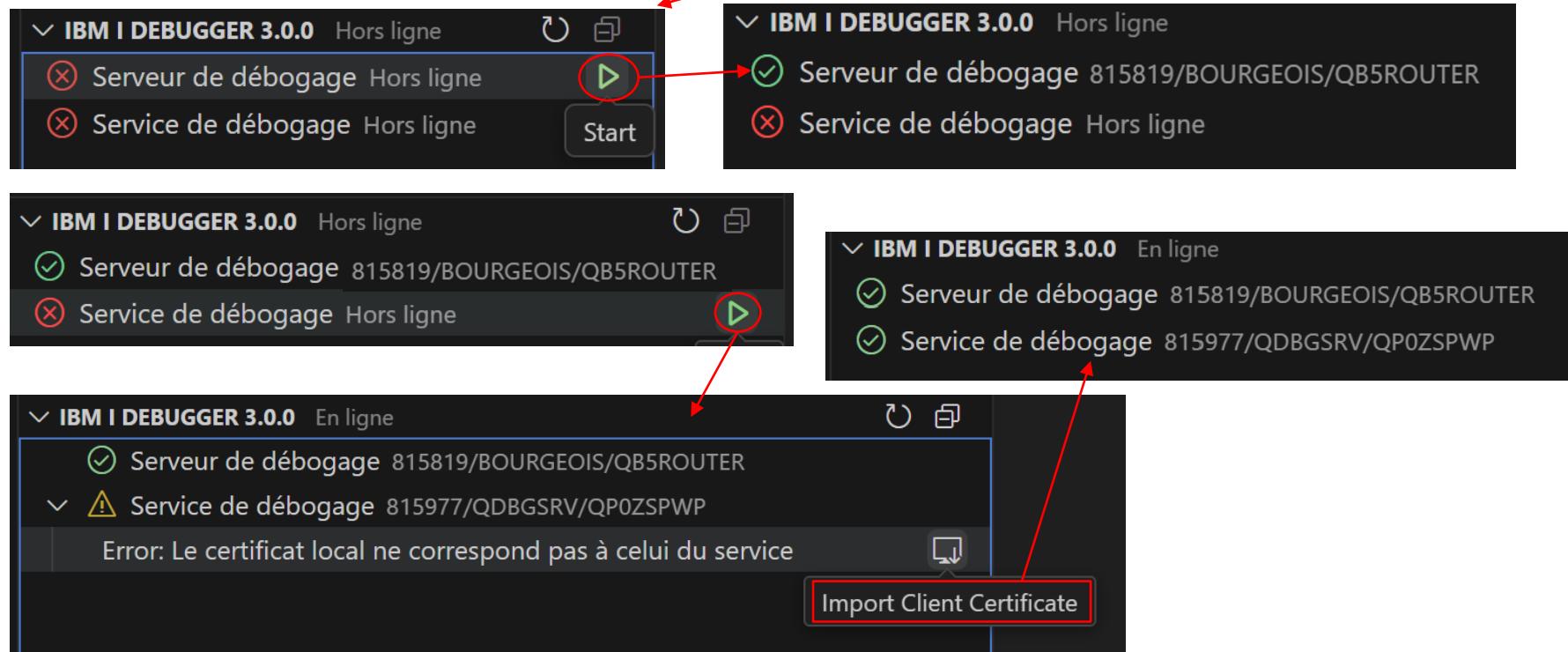
- Côté IBM i (2/2)
 - Le certificat doit être affecté au Debug Service

- Moniteurs
- Mon travail
- Réseau
- Sécurité
- Utilisateurs et groupes
- Performances
- Système de fichiers



Débogueur IBM i – Mise en oeuvre

■ Côté VS Code



Exemple de débogage IBM i

■ Les programmes

≡ PGM1.CLLE X

PB > QCLLESRC > ≡ PGM1.CLLE > ...

```
1 | | | PGM      PARM(&ANNEE)
2 | | | DCL      VAR(&ANNEE) TYPE(*DEC) LEN(4 0)
3 | | | SBMJOB  CMD(CALL PGM(PGM2) PARM((&ANNEE (*DEC 4 0))))
4 | | | ENDPGM
```

CALL PGM(PGM1)

PARM((2005 (*DEC 4 0)))

```
DSPLY  Année : 2005
DSPLY  D21 AZOULAY
DSPLY  D11 COLAS
DSPLY  D31 DURAND
DSPLY  D11 MARTIN
DSPLY  D11 COLAS
DSPLY  D11 MARTIN
DSPLY  D21 AZOULAY
DSPLY  D31 DURAND
```

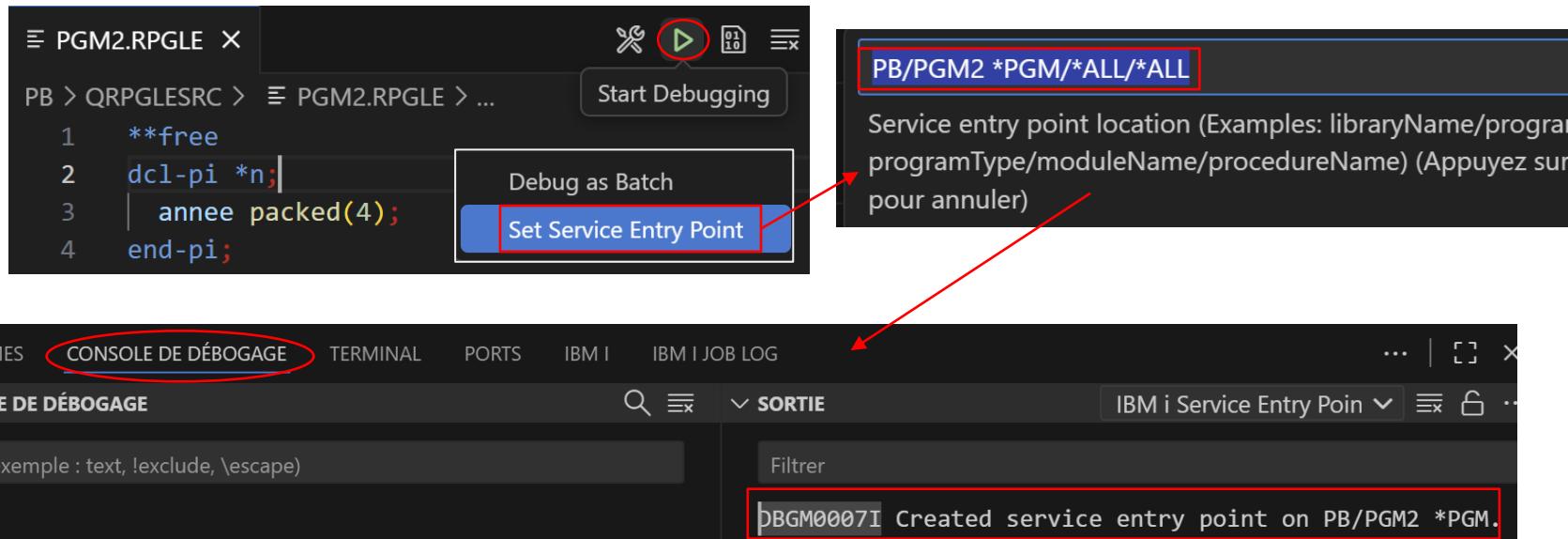
≡ PGM2.RPGLE X

PB > QRPGLESRC > ≡ PGM2.RPGLE > ...

```
1  **free
2  dcl-pi *n;
3  |  annee packed(4);
4  end-pi;
5
6  dcl-s cpt int(3);
7
8  dcl-ds emp dim(4) qualified;
9  |  nomemp varchar(20);
10 |  numsrv char(3);
11 end-ds;
12
13 dsply ('Année : ' + %char(annee));
14
15 emp(1).nomemp = 'MARTIN';
.
.
.
28 for cpt = 1 to %elem(emp);
29 |  dsply (emp(cpt).numsrv + ' ' + emp(cpt).nomemp);
30 endfor;
31
32 sorta(a) emp %fields(numsrv:nomemp); // Tri par numéro de
33
34 for cpt = 1 to %elem(emp);
35 |  dsply (emp(cpt).numsrv + ' ' + emp(cpt).nomemp);
36 endfor;
37
38 *inlr = *on;
```

