

Visual Studio Code Extension for NTT DATA COBOL

User Guide

Table of Contents

Copyrights	3
Installation	4
Getting Started.....	6
SSH Settings.....	9
Public and Private SSH Keys	9
SSH_VCOBOL.....	9
Linux Bash Shell.....	10
Creating a Project Folder	11
Build and Compile Commands	15
Primer Example	15
Build Command.....	16
Problems View	17
Compile Command.....	18
Substitution Strings.....	18
Workspaces.....	20
Debugging	21
Debugging Stand-alone COBOL Programs	21
Debugging Online Programs	22
Debugging Batch Jobs	22
Debug Listener	24
VS Code Tasks	26
Task Definition Items	27
Task Variables	27
Task Icon	27

Other Task Features	28
Editor Features.....	28
Copybooks.....	28
Breadcrumbs	29
NTT DATA COBOL 6.5 or higher	29
Auto Complete	30
Files with no Extension	31
Additional Resources	32
Known Issues.....	33

Copyrights

Copyright (c) 2023 NTT DATA, Inc. All rights reserved.

Unpublished - rights reserved under the Copyright Laws of the United States.

REPRODUCTION, DISTRIBUTION OR DISPLAY OF ANY PORTION OF THIS PRODUCT IS PROHIBITED WITHOUT THE PRIOR EXPRESS WRITTEN PERMISSION OF NTT DATA, INC. NO DERIVATIVE WORKS MAY BE PREPARED.

Use of this software is only as permitted by the license grant and restrictions in the applicable software license terms. No other rights are conveyed.

Trademarks, logos and service marks mentioned in this product are registered and unregistered trademarks of NTT DATA, Inc. or other parties. All of these trademarks, logos and service marks are the property of their respective owners.

Introduction

The *VS Code Extension for NTT DATA COBOL* is an editor and debugger designed specifically for *NTT DATA COBOL* and the UniKix TPE/BPE environment. With the extension, you can create project folders along with the build and compile commands to manage your COBOL projects. The editor includes a number of features to assist you in editing your COBOL, BMS, and JCL source code.

The *VS Code Extension for NTT DATA COBOL* requires VS Code version 1.7 or higher. All versions of *NTT DATA COBOL* are supported.

The extension can be installed on Linux, Windows, and the Mac. A typical use case might be to install the extension on your Windows workstation and connect remotely to your Linux development environment. This procedure is documented in the Getting Started section.

Terminology

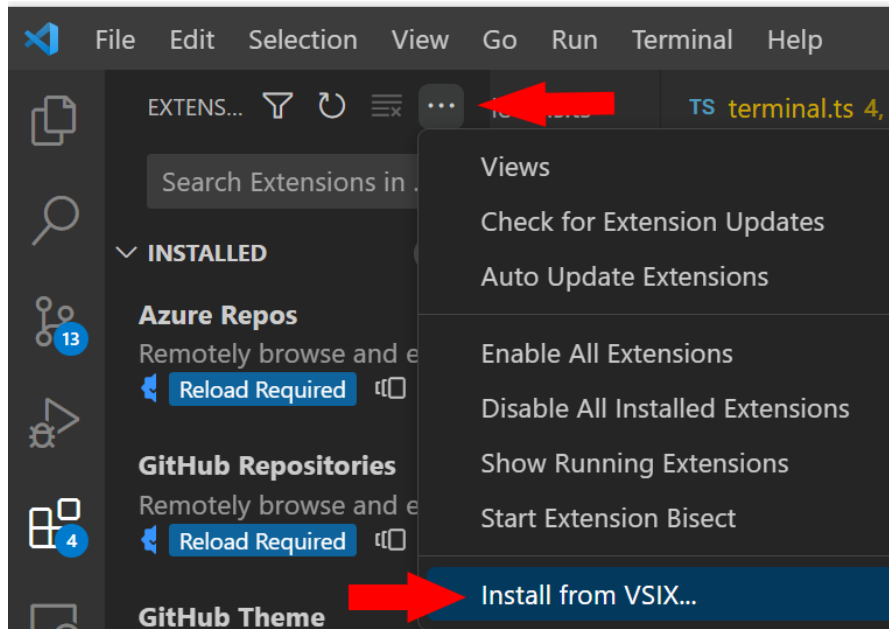
“VS Code”	Short name for Visual Studio Code
“extension”	Refers to the <i>VS Code Extension for NTT DATA COBOL</i> .
“workstation”	The Windows, Mac, or Linux system running VS Code.
“server”	The host running the UniKix software which will be a Linux based system.
“vcobol”	This is the internal name for the <i>NTT DATA Enterprise COBOL</i> software and the internal name of the extension. Using short names, without spaces, works best in extensions.

Installation

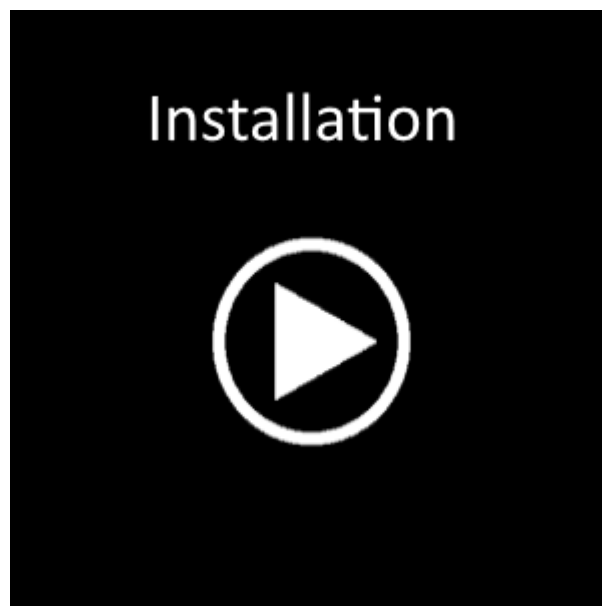
Download and install VS Code from the following link and run the installation program.

<https://code.visualstudio.com/download>

From VS Code, click on the Extensions icon in the Activity bar. In the Extensions toolbar (at the top of the view), click on the ellipsis and choose Install from VSIX.



Navigate to your download folder for the 'vscode-vcobol-###.vsix' file and press OK. You can uninstall the extension from the Extensions icon.

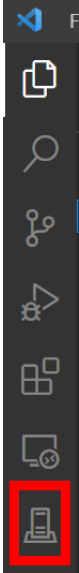


*Click the Play button above to watch the video.
Click the back button in your browser to return to this document.*

Getting Started

The *VS Code Extension for NTT DATA COBOL* requires an SSH2 connection to your server, it does not currently work with the local filesystem. The server can be running Windows, Linux, or Mac, however it must also be running an SSH2 compatible service.

To connect to a remote SSH server, click on the server icon in VS Code's Activity bar.



This will open a dialog where you can specify your connection details.

SSH Connection

Configuration Name

Host or IP Address

SSH Port

22

Username

Provide either the password or a private key - not both.

Password

Only provide a password if you want to update an existing one or set a new one.

Private Key

Only provide a private key if you want to update from the existing one or set one (OpenSSH format).

