
Rocket® Visual COBOL® (formerly a product of Micro Focus or formerly a product of Open Text) documentation that was created prior to Rocket Software's acquisition of certain products from OpenText may contain outdated references to "Micro Focus" or "OpenText", which are trademarks of OpenText. Rocket Software is not affiliated with Micro Focus or OpenText and has rebranded these products as Rocket® products.



Visual COBOL for Eclipse on Windows V10.0

Notices

Copyright

© 1996-2025 Rocket Software, Inc. or its affiliates. All Rights Reserved.

Trademarks

Rocket is a registered trademark of Rocket Software, Inc. For a list of Rocket registered trademarks go to: www.rocketsoftware.com/about/legal. All other products or services mentioned in this document may be covered by the trademarks, service marks, or product names of their respective owners.

Examples

This information might contain examples of data and reports. The examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

License agreement

This software and the associated documentation are proprietary and confidential to Rocket Software, Inc. or its affiliates, are furnished under license, and may be used and copied only in accordance with the terms of such license.

Note: This product may contain encryption technology. Many countries prohibit or restrict the use, import, or export of encryption technologies, and current use, import, and export regulations should be followed when exporting this product.

Corporate information

Rocket Software, Inc. develops enterprise infrastructure products in four key areas: storage, networks, and compliance; database servers and tools; business information and analytics; and application development, integration, and modernization.

Website: www.rocketsoftware.com

Rocket Global Headquarters

77 4th Avenue, Suite 100

Waltham, MA 02451-1468

USA

To contact Rocket Software by telephone for any reason, including obtaining pre-sales information and technical support, use one of the following telephone numbers.

Country and Toll-free telephone number

- United States: 1-855-577-4323
- Australia: 1-800-823-405
- Belgium: 0800-266-65
- Canada: 1-855-577-4323
- China: 400-120-9242
- France: 08-05-08-05-62
- Germany: 0800-180-0882
- Italy: 800-878-295
- Japan: 0800-170-5464
- Netherlands: 0-800-022-2961
- New Zealand: 0800-003210
- South Africa: 0-800-980-818
- United Kingdom: 0800-520-0439

Contacting Technical Support

The Rocket Community is the primary method of obtaining support. If you have current support and maintenance agreements with Rocket Software, you can access the Rocket Community and report a problem, download an update, or read answers to FAQs. To log in to the Rocket Community or to request a Rocket Community account, go to www.rocketsoftware.com/support. In addition to using the Rocket Community to obtain support, you can use one of the telephone numbers that are listed above or send an email to support@rocketsoftware.com.

Table of contents

Java accessing COBOL working-storage (separate projects)	6
--	---

Java accessing COBOL working-storage (separate projects)

The following example demonstrates a Java program from one project accessing a COBOL program's working-storage items from another project, and displaying them within Java and updating their values.

To use the Java and COBOL from within the same project, see *Java calling COBOL (COBOL/Java Interoperability project)*.


For demonstration purposes, this example creates two new projects, but this example can also be adapted to use existing native COBOL and Java projects.

1. Create the Java project:


- a. Click **File > New > Other**, and then select **Java Project** from the **Java** folder.

The **Create a Java Project** wizard is displayed.

- b. In the **Project name** field, enter **JShare**, select the required JRE for the project.

 Note: The bitism of the selected JRE must match that of the COBOL project you intend to create.

- c. In the **Module** section, ensure that the **Create module-info.java file** check box is not selected, and then click **Next**.
- d. On the **Libraries** tab, select **Classpath**, and then click **Add Library**.
- e. Double-click **COBOL JVM Runtime System**, and then click **Finish**.
- f. Click **Finish**.

If you are prompted to open the Java perspective, select **No**. The **JShare** project is created. To show both COBOL and Java projects, select the **COBOL Explorer** view, click  (**View menu**), and then click **Filters and Customization**. This opens the **Filters and Customization** dialog box. Click the **Pre-set filters** tab and uncheck **Non-Micro Focus projects**, and then click **OK**.

2. Create the Java program (**Demo2.java**):

- a. Select the **JShare** project in the **COBOL Explorer** view and click **File > New > Other > Class**, and then click **Next**.
- b. Ensure that the **Source folder** field specifies **JShare/src**, in the **Package** field enter **com.microfocus.java**, in the **Name** field enter **Demo2**, and then click **Finish**.

The program is opened in the editor.

- c. Replace the text with the following, and then save the program.

```
package com.microfocus.java;
import com.mycompany.demo2.strg;

public class Demo2
{
    public static void main(String[] args)
    {
        System.out.println("--COBOL items now accessible in Java--");
        int i1 = strg.demo2.grp1.i1.get();
        int i2 = strg.demo2.grp1.i2.get();
    }
}
```

```

        System.out.println("-- COBOL item i1 = " + i1);
        System.out.println("-- COBOL item i2 = " + i2);
        strg.demo2.grp1.i1.put(77777777);
        System.out.println("-- i1 updated from Java = " + strg.demo2.grp1.i1.get());
    }
}

```


If your workspace is set to build automatically, the program is compiled. If the workspace is not set to build automatically, on the **Project** menu, click **Build Project**. At this point, there will be errors because the COBOL program does not yet exist. These will be resolved during the following steps.