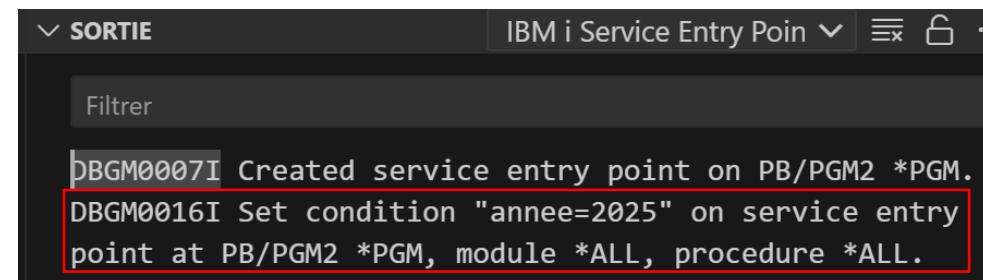
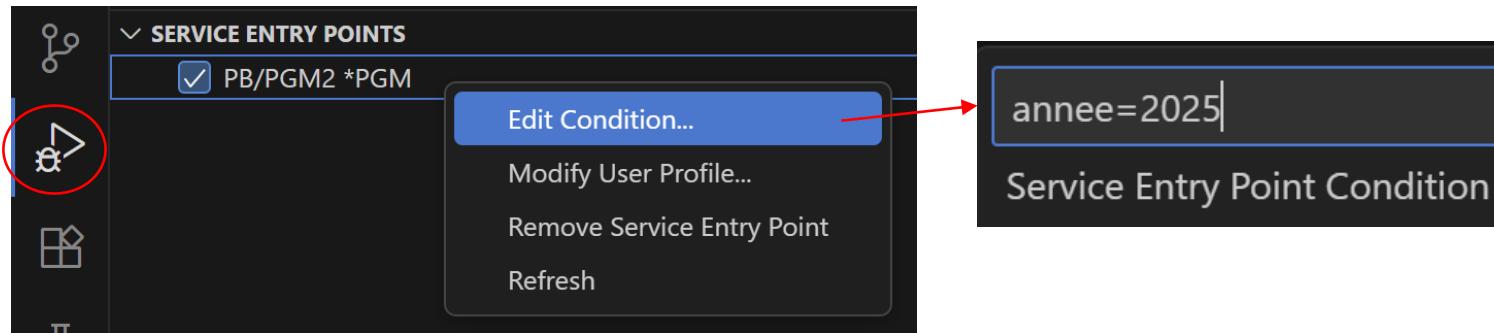
Exemple de débogage IBM i

- On veut déboguer PGM2 et uniquement si le paramètre ANNEE est égal à 2025 :
 - On va ajouter un point d'entrée de service sur PGM2
 - Puis ajouter une condition sur le point d'entrée de service



Exemple de débogage IBM i

- On veut déboguer PGM2 et uniquement si le paramètre ANNEE est égal à 2025 :
 - On va ajouter un point d'entrée de service sur PGM2
 - Puis ajouter une condition sur le point d'entrée de service

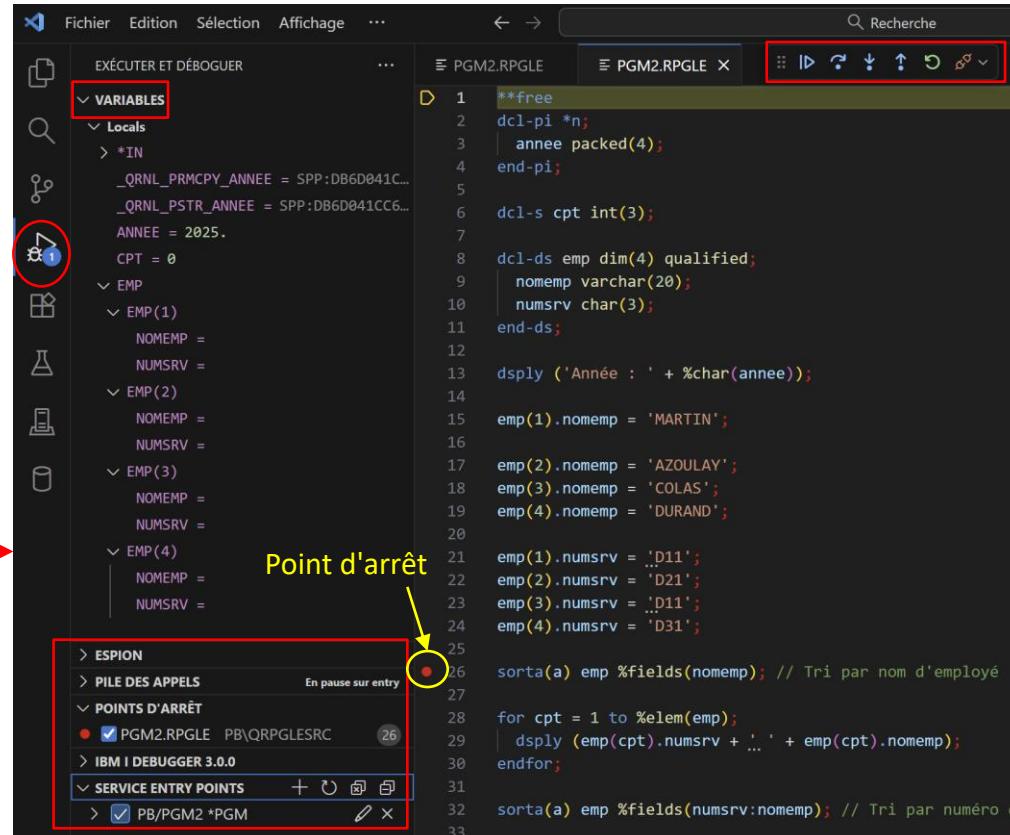


Exemple de débogage IBM i

- Exécution du programme PGM1 avec ANNEE = 2025
 - PGM1 fait un SBMJOB du CALL de PGM2
 - La session de débogage démarre pour PGM2

CALL PGM(PGM1)

PARM((2025 (*DEC 4 0)))



Débogueur IBM i – Fonctionnalités

Démarrer / reprendre l'exécution du programme (F5)

Pas à pas – STEP OVER (F10)

Pas à pas – STEP INTO (F11)



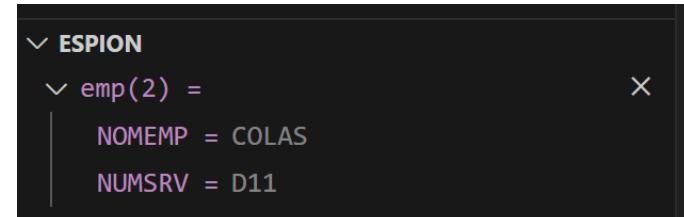
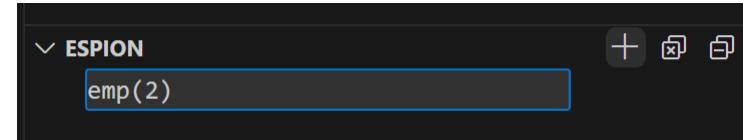
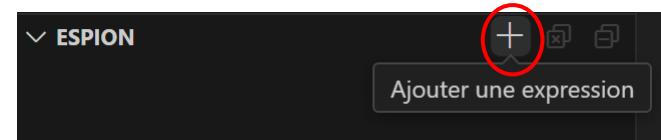
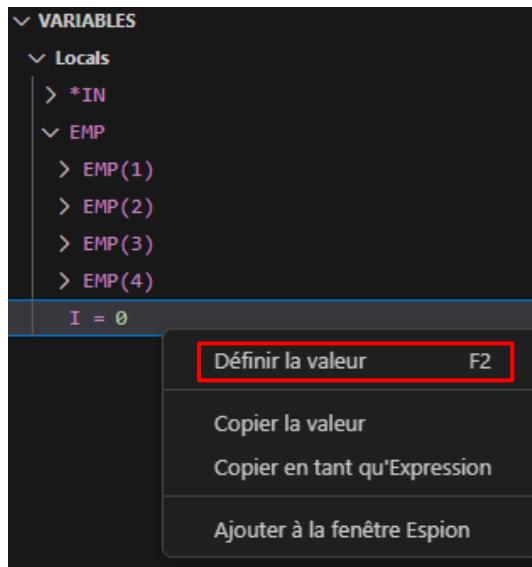
Pas à pas – STEP OUT (MAJ+F11)

Arrêter le débogage (MAJ+F5)

Redémarrer (CTRL+MAJ+F5)

Débogueur IBM i – Fonctionnalités

- Modification de variables
- Monitoring de variables



3. Les tests unitaires IBM i avec VS Code



Tests unitaires

- Tests unitaires
 - Tests d'unités de code isolées :
 - Logique métier ou technique (règles métier, calculs, formatage de données...)
 - Sur l'IBM i :
 - Test de procédures ou de programmes "métier" (non 5250)
 - Pas de test de programmes complets, qu'ils soient interactifs ou batch
 - Pour tester ce type de programme il existe des solutions comme Arcad Verifier (Arcad Software), ReplicTest (Polverini & Partners), X-DataTest (Fresche Solutions), etc.
 - On va développer (ou faire générer si VS Code) des procédures de test qui vont tester les procédures métier
 - Couverture de code
 - Donne le pourcentage du code qui a été exécuté dans la procédure de test
 - Permet de connaître l'efficacité des procédures de test
 - Plus le % est élevé, plus la procédure de test est efficace

Remarque : couverture de code de programmes IBM i

- L'extension "IBM i Code Coverage" pour VS Code, qui permet d'obtenir le taux de couverture de code d'un programme, n'est actuellement pas disponible

This repository was archived by the owner on Nov 19, 2023. It is now read-only.