Choose File

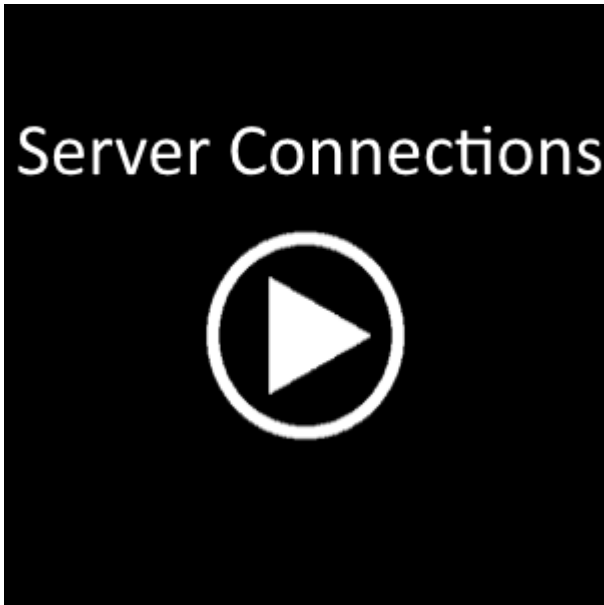
No file chosen

Initial Directory
(optional)

Save

Configuration Name	Enter a name for this connection.
Host of IP Address	Enter the address of the SSH server.
SSH Port	The SSH port number is typically 22.
Username	Enter the username for your login
Password	Enter the password for your login or leave blank if you wish to use a Private Key. Note: Passwords are stored in

	protected storage and cannot be viewed by others.
Private Key	Using private keys is often the preferred method for logging in due to its increased security. See the section for Private Keys below for more information.
Initial Directory	Enter an initial directory, such as your home directory, to navigate to when first connecting. It is recommended to use a directory that you have write permissions for.



Click the Play button above to watch the video.
Click the back button in your browser to return to this document.

SSH Settings

Public and Private SSH Keys

This section is only if you prefer to use a public/private key in lieu of a password.

You can generate SSH keys under windows using a number of tools including -- PowerShell, GIT-bash, and the Java JDK.

Execute the following commands:

```
ssh-keygen -t ed25519
ssh-copy-id -i .ssh/id_ed25519 <your username>@<your server name>
```

Note that the OpenSSH format is required and that the newer ed25519 is compatible with more systems compared to the older RSA format.

IMPORTANT: Your /etc/ssh/sshd_config file must include the following condition:

```
PubkeyAuthentication yes
```

Note: If you get an SSH error similar to 'Number too large', it means that your login profiles are too complex. Perhaps they include a bash 'read' command. You will need to simplify your login profile or create an alternate.

SSH_VCOBOL

If your shell profiles prompt you for information, such as which environment to set, or for other information, the prompt will interfere with the extension's execution of build, compile, debug and syntax checking operations. To detect if the extension is running your shell, a variable named SSH_VCOBOL is set in your environment.

In your profile script, you should add statements such as those listed below to check for this environment variable.

```
if [ -z "${SSH_VCOBOL}" ]
then
    # not executed by the vscode vcobol extension, run the prompt
    echo "PRESS ENTER"
    read abc
else
    # executed by vscode vcobol, so do a default option
    return
fi
```

IMPORTANT: Your system administrator must add the following line to your /etc/ssh/sshd_config file.

```
AcceptEnv SSH_VCOBOL
```

Linux Bash Shell

There are a plethora of Linux shells -- bash, fish, ksh, tcsh, zsh and many more. Unfortunately, these shells are not 100% compatible and the extension uses the shell for a lot -- building, compiling, debugging, executing JCL and COBOL, syntax checking and so on.

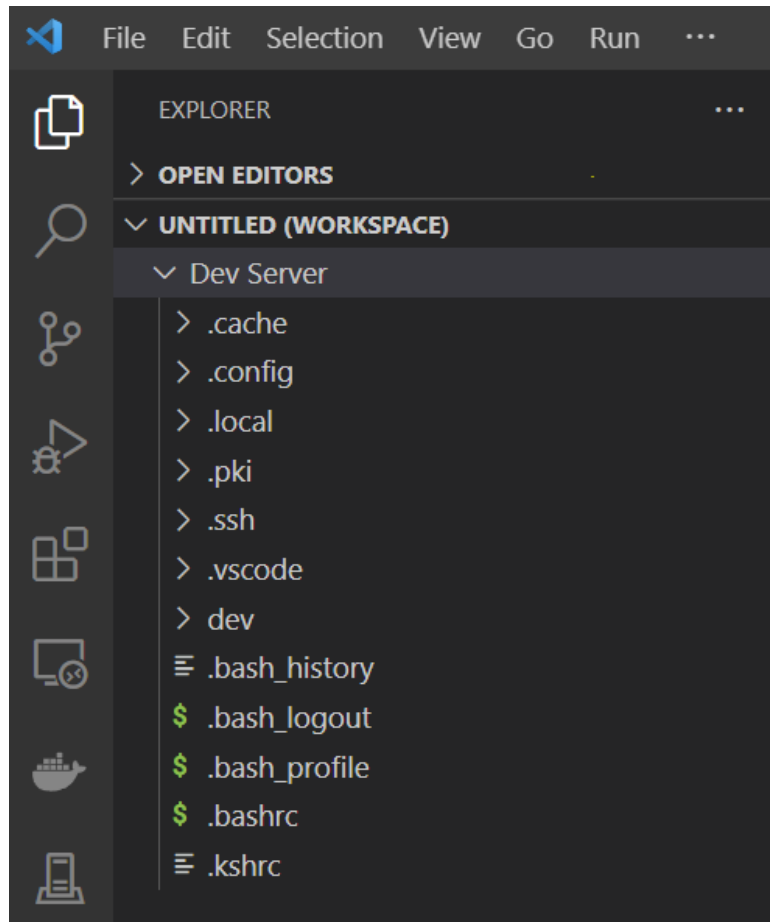
As you may suspect, the extension executes all the commands specified in your Project Folder settings and Tasks. And you probably don't want all these commands polluting your history file. So, great lengths have been taken to eliminate these commands from your history file and of the current release, the extension will try to launch a bash shell when executing commands since bash is the most flexible.

If you need support for a specific shell, please send feedback.

The Launch Terminal menu option will use your default shell.