3. Create the native COBOL project:

- a. Click **File > New > COBOL Project**.

The **COBOL Project** wizard is displayed.

- b. In the **Project name** field, enter **CShare**, select a project template, and then click **Finish**.

 Note: The bitism of the selected project template must match that of the JRE selected in the **JShare** project.

The **CShare** project is created and displayed in the workspace.

4. Create the COBOL program (**demo2.cb1**):

- a. Select **CShare** in the COBOL Explorer view and click **File > New > COBOL Program**.
- b. In the **New file name** field, type **demo2.cb1**, and then click **Finish**.

The program is opened in the editor.

- c. Replace the text with the following, and then save the program.

```

$set sourceformat(variable)
program-id. demo2 as "demo2".
working-storage section.
>>JAVA-SHAREABLE ON
01 grp1.
   03 i1 pic 9(8) comp-5 value 88888888.
   03 i2 pic 9(8) comp-5 value 12345678.
>>JAVA-SHAREABLE OFF
01 grp2.
   03 p1 pic x.
   03 p2 pic 9.
procedure division.
display "start".
display "value of shared CBL grp is: "
display grp1::i1.
display grp1::i2.
end program demo2.

```

5. Set the COBOL project properties:

- a. Ensure that the **CShare** project is selected, then on the **Project** menu, click **Properties**.

The **Properties for CShare** dialog box appears.

- b. Select **Micro Focus > Build Configurations > Link**.

The **Link** settings are displayed.

- c. Change the following settings, and then click **Apply**:

Windows:

Option	Value
Target type	Single Native Library File

UNIX:

Option	Value
Output name	prefix the current value with ' lib '
Target type	Single Native Library File
Callable by non-COBOL applications	Yes
Multi-threaded	Yes

- d. Select **Micro Focus > Project Settings > COBOL**.

- e. Click  in the **Additional directives** field, and then add the following Compiler directives and click **OK**:

- java-output-path"<path-to-src-folder-of-java-project>"
- java-package-name"com.mycompany.demo2"

- f. Click **Apply and Close**.

If your workspace is set to build automatically, the program is compiled. If the workspace is not set to build automatically, on the **Project** menu, click **Build Project**.

6. In the **JShare** project, right-click the **src** folder and select **Refresh**.

The **demo2ws.java** file is displayed in a folder structure that represents the namespace.

7. Configure the **genjava** utility:

The **genjava** utility is required to produce some Java artifacts that interoperate with the COBOL program.

- a. On the **Run** menu, point to **External Tools**, and then select **External Tools Configurations**.

The **Create, manage, and run configurations** dialog box is displayed.

- b. Double-click **Program**.

A new configuration is displayed.

- c. In the **Name** field, type **genjava**.

- d. In the **Location** field, click **Browse File System** and select the full path and executable name for the **genjava** utility.

By default, the executable (**genjava**) is in the **bin** sub-folder of the product installation folder.

- e. In the **Working Directory** field, click **Browse Workspace** and select the **src** folder of the **JShare** project.
- f. In the **Arguments** field, enter the following arguments for the command:

```
<COBOL-Output-Name> -s demo2 -k com.mycompany.demo2
```

- g. Click **Run**.

Refresh the **src** folder in the **JShare** project again and a **strg.java** file has been added. At this point, the errors in the Java program are now resolved.

8. Create the run configuration, and run the application:

- a. Right-click **JShare** and select **Run As > Run Configurations**.

The **Run Configurations** dialog box is displayed.

- b. Double-click the **Java Application** launch configuration type in the left-hand pane.

A new configuration is displayed in the right-hand pane.

- c. In the **Name** field, enter a name for the configuration.
- d. In the **Main class** field, enter **com.microfocus.java.Demo2**.
- e. Click the **Arguments** tab, and in the **VM arguments** field, enter:

```
-Djava.library.path="<path-to-COBOL-project-output-folder>"
```

replacing *<path-to-COBOL-project-output-folder>* with the full path name to the **CShare** project's output folder.

- f. Click **Run**.

The following output is produced in the **Console** window:

```
--COBOL items now accessible in Java--  
-- COBOL item i1 = 88888888  
-- COBOL item i2 = 12345678  
-- i1 updated from Java = 77777777
```

The code and the output shows that the COBOL program has shared two PIC 9 COMP-5 items with the Java program. The Java program has used the `get` method to view the value of the COBOL data (to do this, the COMP-5 item was mapped to int types, as per *Mapping COBOL Items and Java Types*). The Java program then used the `put` method to change the value of the COBOL item (where again the mapping process was used, under the covers).

Related information

[Java accessing COBOL working-storage \(COBOL/Java Interoperability project\)](#)

[Java Accessing COBOL Working Storage Items](#)