 [codefori / code-coverage-ibmi](#) Public archive

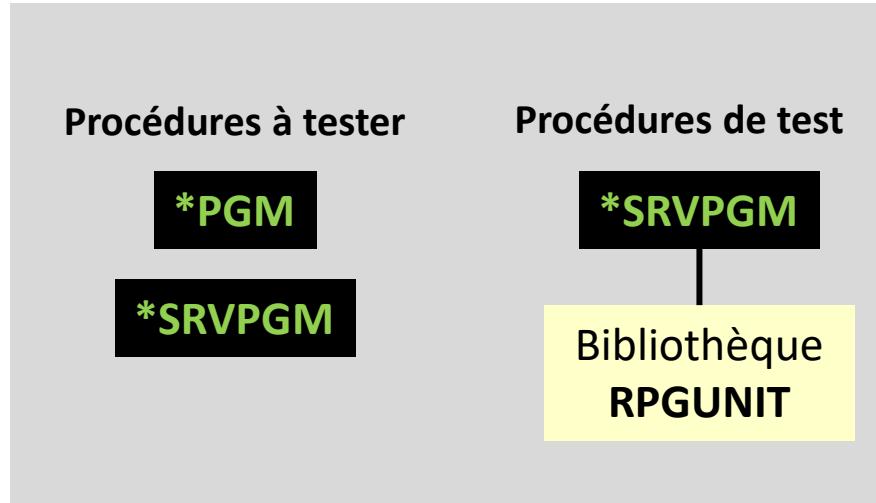
- La fonction de couverture de code proposée dans les tests unitaires permet de visualiser le taux de couverture **de la procédure de test, pas de la procédure testée**

Tests unitaires IBM i avec VS Code – Fonctionnalités

- L'extension **IBM i Testing** dans VS Code va permettre :
 - De générer automatiquement des **stubs** : génération des procédures de test
 - De **configurer** les tests (paramètres de compilation et d'exécution)
 - D'**exécuter** un test individuel ou une suite de tests
 - De **visualiser** les résultats des tests
 - De générer le résultat de la **couverture de code**, aux niveaux procédure et ligne
 - D'**automatiser** les tests pour une intégration dans un pipeline **CI/CD**

Tests unitaires IBM i avec VS Code – Composants et prérequis

IBM i



- Installer la bibliothèque **RPGUNIT**

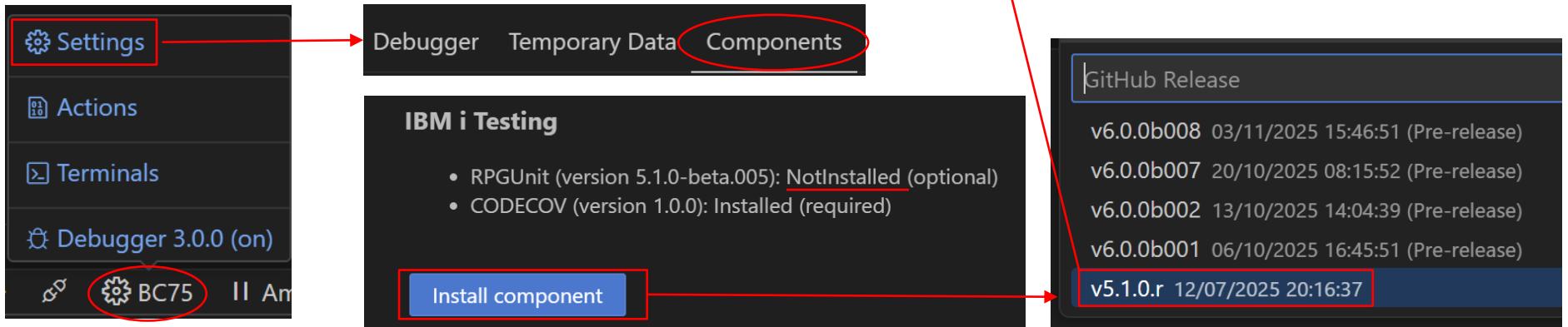
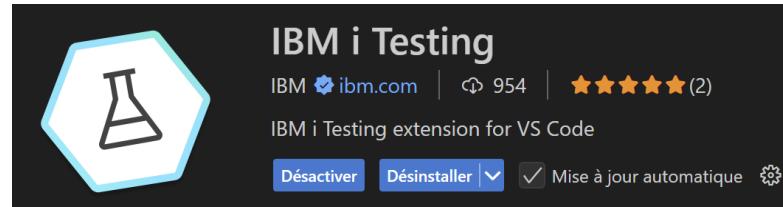


VS Code

- Installer l'extension **IBM i Testing**
- Ajouter les bibliothèques RPGUNIT et QDEVTOOLS dans la LIBL
- Si besoin modifier le fichier .bashrc

Tests unitaires IBM i avec VS Code – Composants et prérequis

- Dans VS Code
 - Extension IBM i Testing
- Sur l'IBM i
 - Bibliothèque RPGUNIT
 - Peut être installée depuis VS Code



2025-11-12 14:41:01.541 [info] RPGUnit v5.1.0.r installed successfully into RPGUNIT.LIB

IBM i Testing

- RPGUnit (version 5.1.0-beta.005): NotInstalled (optional)
- CODECOV (version 1.0.0): Installed (required)

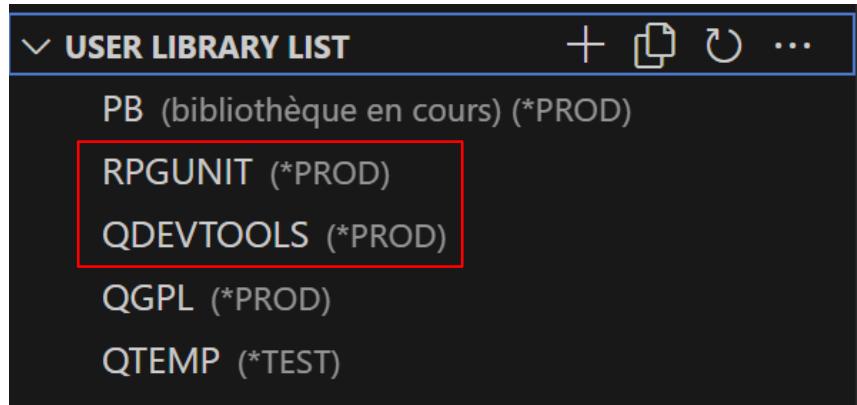
Install component

GitHub Release

Version	Released
v6.0.0b008	03/11/2025 15:46:51 (Pre-release)
v6.0.0b007	20/10/2025 08:15:52 (Pre-release)
v6.0.0b002	13/10/2025 14:04:39 (Pre-release)
v6.0.0b001	06/10/2025 16:45:51 (Pre-release)
v5.1.0.r	12/07/2025 20:16:37

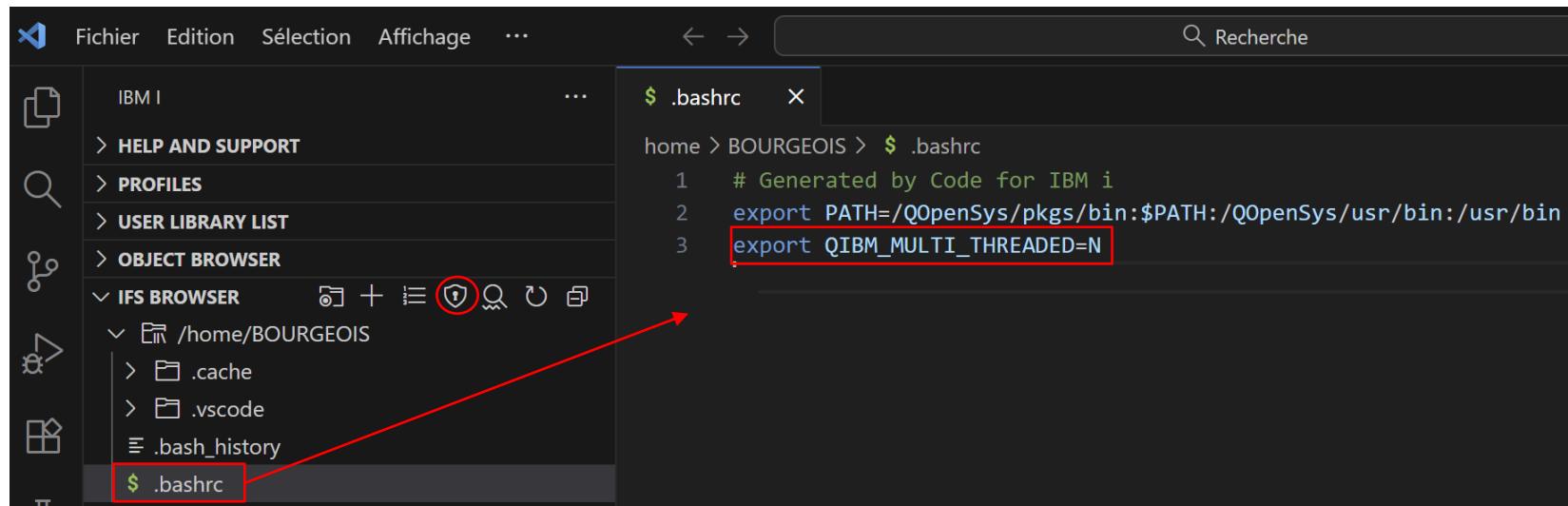
Tests unitaires IBM i avec VS Code – Composants et prérequis

- Ajouter les bibliothèques **RPGUNIT** et **QDEVTOOLS** dans la **LIBL**



Tests unitaires IBM i avec VS Code – Composants et prérequis

- **Optionnel** : si le message d'erreur "Commande STRDBDG non admise dans les travaux admettant des unités d'exécution" apparaît lors de l'exécution des tests (plus précisément la couverture de code), il faut modifier le fichier `.bashrc`



The screenshot shows the VS Code interface with the following details:

- Left Sidebar (IFS Browser):** Shows the file structure: `home > BOURGEOIS > .bashrc`. The `.bashrc` file is highlighted with a red box.
- Code Editor:** The file `$.bashrc` is open. The content is as follows:

```
1 # Generated by Code for IBM i
2 export PATH=/QOpenSys/pkgs/bin:$PATH:/QOpenSys/usr/bin:/usr/bin
3 export QIBM_MULTI_THREADED=N
```

The line `3 export QIBM_MULTI_THREADED=N` is also highlighted with a red box.
- Status Bar:** Shows the status `home > BOURGEOIS > $.bashrc`.
- Icons:** The IFS Browser icon in the sidebar has a red circle with a question mark over it, indicating a potential issue.