Explorer View

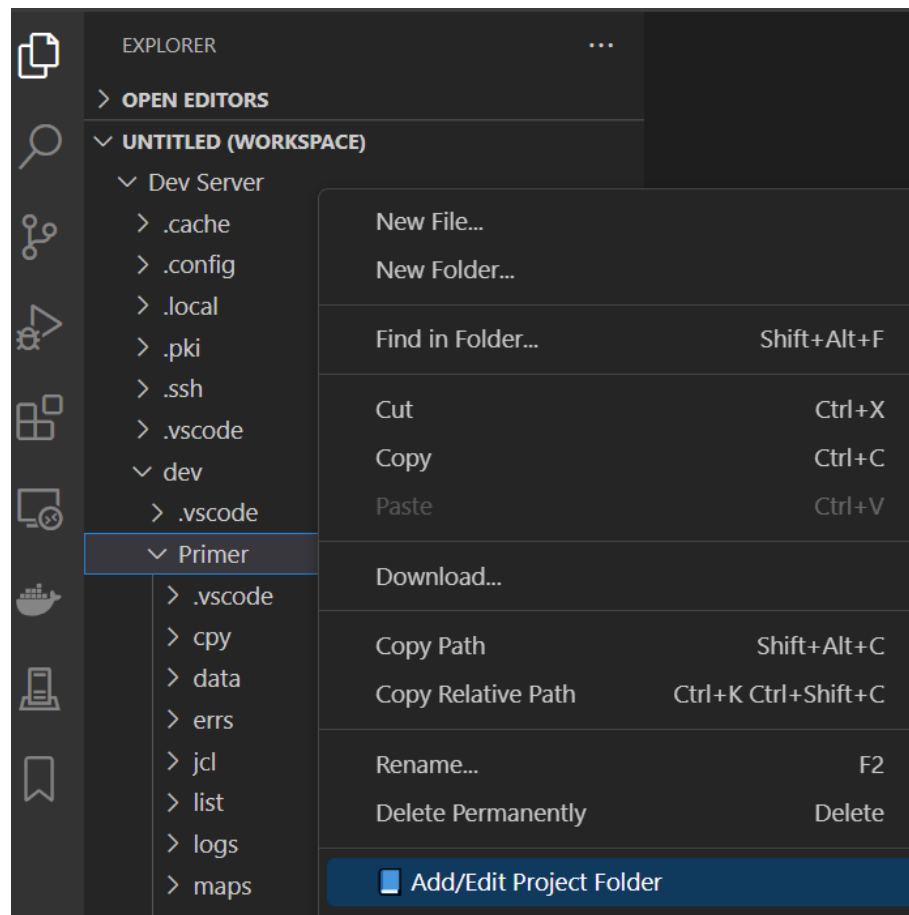
Once connected to your server, the VS Code Explorer pane will show the WORKSPACE view with your server connection. This WORKSPACE will likely be named UNTITLED as shown below. Expand the folder to view the files on your system.



Creating a Project Folder

Now that we have a connection to our server, we can start using VS Code for development. In this example, we want to create a Project Folder for the Primer COBOL example which included with the TPE installation.

To mark a folder as a Project folder, right click on the folder and choose [Add/Edit project folder](#).



This will open the Project Folder Editor dialog as shown below.

■ **Project Folder Configuration for:** /home/ms/dev

Project name:

Primer

All fields below are *optional*. Relative paths supported.

Build commands:

Compile commands:

BPESUB commands:

Use this setting to source your environment for build, compile, debug and program execution commands.

Source environment
commands:

source ~/.profile

Debug port:

9999

Debug listener port:

Copybook folders:

separate folders with spaces

Output Folder:

Source Folder:



Enable syntax check

Save

Apply

Cancel

Project Name	The name for the Project Folder. This is the only required field when creating a project folder. All other fields are optional .
Build Command	The command or commands to build the entire folder.

Compile Command	The command or commands to compile a single file.
BPESUB Command	<p>The command or commands to use when submitting a JCL file to BPESUB for debugging, running or translating. The extension will add the necessary command line switches to the BPESUB command.</p> <p>IMPORTANT: If you get errors when executing BPESUB commands which reference mail box, or message channel, you have likely sourced the wrong batchenv. Please recheck your settings on the Project Folder settings page.</p>
Source environment commands	<p>Enter the command(s) to use when sourcing your development environment. If this command is present, it will be executed prior to the following actions:</p> <p style="text-align: center;">Build, Compile, Debug and Run.</p>
Debug Port	The port number to use when debugging a COBOL program (when using vcrun -d). Typically, this port is 9999.
Debug Listener Port	The port number to use when debugging TPE/Online transactions via CEDF, or when using vcrun -o. This port number must be different from the Debug Port above.
Copybook folders	This setting is used for the COBOL Language Parser in the extension, specifically for use in Go to Definition, Find/Replace References. If you don't specify Copybook folders then the Parser may not be able to find your copybooks.
Output Folder	This setting is used when executing the Run Selected File option and is used to locate the compiled source file.
Source Folder	Depending upon how your COBOL programs are compiled, the NTT DATA COBOL debugger may, or may not, provide the full path to your source code. If when debugging you get an error message that the source file cannot be found you will need to specify the Source Folder for your source files.
Enable Syntax Check	If checked, and if Compile commands have been specified, the extension will perform syntax checking in the background to look for errors while you are making edits. Errors and warnings will appear in VS Code's Problems

	<p>View and as squiggly lines in the editor.</p> <p>If you are working with very large source files you may which to not enable this feature.</p> <p>NOTE: Previous versions of the extension would write the contents of the text editor to a temporary file under the /tmp folder and then perform the syntax check against that temporary file. This was possible because the extension could parse the Compile commands and modify them to work against the temporary file. However, if you use external commands, like make, this substitution is no longer possible and the only way to syntax check the file is to save it first. If auto saving is an issue for you, please uncheck this option.</p>
--	--

NOTE: The Copybook folders, Output Folder, and Source Folder settings are used by the extension’s internal editor, debugger, and program execution options. These settings are **not** used when building or compiling your source files.



The above settings are saved to a file named project.json in a .vscode subfolder of the project folder. This facilitates multi-user environments so that if another user checks out the project, they will automatically get the configuration for the project. You will want to be sure to include this file in your GIT repository.

Build and Compile Commands

You can enter any number of commands in the Build, Compile, and BPESUB options. Commands can be shell scripts, Unix commands, or anything else supported by your operating system. To illustrate an implementation of this, we will use the Primer example that is included with TPE installations.

Primer Example

This Primer example includes a `makefile` that can be used to build the sample COBOL files. You can execute the `makefile` by from the Linux console.

In the console, you can execute the following commands from the command line:

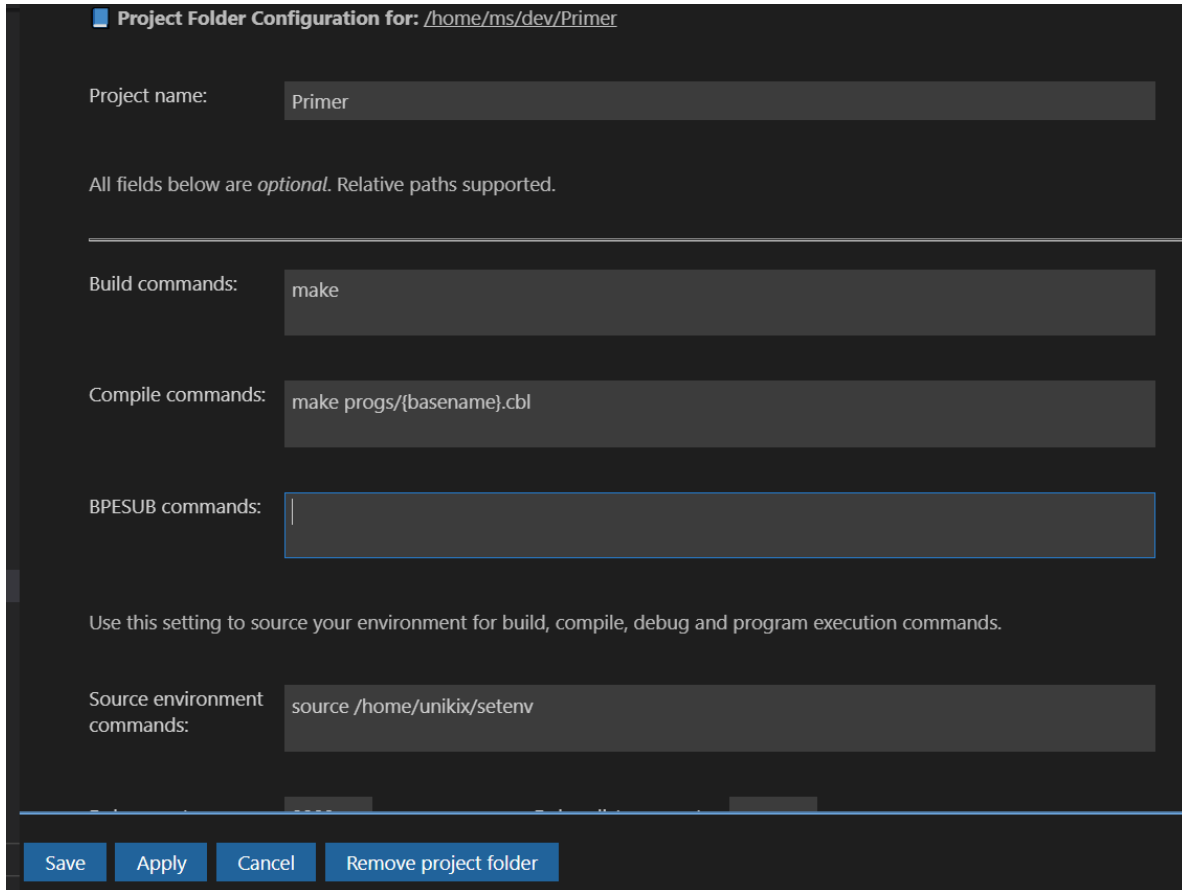
```
[unikix@rocky9 Primer]$ . /home/unikix/demo/acct16/sys/bin/userenv
[unikix@rocky9 Primer]$ make
```

In line 1, we source the environment for our TPE configuration. Line 2 executes the build process using the `makefile`. This is the process that you might use when using a terminal emulator such as Putty.

Build Command

However, when building from the command line, and if there was some type of error in the process, you would need to scroll through the entire output to find the error. A better approach might be to allow the extension to perform the build for you.

In the Build Commands option, enter the same commands as used when building from the command line. Your Build command would look as shown below. Don't forget to Save your changes.



The screenshot shows a 'Project Folder Configuration' dialog box for the project located at `/home/ms/dev/Primer`. The dialog has a dark theme. It contains several input fields: 'Project name' with the value 'Primer', 'Build commands' with 'make', 'Compile commands' with 'make progs/{basename}.cbl', and 'BPESUB commands' which is empty. Below these fields is a note: 'All fields below are optional. Relative paths supported.' At the bottom, there is a section for 'Source environment commands' with the value 'source /home/unikix/setenv'. At the very bottom, there are four buttons: 'Save', 'Apply', 'Cancel', and 'Remove project folder'.

Project Folder Configuration for: `/home/ms/dev/Primer`

Project name: `Primer`

All fields below are *optional*. Relative paths supported.

Build commands: `make`

Compile commands: `make progs/{basename}.cbl`

BPESUB commands:

Use this setting to source your environment for build, compile, debug and program execution commands.

Source environment commands: `source /home/unikix/setenv`

Buttons: Save, Apply, Cancel, Remove project folder

Pro Tip!

The Project name field can be any name of your choice, but please do not use path separators like `\` or `/`. You can also insert an emoji into the name. Actually, this is how the extension puts a bookmark emoji 📌 at the beginning of the Project name string. Custom emojis may help you identify the project/customer you are working with.

Note that we source our development environment in the 'Source environment commands' entry. This entry is used for Build, Compile, Debug and Run.

In the Explorer view, right click Primer project folder and choose Build from the context menu.

```

cd progs;kixclt -lc12 ACCT01.list;
TPE CLT TRANSLATOR (Version 16.0.8393 - 10/18/2020)
kixcpx: Translating ACCT01.list_.cl2
.....COPY Expansion completed - No error.
KIXCLT Processing ACCT01.list_.cl2
KIXCLT Processing to Build ACCT01.cbl
vcc -d64 -ef -cv -dv=0 -dcmi -pt2 -align=8 -d -sp=/home/unikix/opt/unikix16/tpe/copy:cpy progs/ACCT01.cbl
--I: #14 Compiling progs/ACCT01.cbl
--E: #15 Unexpected token <SC>; file = ACCT01.cl2, line = 20, col = 7
--E: #27 Invalid level number 02; file = ACCT01.cl2, line = 21, col = 4
--E: #27 Invalid level number 02; file = ACCT01.cl2, line = 22, col = 4

```

Problems View

The extension will now execute the commands just as if you had typed them in from the command line. A benefit of using the extension to perform the build are that errors are highlighted in **RED** and the error lines in the source code are listed in the Problems view in VS Code.

The screenshot shows the VS Code editor with a COBOL source file open. The source code is as follows:

```

8      PROGRAM-ID. ACCT01.
9      REMARKS. THIS PROGRAM IS THE FIRST INVOKED BY THE 'AC01'
10     TRANSACTION. IT ANALYZES ALL REQUESTS, AND COMPLETES
11     THOSE FOR NAME INQUIRES AND RECORD DISPLAYS. FOR
12     UPDATE TRANSACTIONS, IT SENDS THE APPROPRIATE DATA ENTRY
13     SCREEN AND SETS THE NEXT TRANSACTION IDENTIFIER TO
14     'AC02', WHICH COMPLETE THE UPDATE OPERATION. FOR PRINT
15     REQUESTS, IT STARTS TRANSACTION 'AC03' TO DO THE ACTUAL
16     PRINTING.
17     ENVIRONMENT DIVISION.
18     DATA DIVISION.
19     WORKING-STORAGE SEhhhChTION.
20     01 MISC.
21         02 MSG-NO          PIC S9(4) COMP VALUE +0.
22         02 ACCT-LNG        PIC S9(4) COMP VALUE +383.
23         02 ACIX-LNG        PIC S9(4) COMP VALUE +63.
24         02 DTL-LNG         PIC S9(4) COMP.
25         02 STARS           PIC X(12) VALUE '*****'.
26         02 USE-QID.
27             04 USE-QID1     PIC X(3) VALUE 'AC0'.
28             04 USE-QID2     PIC X(5).
29         02 USE-REC.
30             04 USE-TERM     PIC X(4) VALUE SPACES.

```

The Problems View at the bottom shows two errors:

- #15 Unexpected token <01> [Ln 19, Col 8]
- #31 Expected 'SECTION' but found 'SEhhhChTION' [Ln 19, Col 24]

Clicking on any item in the Problems View will open the source file to the line containing the error for editing. If you get an error 'File not found', please check the Source folder entry in your Project Folder settings.

Compile Command

Rather than building all COBOL files, you might wish to compile individual files. To compile a file, right click on the source code in the editor view and choose compile from the context menu.

Since the Primer project uses a makefile, we can compile individual files by specifying the target as the parameter to make. For example,

```
make progs/{basename}.cbl
```

Note the use of the substitution variable `${basename}`. This string will be replaced with the filename portion of the selected file – such as `ACCT00` and the `.cbl` extension will be appended.

Substitution Strings

When doing a build or compile you can use the following substitution strings in your commands to format the file being compiled.

- `${fullpath}` the fully qualified path to the source file
- `${filename}` the name and extension of the source file
- `${basename}` the name of the source file without the extension

You can force the substitution to output uppercase string by specifying the uppercase version of the string, for example `${BASENAME}`

Example. Given the following path: `/home/unikix/dev/project/program1.cbl`

- `${fullpath}` yields `/home/unikix/dev/project/program1.cbl`
- `${filename}` yields `program1.cbl`
- `${basename}` yields `program1`
- `${BASENAME}` yields `PROGRAM1`

You should only need to use the uppercase format of the substitution string if your filename is lowercase.

Project Folders, Build and Compile Commands

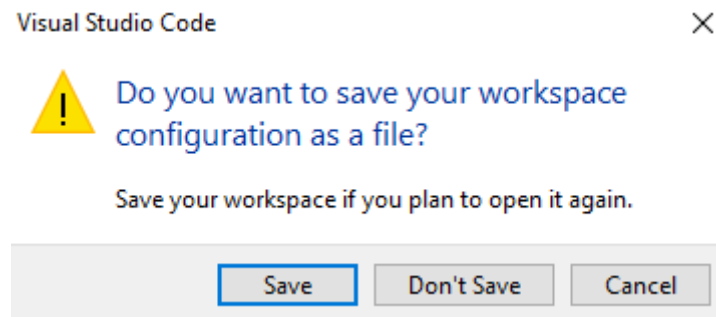


Click the Play button above to watch the video.

Click the back button in your browser to return to this document

Workspaces

If you have created a Project folder and attempt to close the VS Code editor, you will be prompted with a message box like the following:



What VS Code is asking is if it can create a file on your PC that will contain the currently opened folders on the Linux server so that the next time you run VS Code it will automatically reopen those folders.

I recommend choosing Save because it will save you time when working on projects. Since VS Code will save this file to your PC, I also recommend creating a folder, for example, C:\WORKSPACES, to save your workspace files in. This way the workspace files are not randomly scattered about your hard drives.

Also, if you give your workspace file a meaningful name – like Primer, Dev, Prod, Testing..., you can easily open the desired configuration by double clicking the workspaces file in your C:\WORKSPACES folder.

Debugging

The VS Code extension for NTT DATA COBOL supports debugging stand-alone COBOL programs, Online COBOL programs, and programs launched via BPESUB. The “Source Environment commands”, in your project folder’s settings must properly set to include the NTT DATA COBOL and JDK bin folders in the PATH environment variable.

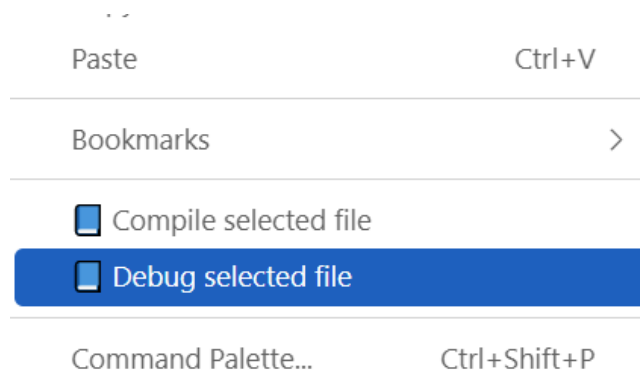
Debugging Stand-alone COBOL Programs

To debug a COBOL program, open the source file in the Editor and set any breakpoints desired by right clicking in the left margin of the Editor window. A red circle will appear.

```
91
92      PERFORM VARYING I FROM 1 BY 1 UNTIL I > 8
93      DISPLAY "USING SUBSCRIPT: " MY-TABLE-ENTRY(I)
94      DISPLAY "USING REFERENCE MOD: " MY-TABLE(I:1)
95      END-PERFORM.
96
97      PERFORM VARYING I FROM 1 BY 1 UNTIL I > 2
98      AFTER J FROM 1 BY 1 UNTIL J > 3
99      DISPLAY "MULTI-LEVEL TABLE: " MY-TABLE-2-PART-1(I) ", "
100      MY-TABLE-2-PART-2(I,J)
101      END-PERFORM
102      move nulls to string-test.
```

Verify that you have set the Debug port in your Project Folder settings (typically 9999) and that your Output and Source folders are defined. Also verify that you are working with the correct project file folder.

Right click on the source file in the Editor and choose Debug selected file from the context menu. **Do not use the Run and Debug icon from the Activity bar** -- VS Code will attempt to debug a program on your Windows desktop.



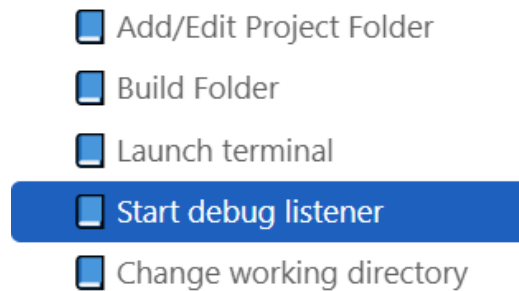
Use the Debug toolbar to navigate through your program.



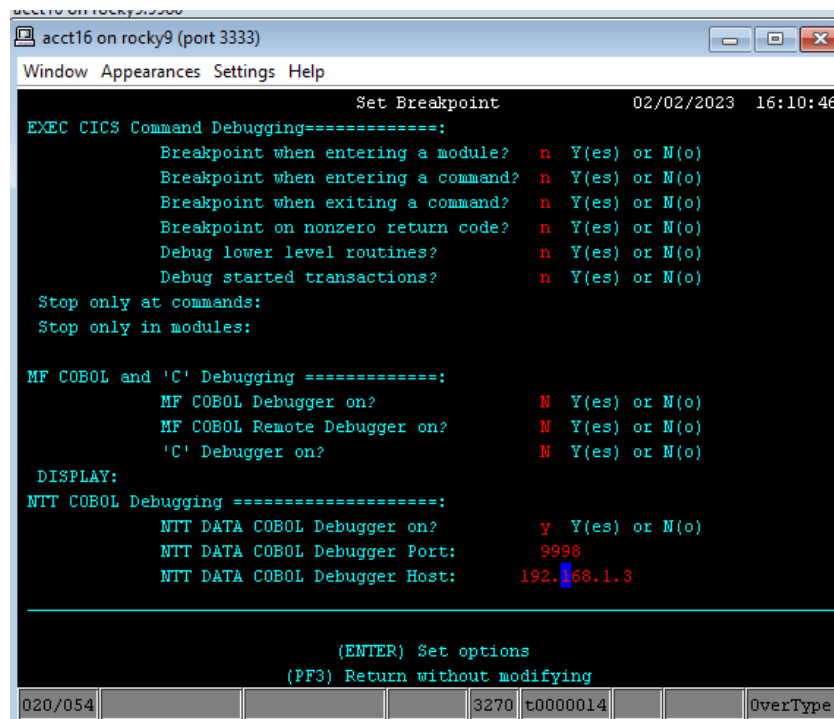
You can set/remove breakpoints at any time and add watches for variables using the right click context menu. Hovering your mouse over working storage variables will display the value of the item.