RPGUnit + extension IBM i Testing – Vocabulaire

- **Stub**
 - La source qui contient la ou les procédure(s) de test
- **Test Case**
 - Cas de test individuel
 - Une procédure de test, dont le nom commence obligatoirement par **test**, permettant de tester une procédure ou un programme métier
 - Par exemple si la procédure à tester s'appelle **isPalindrome**, la procédure de test pourra s'appeler **testisPalindrome**, **test_isPalindrome**, etc.
- **Test Suite**
 - Suite entière de tests
 - Ensemble de cas de tests

RPGUnit + extension IBM i Testing – Stubs

- Les procédures de test (cas et suites de test) peuvent être :
 - Développées **manuellement**
 - En s'appuyant sur l'API Reference de RPGUnit
 - <https://codefori.github.io/docs/developing/testing/writing/>

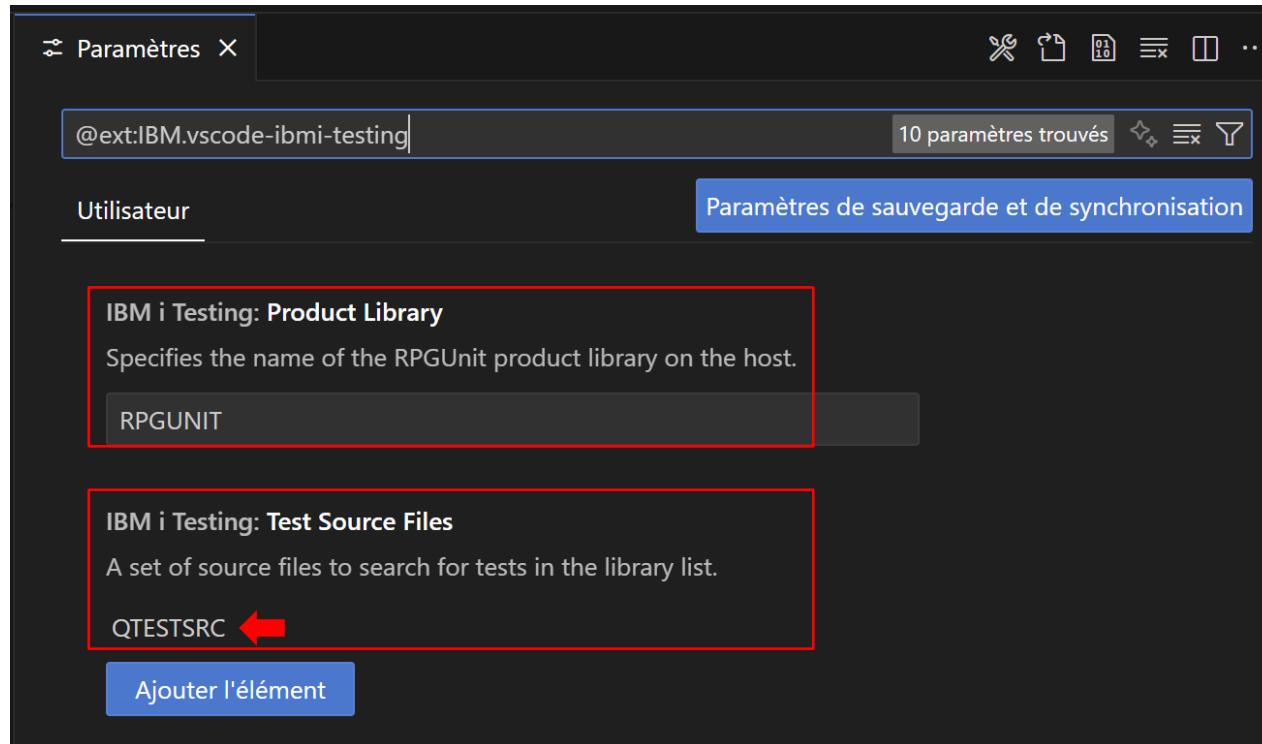
API Reference

In order to use the APIs provided by RPGUnit, you need to add the `TESTCASE` copybook to your test suite:

```
/include qinclude,TESTCASE
```

- Générées **automatiquement** à partir de VS Code
 - Création de **stubs** à partir des procédures à tester
 - Modification possible en s'appuyant sur l'API Reference de RPGUnit

Extension "IBM i Testing" de VS Code – Paramètres



Extension "IBM i Testing" de VS Code – Paramètres

IBM i Testing: Test Stub Preferences

Preferences for how test stubs should be generated:

- **Show Test Stub Preview**: Controls whether to show a preview of the test stub before adding it to the file or source member. This refactoring preview can also be used to selectively insert portions of the stub.
- **Test Source File**: The default source file to generate new test members into.
- **Test Source Directory**: The default directory to generate new test files into.
- **Prompt For Test Name**: Controls whether to prompt for the name of the test including where it is located. The default source file is set using the **Test Source File** preference, the default directory is set using the **Test Source Directory** preference, and the test file or member will be named according to the rules described [here](#).
- **Add Control Options and Directives**: Controls whether to add control options (`ctl-opt nomain ccsidcvt(*excp) ccsid(*char : *jobrun);`) and directives (`**free`) for new test files or members.
- **Add Includes**: Controls whether to generate relevant includes.
- **Add Prototypes**: Controls whether to generate a prototype for the procedure being tested (if it does not already exist).
- **Add Stub Comments**: Controls whether to add comments to the test stub to distinguish inputs, actual results, expected results, and assertions.

Élément	Valeur
Show Test Stub Preview	true
Test Source File	QTESTSRC
Test Source Directory	qtestsrd
Prompt For Test Name	false
Add Control Options and Directives	true
Add Includes	true
Add Prototypes	true
Add Stub Comments	false

Prompt For Test Name	true
----------------------	------

...

Exemple 1 – Génération d'un cas de test

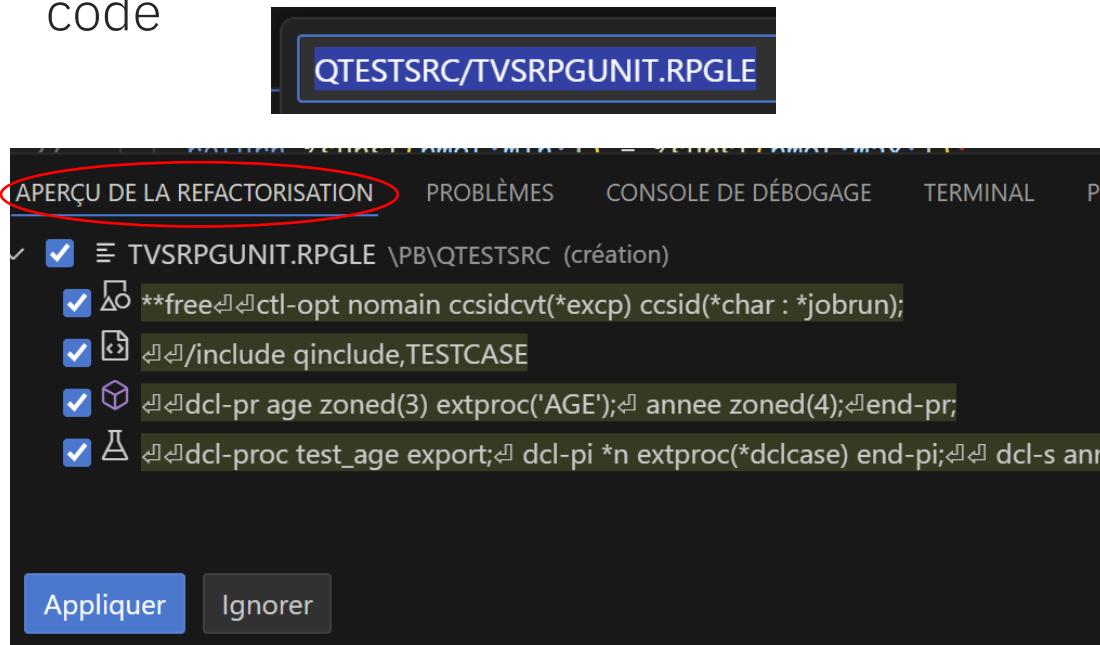
- On souhaite tester la procédure AGE qui se trouve dans le programme de service VSRPGUNIT
 - La procédure à tester doit être indiquée en EXPORT

The screenshot shows a Visual Studio Code editor window with the title bar "≡ VSRPGUNIT.RPGLE X". The code editor displays RPGLE code for a procedure named "age". A tooltip "Extraire" is open, showing two options: "Generate test case for 'age'" and "Generate test suite for 'VSRPGUNIT.RPGLE'". A red box highlights these options, and a red arrow points from the "Generate test case for 'age'" option to a confirmation dialog box. The dialog box, titled "Visual Studio Code", contains the message "The source file PB/QTESTSRC does not exist. Can it be created?" with "Yes", "No", and "Annuler" buttons. The code in the editor is as follows:

```
1  **free
2  ctl-opt nomain alwnull(*usrctl);
3
4  // Renvoi de l'âge
5  dcl-proc age export;
6  dcl-pi *n zoned(3);
7
8  Extraire
9
10 // Test si un mot est une p
11 dcl-proc isPalindrome export;
12
13 dcl-pi *n ind;
14 | pmot varchar(50) const;
15 | end-pi;
16
17 dcl-s min uns(3) inz(1);
```