Debugging Online Programs

Online programs requires that the Debug Listener be started. Verify that the Debug Listener port is specified in your Project Folder, and then right click on the folder in the Explorer view. Choose Start debug listener (you only need to do this once).



In the CEDF utility, set the NTT COBOL Debugging port to the port number for your debug listener and enter the IP address of your desktop computer.



Once you start your program, the debug extension should automatically connect and display your source file.

Debugging Batch Jobs

You can use the extension to translate and debug your JCL files. For this example, a very simple JCL script is used.

```
//*  
//PROGRAM1 JOB  
//*  
//PROGRAM1 EXEC PGM=PROGRAM1  
//*
```

The JCL launches a program named PROGRAM1 that is part of our project. In the BPESUB command for the Project Folder, the following lines are specified:

BPESUB commands:

```
./home/unikix/demo/node16/batchenv sub1
BPESUB -Asub1 {basename}
```

The first line sources the environment for our 'sub1' subsystem. The second line executes BPESUB for the selected file in the editor.

This same command is used when debugging the PROGRAM1 that is launched by the JCL – note that -a is **not** specified in our command (it is automatically inserted by the extension if you are debugging). To start a debug session simply right click on your JCL file in the editor and choose Debug selected file.

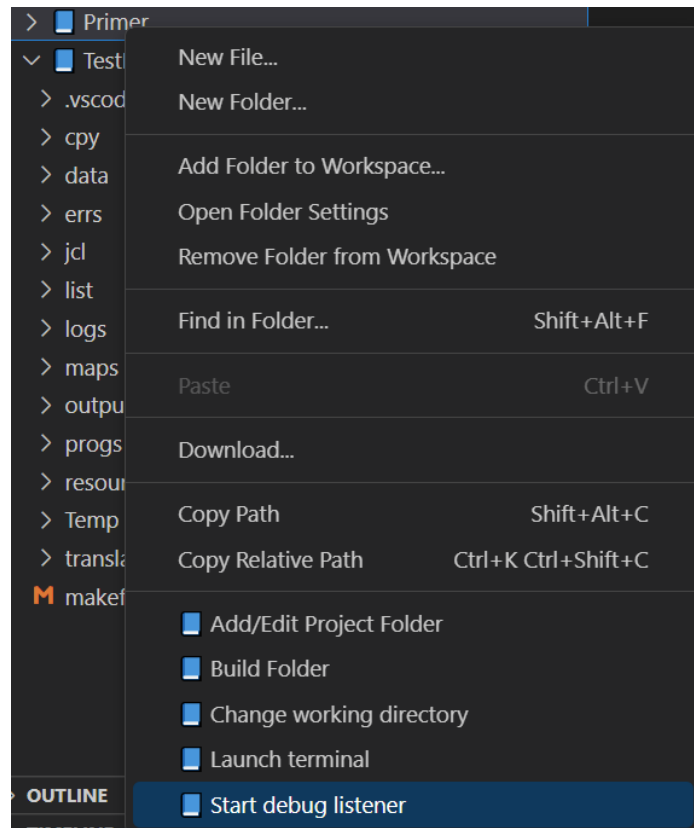
When debugging, the extension automates the following steps:

- 1) Appends the -a option to the BPESUB command.
- 2) Exports the VDEBUGREMOTE=Y environment variable.
- 3) Watches the BPESUB output for the Job Number.
- 4) Launches the anmjob with the Job Number.
- 5) Loads PROGRAM1 into the VS Code debugger when the EXEC command is executed.

Without the extension, the user would need to perform all steps above manually using two Putty connections. The extension automates all of these steps for you.

Debug Listener

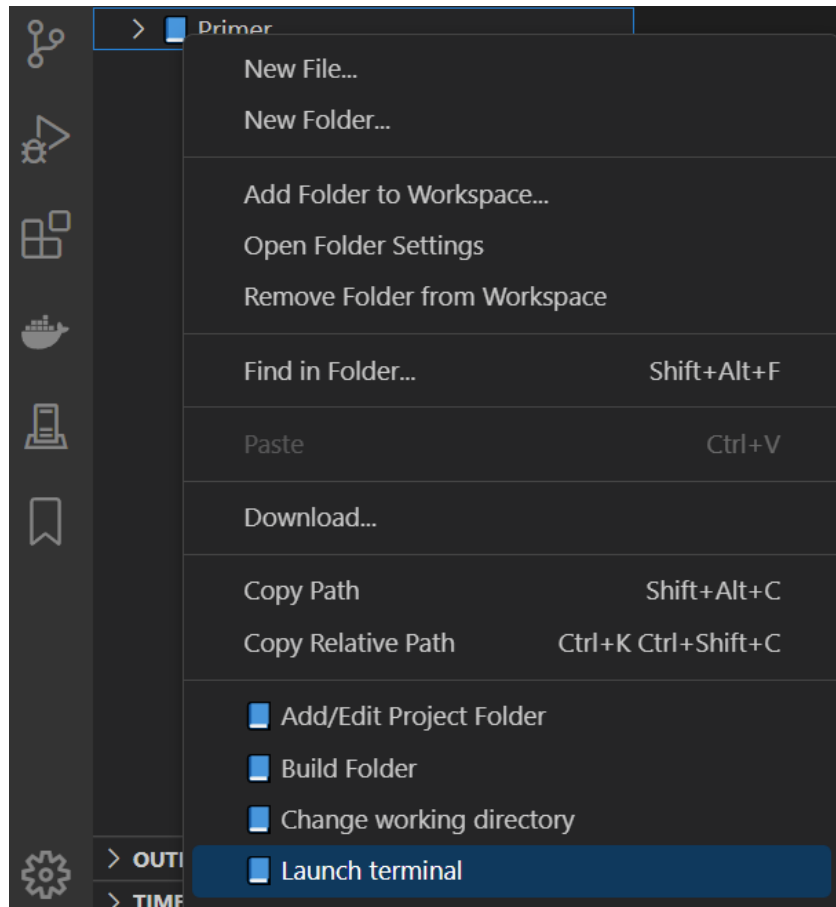
You can start the Debug Listener in the extension to listen for debugging connections from your Online programs or from programs launched with `vcrun -o`.



Access the Start debug listener menu option by right clicking the desired project and choosing the menu option. It is important to note that the debug listener is 'assigned' to whichever Project folder it was launched from. When a debug connection is established, the source files for the program being debugged will be located using the Project folder's path and the Source directory specified in that Project folder.

Integrated Terminal

Right clicking on any folder in the Explorer view will display the menu below:



Choosing the [Launch Terminal](#) option will open a terminal window for the specified folder. In the terminal, you can execute any command or program and even copy/paste text to/from Windows. You can open as many terminal sessions as required. The icons to the right of the terminal window can be used to select or remove the terminal

VS Code Tasks

You will likely use lots of tools, scripts and other tools like building, packaging, testing or deploying your software during development. The Tasks feature in VS Code is designed just for this purpose. Existing tools can be used from within VS Code without having to type commands into a terminal session. Task definitions are stored in a file named `tasks.json` and should be saved under the `.vscode` folder for your project. For example,

```
<project folder path>/vscode/tasks.json
```

When creating your first task, VS Code will want to create the `tasks.json` file in the topmost folder of your server connection. A preferred approach would be to manually create the tasks that are specific to the project you are working on – for example, the Primer project, in the `unikix/home/dev/Primer/.vscode/` folder. You will want to be sure to include this file in your GIT repository.

Navigate to your Primer folder and locate the `.vscode` folder. Right click the `.vscode` folder and choose `New File...` Enter `tasks.json` as the filename.

Paste the following into the `tasks.json` file:

```
{
  "version": "2.0.0",
  "tasks": [
    {
      "type": "NTT DATA",
      "label": "Sample Task Template",
      "commands": [
        "pwd",
        "ls"
      ]
    },
  ]
}
```

Replace the 'commands' strings to the commands you wish to execute.

Please note that VS Code uses the JSON format for most, if not all, of its configuration files. There are plenty of online tutorials on the JSON format and VS Code offers intelli-sense and Problems View reporting to assist you with editing the file.

Task Definition Items

type	Use NTT DATA COBOL as the type to execute the task on the remote server. Other task types will execute the task locally.
label	This is the text that is displayed when you enter Task Run from the Command Palette in VS Code.
commands	Any number of commands to execute. Note that the extension will first run any commands entered into the Project Folder's "Source environment commands", and will then change directory to the Project folder.

Task Variables

You can use a number of substitution variables in your task's commands. A full list of predefined variables is here:

<https://code.visualstudio.com/docs/editor/variables-reference>

You can also use the extension's substitution variables described earlier in this document.

Task Icon

You can also configure an icon for your task. The options for the id and color parameters will be provided by intellisense.

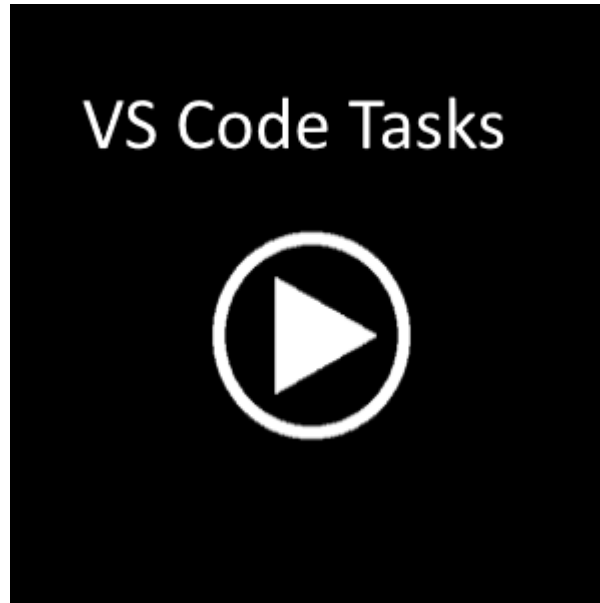
```
{
  "version": "2.0.0",
  "tasks": [
    {
      "type": "NTT DATA COBOL",
      "label": "Sample Task Template",
      "commands": [
        "pwd",
        "ls"
      ],
      "icon": {
        "id": "chevron-up",
        "color": "terminal.ansiMagenta"
      }
    },
  ],
}
```

Copy and paste this text to your tasks.json file for a template to work with.

Other Task Features

You can also use *pickString*, *promptString*, and other configuration variables (*like globals*) in your task definition. Please see <https://code.visualstudio.com/docs/editor/variables-reference> for more details. Step-by-step instructions on these features will be added to this document in later versions.

WARNING: If you use *pickString*, *promptString* inputs, you must be working in a VS Code workspace. If your Explorer shows UNTITLED (WORKSPACE) these commands will not work and will hang the task execution. Workspaces are discussed earlier in this document. This unexpected behavior will be reported to the VS Code team.



*Click the Play button above to watch the video.
Click the back button in your browser to return to this document*

Editor Features

To configure setting for the COBOL Editor, choose the settings icon from the Activity bar, Settings, Extension and then navigate to the NTT Data Enterprise COBOL option. From there you can configure Autocomplete style, valid Copybook Extensions, valid COBOL file extensions, tab stops and more.

Deleting/adding tab stops in Setting Configuration scrolls to the next extension. Unfortunately, the Configuration screen is controlled completely by VS Code. It is much easier to edit the settings JSON file manually for the tab

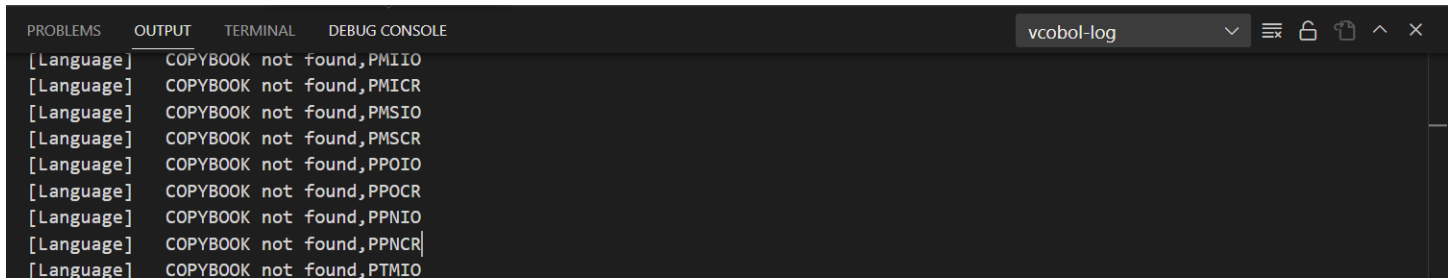


settings using the Open Settings button, , in the top right corner of the VS Code window.