Exemple 1 – Génération d'un cas de test

- Mode PREVIEW (aperçu de la factorisation) disponible avant de générer le code



QTESTSRC/TVSRPGUNIT.RPGLE

APERÇU DE LA REFACTORISATION PROBLÈMES CONSOLE DE DÉBOGAGE TERMINAL PC

✓ `≡ TVSRPGUNIT.RPGLE \PB\QTESTSRC (création)`

✓ `Δ free` `ctl-opt nomain ccsidcvt(*excp) ccsid(*char : *jobrun);`

✓ `Δ /include qinclude,TESTCASE`

✓ `Δ dcl-pr age zoned(3) extproc('AGE');` `annee zoned(4);` `end-pr;`

✓ `Δ dcl-proc test_age export;` `dcl-pi *n extproc(*dclcase) end-pi;` `dcl-s ann`

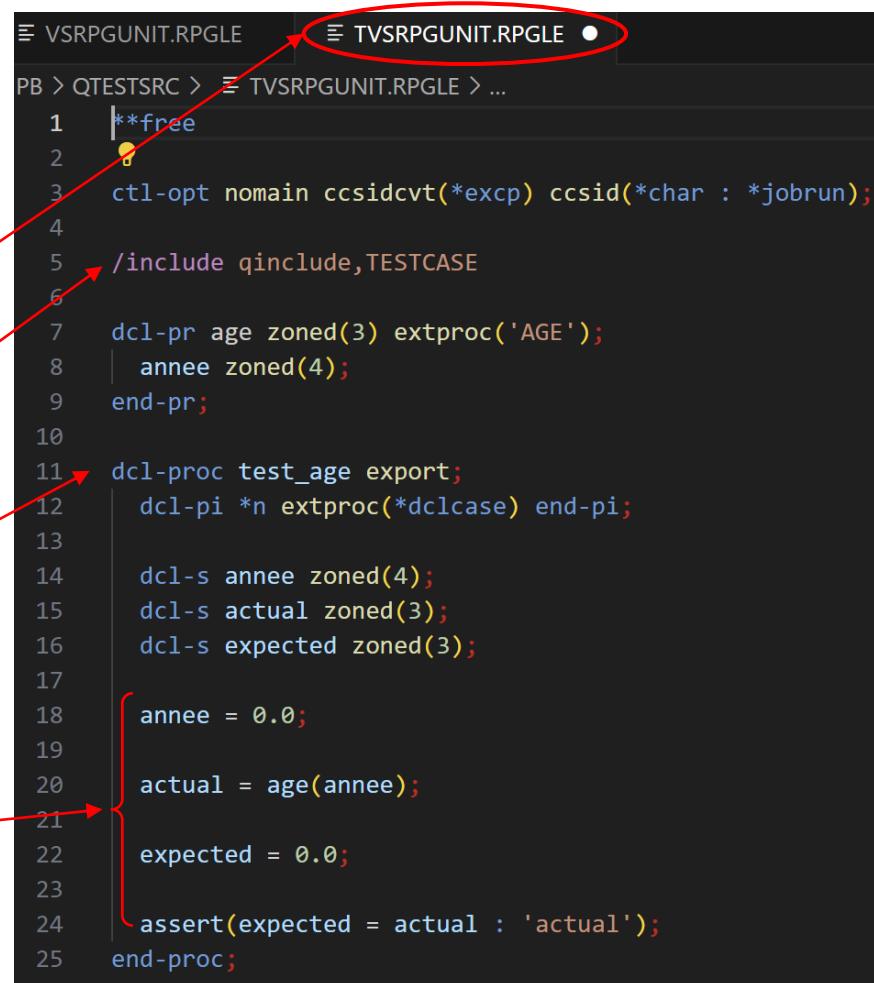
Appliquer Ignorer

Élément	Valeur
Show Test Stub Preview	true
Test Source File	QTESTSRC
Test Source Directory	qtestsrc
Prompt For Test Name	true
Add Control Options and Directives	true
Add Includes	true
Add Prototypes	true
Add Stub Comments	false

Exemple 1 – Génération d'un cas de test

■ Résultat :

- Génération d'un membre source de test intitulé **Txxx** (xxx : nom du membre d'origine)
- Ajout des directives **/include** nécessaires
- Ajout d'une procédure de test intitulée **test_yyy** (yyy : nom de la procédure à tester)
 - Avec variables en entrée, appel de la procédure à tester, définition de ce qui est attendu...



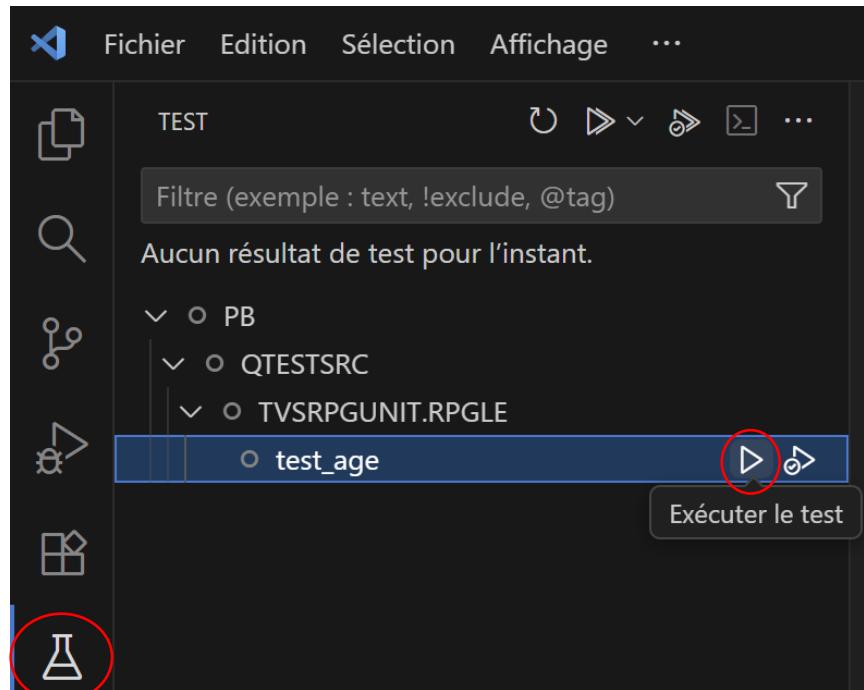
```
1  **free
2
3  ctl-opt nomain ccsidcvt(*excp) ccsid(*char : *jobrun);
4
5  /include qinclude,TESTCASE
6
7  dcl-pr age zoned(3) extproc('AGE');
8  | annee zoned(4);
9  end-pr;
10
11 dcl-proc test_age export;
12 | dcl-pi *n extproc(*dclcase) end-pi;
13
14 dcl-s annee zoned(4);
15 dcl-s actual zoned(3);
16 dcl-s expected zoned(3);
17
18 annee = 0.0;
19
20 actual = age(annee);
21
22 expected = 0.0;
23
24 assert(expected = actual : 'actual');
25 end-proc;
```

Exemple 1 – Exécution d'un cas de test

```
VS RPGUNIT.RPGLE    TVSRPGUNIT.RPGLE X

PB > QTESTSRC > TVSRPGUNIT.RPGLE > ...
1  **free
2
3  ctl-opt nomain ccsidcvt(*excp) ccsid(*char : *jobrun);
4
5  /include qinclude,TESTCASE
6
7  dcl-pr age zoned(3) extproc('AGE');
8  |    annee zoned(4);
9  end-pr;
10
11 dcl-proc test_age export;
12 |    dcl-pi *n extproc(*dclcase) end-pi;
13
14 dcl-s annee zoned(4);
15 dcl-s actual zoned(3);
16 dcl-s expected zoned(3);
17
18 annee = 1966;
19
20 actual = age(annee);
21
22 expected = 59;
23
24 assert(expected = actual : 'actual');
25 end-proc;
```

Détection automatique des stubs qui se trouvent dans les fichiers **QTESTSRC** de la **LIBL** et des procédures de test dont le nom commence par **test**



Exemple 1 – Exécution d'un cas de test

- Exécution des différentes étapes (creation du module, du programme de service...)

RÉSULTATS DES TESTS CONSOLE DE DÉBOGAGE IBM I JOB LOG TERMINAL PROBLÈMES 452 PORTS IBM I

```
: CRTSRVPGM SRVPGM(PB/TVSRPGUNIT) MODULE(PB/TVSRPGUNIT) EXPORT(*ALL) BNDSRVPGM(*LIBL/RUTESTCASE) ACTGRP(*CALLER) DETAIL('*BASIC') TGTRLS(*CURRENT)
CPD4090: Imprimante PRT01 non trouvée. File de sortie remplacée par QPRINT de QGPL.
CPD5D02: Définition introuvable pour le symbole 'AGE'.
CPF5D05: Programme de service TVSRPGUNIT non créé.
CPC2191: Objet TVSRPGUNIT de PB type *MODULE supprimé.
CPF9897: Unable to create test TVSRPGUNIT.
⚠ test_age
```

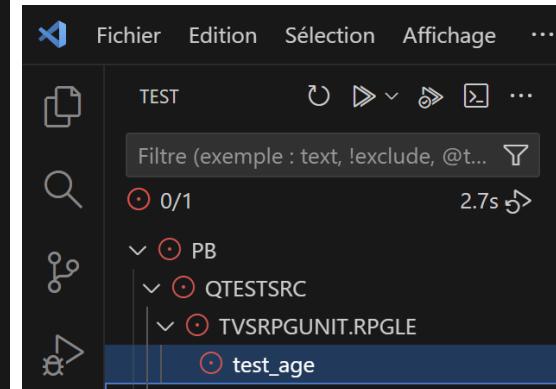
EXECUTION

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

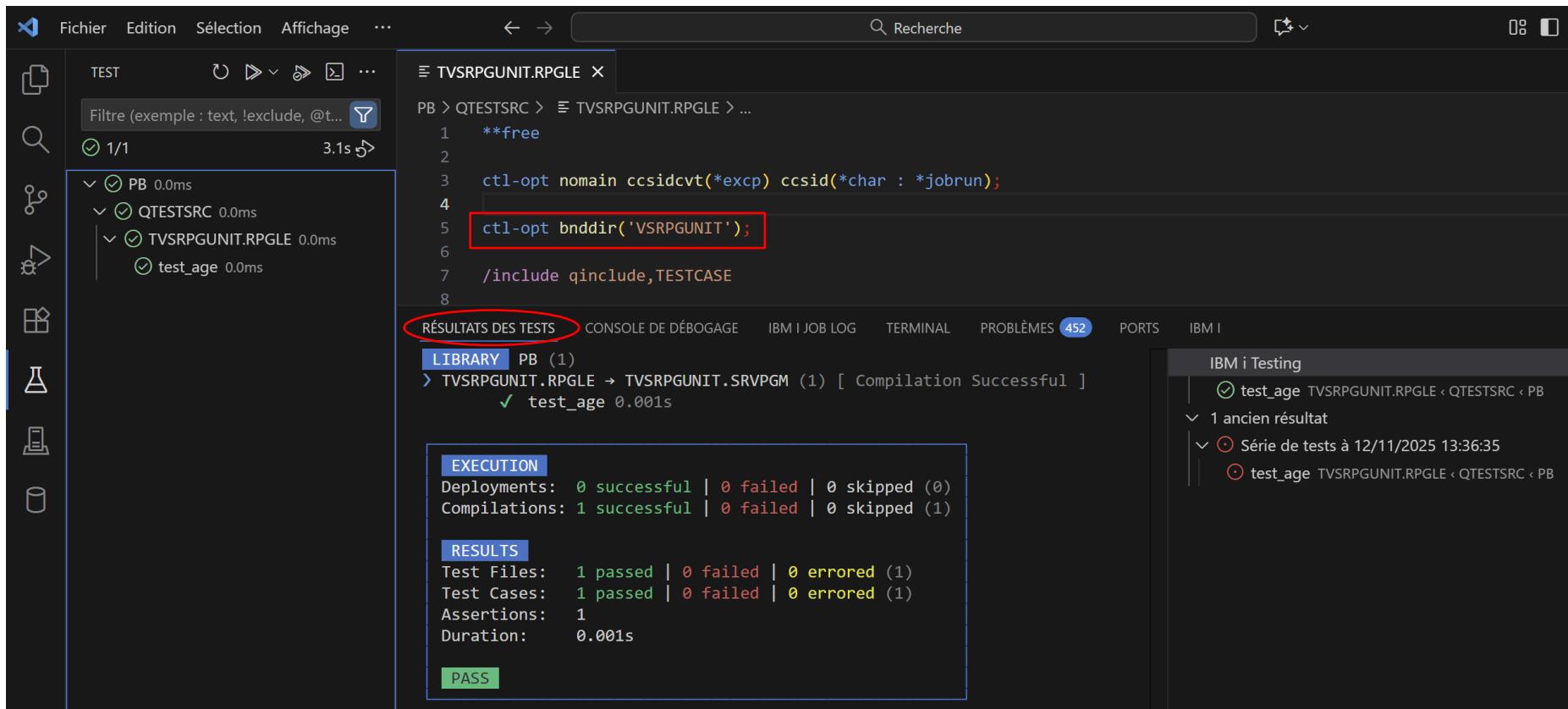
RESULTS

```
Test Files: 0 passed | 0 failed | 0 errored (0)
Test Cases: 0 passed | 0 failed | 1 errored (1)
Assertions: 0
Duration: 0s
```

ERROR



Exemple 1 – Exécution d'un cas de test



The screenshot shows the IBM i Testing interface within a code editor. The left sidebar displays a tree view of test cases: PB (0.0ms), QTESTSRC (0.0ms), TVSRPGUNIT.RPGLE (0.0ms), and test_age (0.0ms). The main editor area shows a RPGLE source code snippet with a line highlighted: `ctl-opt bnddir('VSRPGUNIT');`. The results section, circled in red, shows a successful compilation and execution of the `test_age` test case. The execution results are as follows:

```
LIBRARY PB (1)
TVSRPGUNIT.RPGLE → TVSRPGUNIT.SRVM (1) [ Compilation Successful ]
✓ test_age 0.001s

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

RESULTS
Test Files: 1 passed | 0 failed | 0 errored (1)
Test Cases: 1 passed | 0 failed | 0 errored (1)
Assertions: 1
Duration: 0.001s

PASS
```

The right sidebar shows a history of test results, with the most recent entry for `test_age` marked as successful.

Exemple 1 – Exécution d'un cas de test avec couverture de code

The screenshot shows the Rational Test Studio interface with two main windows. The left window is a 'TEST' view showing a test tree with a test named 'test_age' (0.0ms). The right window is a code editor showing a RPGLE source file named 'TVSRPGUNIT.RPGLE' with the following code:

```
1  PB > QTESTSRC
2
3  C
4  C
5
6  /
7
8  dcl-pr age zoned(3) extproc('AGE');
9
10 end-pr;
11
12 dcl-proc test_age export;
13 dcl-pi *n extproc(*dclcase) end-pi;
14
15 dcl-s annee zoned(4);
16 dcl-s actual zoned(3);
17 dcl-s expected zoned(3);
18
19 annee = 1966;
20
21 actual = age(annee);
22
23 expected = 59;
24
25 assert(expected = actual : 'actual');
26 end-proc;
```

The 'COUVERTURE DE TEST' panel at the bottom of the right window shows a coverage report for 'TVSRPGUNIT.RPGLE' with 100.00% coverage. The line numbers 19 through 26 are highlighted with a red box, indicating the code was executed during the test run.

Exemple 2 – Ajout d'une procédure à un cas de test

```
13 // Test si un mot est une palindrome
14 dcl-proc isPalindrome export;
15   dcl-pi *n ind;
16
17   Extraire
18   ⚡ Generate test case for 'isPalindrome'
19   ⚡ Generate test suite for 'VSRPGUNIT.RPGLE'
20
21   max = %len(pmot);
22   if max <> 0;
23     dow min < max and %subst(pmot:min) = %subst(pmot: max);
24       min += 1 ;
25       max -= 1 ;
26     enddo ;
27     return %subst(pmot:min:1) = %subst(pmot: max);
28   else;
29     | return 'Mot non fourni';
30   endif;
31 end-proc;
```

APERÇU DE LA REFACTURATION RÉSULTATS DES TESTS CON...

✓ TVSRPGUNIT.RPGLE \PB\QTESTSRC

✓ ⚡ oned(3) extproc('AGE'); ↴ ↴ dcl-pr isPalindrome ind

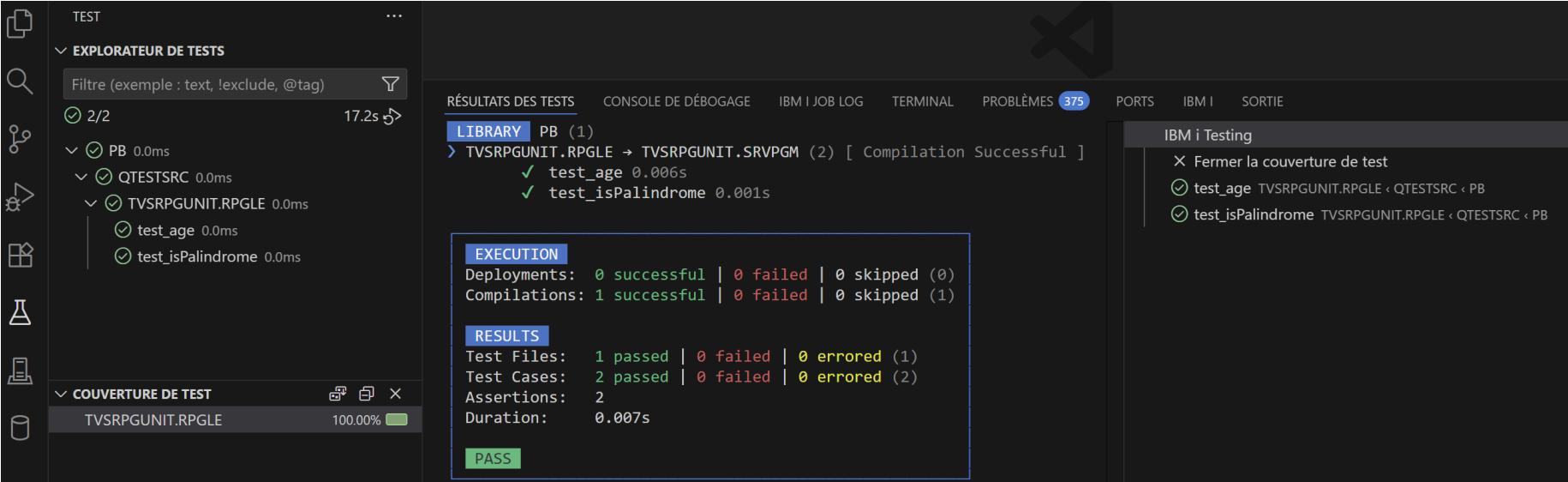
✓ ⚡ ↴ ↴ dcl-proc test_isPalindrome export; ↴ dcl-pi *n e

```
dcl-pr isPalindrome ind extproc('ISPALINDROME');
| pmot varchar(50) const;
end-pr;
```

```
dcl-proc test_isPalindrome export;
| dcl-pi *n extproc(*dclcase) end-pi;
| dcl-s pmot varchar(50);
| dcl-s actual ind inz(*on);
| dcl-s expected ind inz(*on);

pmot = 'RESSASSER';
actual = isPalindrome(pmot);
expected = *on;
nEqual(expected : actual : ...actual');
end-proc;
```

Exemple 2 – Ajout d'une procédure à un cas de test

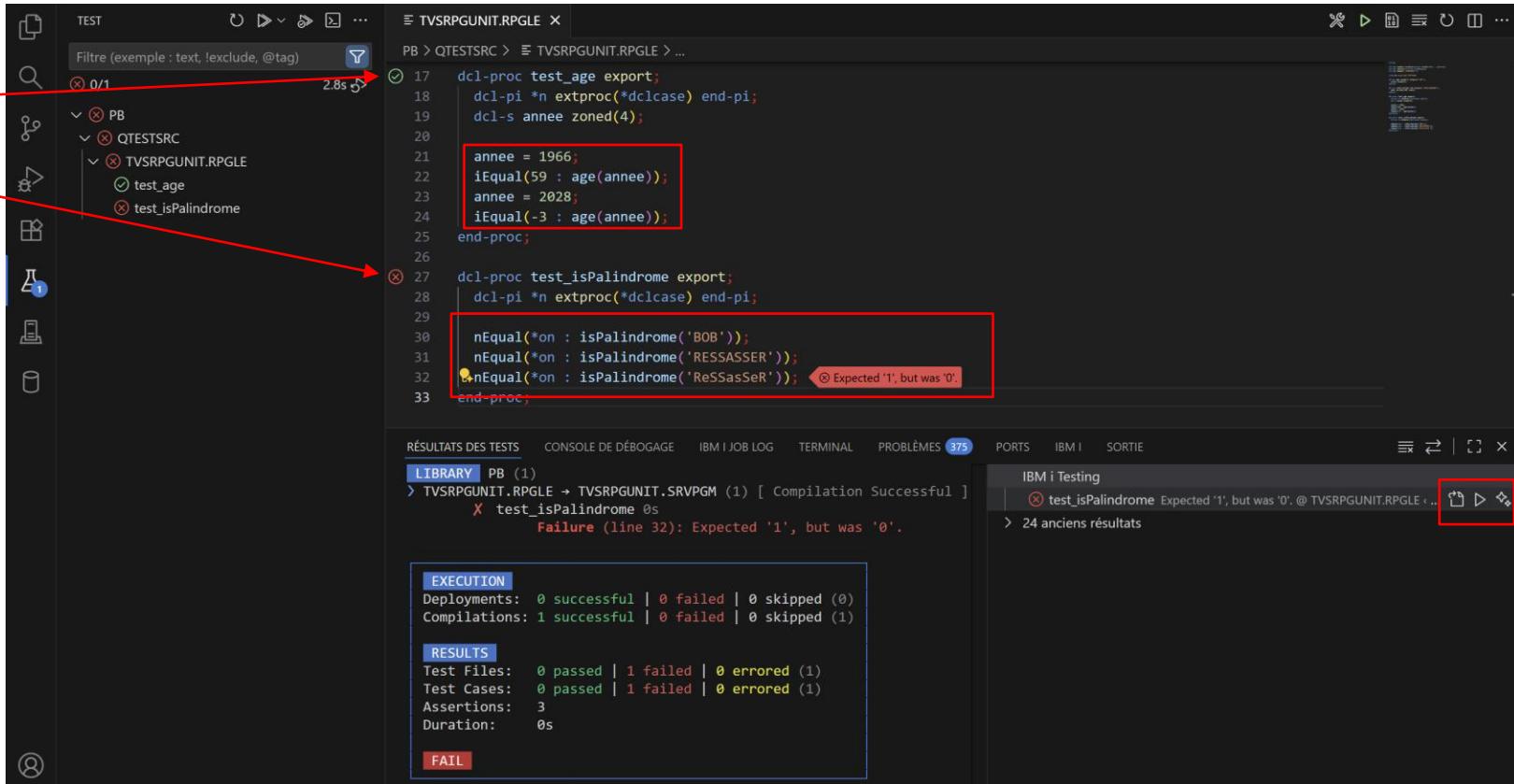


The screenshot shows the IBM i Test Studio interface with the following details:

- TEST** tab is selected.
- EXPLORATEUR DE TESTS** panel:
 - Filter: Filtre (exemple : text, !exclude, @tag)
 - 2/2 tests run in 17.2s.
 - Test cases:
 - ✓ PB 0.0ms
 - ✓ QTESTSRC 0.0ms
 - ✓ TVSRPGUNIT.RPGL 0.0ms
 - ✓ test_age 0.0ms
 - ✓ test_isPalindrome 0.0ms- RÉSULTATS DES TESTS** panel:
 - LIBRARY** PB (1)
 - TVSRPGUNIT.RPGL → TVSRPGUNIT.SRVPGM (2) [Compilation Successful]
 - ✓ test_age 0.006s
 - ✓ test_isPalindrome 0.001s
- EXECUTION** panel:
 - Deployments: 0 successful | 0 failed | 0 skipped (0)
 - Compilations: 1 successful | 0 failed | 0 skipped (1)
- RESULTS** panel:
 - Test Files: 1 passed | 0 failed | 0 errored (1)
 - Test Cases: 2 passed | 0 failed | 0 errored (2)
 - Assertions: 2
 - Duration: 0.007s
- PASS** button is visible.
- IBM i Testing** sidebar:
 - X Fermer la couverture de test
 - ✓ test_age TVSRPGUNIT.RPGL < QTESTSRC < PB
 - ✓ test_isPalindrome TVSRPGUNIT.RPGL < QTESTSRC < PB

Cas de tests – Autres possibilités

Exécution directe des tests



The screenshot shows the IBM i Test Studio interface with the following components:

- Left Panel (TEST View):** Shows a tree structure of test cases. A red arrow points from the text "Exécution directe des tests" to the tree. A red box highlights the status "0/1" and the duration "2.8s".
- Code Editor:** Displays RPGLE code for two test procedures: `test_age` and `test_isPalindrome`. A red box highlights the assertions in `test_isPalindrome`. The assertion at line 32 fails with the message "Failure (line 32): Expected '1', but was '0'." A red arrow points from this message to the "PROBLÈMES" tab.
- Bottom Tab Bar:** Includes tabs for RÉSULTATS DES TESTS, CONSOLE DE DÉBOGAGE, IBM i JOB LOG, TERMINAL, PROBLÈMES (375), PORTS, IBM i, and SORTIE.
- Bottom Left Panel (EXECUTION):** Shows deployment and compilation statistics: Deployments: 0 successful | 0 failed | 0 skipped (0); Compilations: 1 successful | 0 failed | 0 skipped (1).
- Bottom Middle Panel (RESULTS):** Shows test results: Test Files: 0 passed | 1 failed | 0 errored (1); Test Cases: 0 passed | 1 failed | 0 errored (1); Assertions: 3; Duration: 0s. The status is "FAIL".
- Bottom Right Panel (IBM i Testing):** Shows the failure message: "test_isPalindrome Expected '1', but was '0'." and "24 anciens résultats". A red box highlights the failure message, and a red arrow points from the "PROBLÈMES" tab to this panel.

RPGUnit – L'API Reference

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

Assertions

RPGUnit ≥ 5.1 RPGUnit ≤ 5.1

[assertEqual](#) [assertThat](#) [nEqual](#) [assert](#) [fail](#) [assertJobLogContains](#) [assertMessageQueueContains](#)

Utilities

Commonly Used APIs

[getFullTimeStamp](#) [waitSeconds](#) [getMonitoredMessage](#) [setLowMessageKey](#)

CL Command APIs

[CLRPFM](#) [RCLACTGRP](#) [runCmd](#)

Status Message APIs

[displayStatusMessage](#) [restoreStatusMessage](#) [clearStatusMessage](#)

Source Type APIs

[getMemberType](#) [getStmfType](#)

Exemple 3 – Génération d'une suite de tests

```
// Renvoi de l'âge
dcl-proc age export;
  dcl-pi *n zoned(3);
  ...
  
```

Extraire

- Generate test case for 'age'
- Generate test suite for 'VSRPGUNIT.RPGL' (highlighted with a red box)**

QTESTSRC/TESTVSRPGU.RPGL

APERÇU DE LA REFACTORISATION CONSOLE DE DÉBOGAGE IBM I JOB LOG

✓ TESTVSRPGU.RPGL \PB\QTESTSRC (création)

- ✓ `free` `ctl-opt nomain ccsidcvt(*excp) ccsid(*char : *jobrun);`
- ✓ `include qininclude,TESTCASE`
- ✓ `dcl-pr age zoned(3) extproc('AGE');` `annee zoned(4);` `end-pr;`
- ✓ `dcl-pr isPalindrome ind extproc('ISPALINDROME');` `pmot v`
- ✓ `dcl-pr statutFilm varchar(200) extproc('STATUTFILM');` `titre`
- ✓ `dcl-proc test_age export;` `dcl-pi *n extproc(*dclcase) end-pi;`
- ✓ `dcl-proc test_isPalindrome export;` `dcl-pi *n extproc(*dclcase)`
- ✓ `dcl-proc test_statutFilm export;` `dcl-pi *n extproc(*dclcase)`

Exemple 3 – Génération d'une suite de tests

```
PB > QTESTSRC >  TESTVSRPGU.RPGL > ...
1  **free
2  |
3  ctl-opt nomain ccsidcvt(*excp) ccsid(*char : *jobrun);
4  ctl-opt option(*srcstmt:*nodebugio) alwnull(*usrctl);
5  ctl-opt bnddir('VSRPGUNIT');
6
7  /include qinclude,TESTCASE
8
9  dcl-pr age zoned(3) extproc('AGE');
10 |  annee zoned(4);
11 end-pr;
12
13 dcl-pr isPalindrome ind extproc('ISPALINDROME');
14 |  pmot varchar(50) const;
15 end-pr;
16
17 dcl-pr statutFilm varchar(200) extproc('STATUTFILM');
18 |  titre varchar(100) options(*trim) const;
19 |  etat varchar(10) options(*nullind) const;
20 |  dateSortie varchar(20) options(*nopass:*convert) value;
21 end-pr;
```

```
23  dcl-proc test_age export;
24  |  dcl-pi *n extproc(*dclcase) end-pi;
25  |  dcl-s annee zoned(4);
26  |  dcl-s actual zoned(3);
27  |  dcl-s expected zoned(3);
28
29  |  annee = 2020;
30  |  actual = age(annee);
31  |  expected = 5;
32  |  assert(expected = actual : ...actual');
33 end-proc;
34
35  dcl-proc test_isPalindrome export;
36  |  dcl-pi *n extproc(*dclcase) end-pi;
37  |  dcl-s pmot varchar(50);
38  |  dcl-s actual ind inz(*on);
39  |  dcl-s expected ind inz(*on);
40
41  |  pmot = 'IBM';
42  |  actual = isPalindrome(pmot);
43  |  expected = *off;
44  |  nEqual(expected : actual : ...actual');
45 end-proc;
```

Exemple 3 – Génération d'une suite de tests

```
13
47  dcl-proc test_statutFilm export;
48      dcl-pi *n extproc(*dclcase) end-pi;
49
50      dcl-s titre varchar(100);
51      dcl-s etat varchar(10) nullind;
52      dcl-s dateSortie date;
53      dcl-s actual varchar(200);
54      dcl-s expected varchar(200);
55
56      titre = 'AVATAR 4';
57      etat = 'P00';
58      dateSortie = %date + %years(4) + %months(1) + %days(10);
59
60      actual = statutFilm(titre : etat : dateSortie);
61
62      expected = 'AVATAR 4 - Tournage non démarré - Date de sortie : 2029-12-22';
63
64      assert(expected = actual : 'actual');
65 end-proc;
```

Exemple 3 – Génération d'une suite de tests

The screenshot shows the IBM i Test Studio interface. On the left, the 'EXPLORATEUR DE TESTS' (Test Explorer) displays a tree of tests under 'TEST'. The 'COUVERTURE DE TEST' (Coverage) section shows 100.00% coverage for 'TESTVSRPGU.RPGL'.

The main area shows the 'RÉSULTATS DES TESTS' (Test Results) tab. It includes sections for 'LIBRARY', 'EXECUTION', 'RESULTS', and a summary box.

- LIBRARY:** PB (1)
 - TESTVSRPGU.RPGL → TESTVSRPGU.SRVPGM (3) [Compilation Skipped]
 - ✓ test_age 0.002s
 - ✓ test_isPalindrome 0.001s
 - ✓ test_statutFilm 0.004s
- EXECUTION:**

```
Deployments: 0 successful | 0 failed | 0 skipped (0)
Compilations: 0 successful | 0 failed | 1 skipped (1)
```
- RESULTS:**

```
Test Files: 1 passed | 0 failed | 0 errored (1)
Test Cases: 3 passed | 0 failed | 0 errored (3)
Assertions: 3
Duration: 0.007s
```
- Summary:** PASS

The 'IBM i Testing' sidebar on the right shows a history of results, including the current run and an old result.

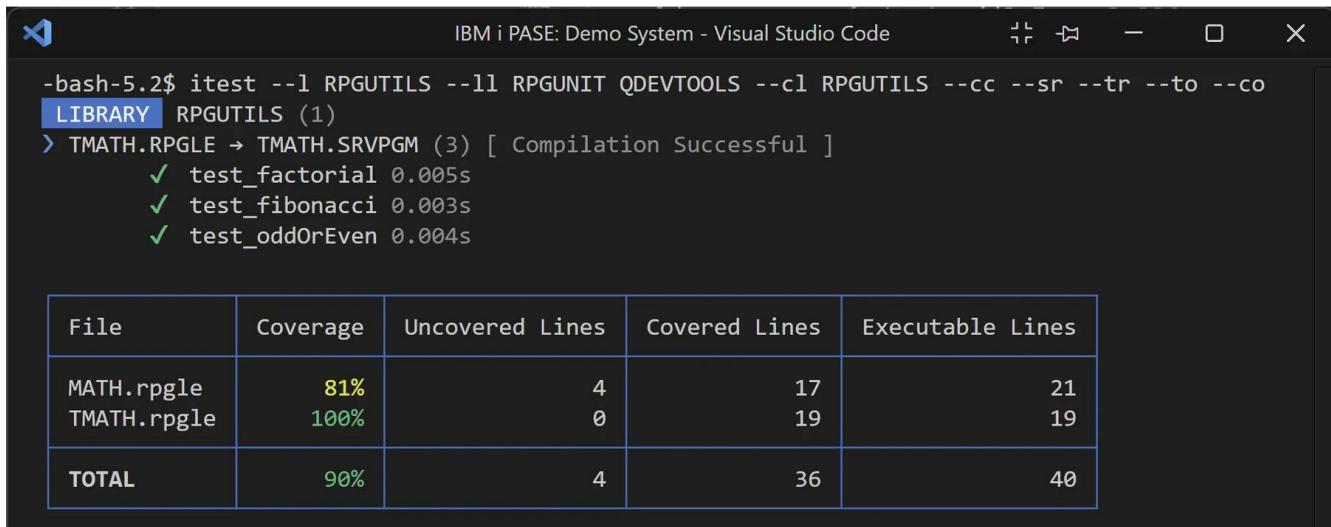
Tests automatisés et CLI

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

CLI & Automated Tests

version v1.2.3 downloads 63/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.



```
-bash-5.2$ itest --l RPGUTILS --ll RPGUNIT QDEVTOOLS --cl RPGUTILS --cc --sr --tr --to --co
LIBRARY RPGUTILS (1)
> TMATH.RPGLE → TMATH.SRVM (3) [ Compilation Successful ]
  ✓ test_factorial 0.005s
  ✓ test_fibonacci 0.003s
  ✓ test_oddOrEven 0.004s
```

File	Coverage	Uncovered Lines	Covered Lines	Executable Lines
MATH.rpgle	81%	4	17	21
TMATH.rpgle	100%	0	19	19
TOTAL	90%	4	36	40



4. Pour terminer

Pour vous former à VS Code : workshops IBM

- Actuellement disponibles : 2 workshops de 4 heures sur VS Code for IBM i :
 - Partie 1
 - Installation, configuration, personnalisation
 - Gestion des connexions IBM i, des filtres, raccourcis et profils
 - Edition de sources IBM i, compilation, actions et variables utilisateur
 - Autres fonctionnalités (comparaison, recherche, téléchargement, terminal, conversion RPG Free...)
 - Partie 2
 - Restructuration du code (refactoring)
 - Exécution, débogage de programmes et introduction aux tests unitaires
 - Autres fonctionnalités (linter : vérification de la qualité du code), recherche dans l'IFS
 - Gestion de DB2 for i (gestion des schémas DB2, exécution SQL, SELF, notebooks, IA...)
- En 2026 : un 3ème workshop de 4 heures
 - IBM i Project Explorer
 - Source Orbit
 - Intégration avec GIT
- Me contacter si vous êtes intéressé(e) : pbourgeois@fr.ibm.com

Pour vous aider dans votre projet de modernisation IBM i

- 2 offres de service IBM Expert Labs France – A partir de **2026** – **NOUVEAU**
 - 1. **IBM i Learning Connect** – Formation de vos équipes aux nouveautés IBM i
 - Formation aux technologies, outils, nouveautés
 - Aide à la prise en main des outils
 - Accompagnement et suivi des équipes
 - Prestation au forfait, renouvelable
 - 2. **IBM i Modernization Connect** – Aide à la modernisation des applications IBM i
 - Audit de l'existant
 - Aide à la définition du projet de modernisation
 - POC des outils de modernisation
 - Lancement du projet
 - Prestation au forfait, renouvelable
- Me contacter si vous êtes intéressé(e) : pbourgeois@fr.ibm.com

WE
RE
CI