Copybooks

In order to process copybooks, you must have the necessary Copybook folder paths specified in your Project Folder configuration. The extension will always search for the name of the copybook specified in the COBOL source file first and then, if not found, will iterate through the list of possible copybook extensions. So, please be frugal with the copybook extension list – searching for copybooks using all the copybook paths times the copybook extensions will be

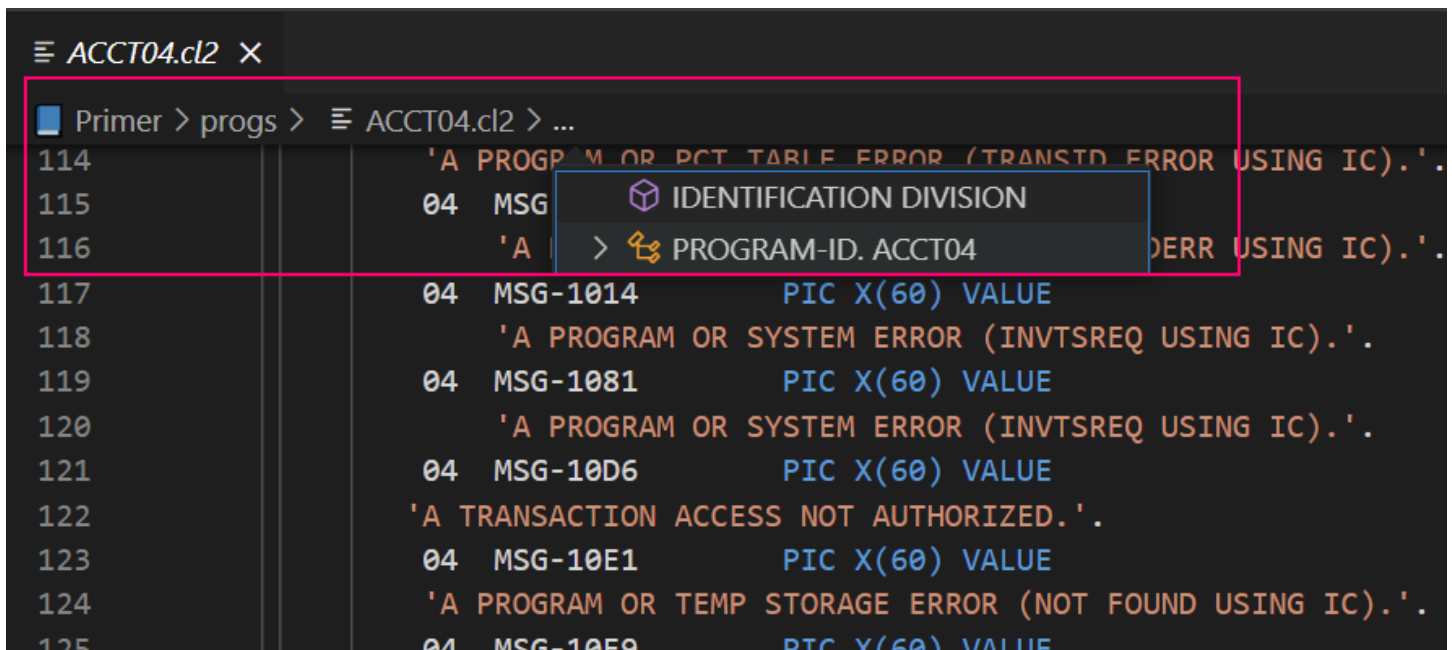
CPU/Disk intensive task if the copybook does not exist. Use the VS Code Output pane and select vcobol-log to view errors from the COBOL language parser.



```
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE vcobol-log
[Language] COPYBOOK not found,PMIIO
[Language] COPYBOOK not found,PMICR
[Language] COPYBOOK not found,PMSIO
[Language] COPYBOOK not found,PMSCR
[Language] COPYBOOK not found,PPOIO
[Language] COPYBOOK not found,PPOCR
[Language] COPYBOOK not found,PPNIO
[Language] COPYBOOK not found,PPNCR
[Language] COPYBOOK not found,PTMIO
```

Breadcrumbs

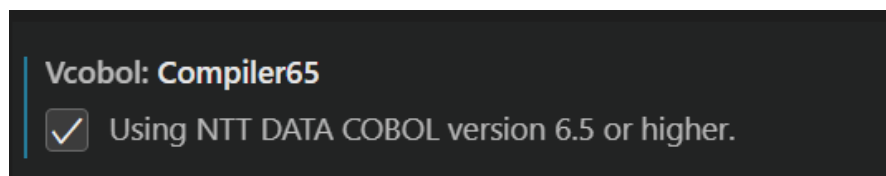
The Editor features an Outline view that is displayed in the Activity bar and in the bread crumbs toolbar in the editor. Using the breadcrumbs toolbar is a great way to gain screen space since you can minimize the Activity bar and click through the various items in the breadcrumbs toolbar to navigate through your folders and files.



```
ACCT04.cl2 X
Primer > progs > ACCT04.cl2 > ...
114 'A PROGRAM OR PCT TABLE ERROR (TRANSD ERROR USING IC)'.
115 04 MSG IDENTIFICATION DIVISION
116 'A > PROGRAM-ID. ACCT04 DERR USING IC)'.
117 04 MSG-1014 PIC X(60) VALUE
118 'A PROGRAM OR SYSTEM ERROR (INVTSREQ USING IC)'.
119 04 MSG-1081 PIC X(60) VALUE
120 'A PROGRAM OR SYSTEM ERROR (INVTSREQ USING IC)'.
121 04 MSG-10D6 PIC X(60) VALUE
122 'A TRANSACTION ACCESS NOT AUTHORIZED.'.
123 04 MSG-10E1 PIC X(60) VALUE
124 'A PROGRAM OR TEMP STORAGE ERROR (NOT FOUND USING IC)'.
125 04 MSG-10F9 PIC X(60) VALUE
```

NTT DATA COBOL 6.5 or higher

If you are using NTT DATA COBOL version 6.5 or higher you will want to select the option shown below:



The only impact this setting will have is when displaying the red and yellow squiggle marks for errors and warnings. NTT DATA COBOL 6.5 sets the column for errors and warning 6 columns to the left of previous versions of COBOL. So, not setting this option appropriately will mean your squiggle marks will be shown in the wrong column.

Auto Complete

The extension also supports auto complete when typing and will present a list of data division, procedure division and keyword items to select from. The auto complete implementation will attempt to determine which items to list based upon the verbs used in the source code currently being edited.

Note that by default, VS Code will supplement the auto completion items with text found within the source file. This text may include non-relevant items such as text from a commented line. It is suggested that you disable the built-in VS Code autocompletion by configuring the following setting:

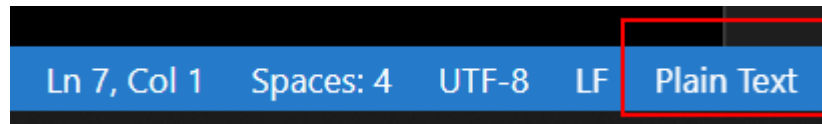
```
"editor.wordBasedSuggestions": false
```

The editor supports a number of other features such as:

- Goto Definition
- Rename symbol
- Peek all references
- Goto references
- Change all occurrences.

Files with no Extension

It is common for mainframe files, like copybooks and JCL files, not to include a file extension. Unfortunately, editors like VS Code depend upon file extensions to determine type of editor to open the file. If VS Code cannot determine the file type, then the **Plain Text** editor will be used. You can observe which editor VS Code is using by viewing the status bar.



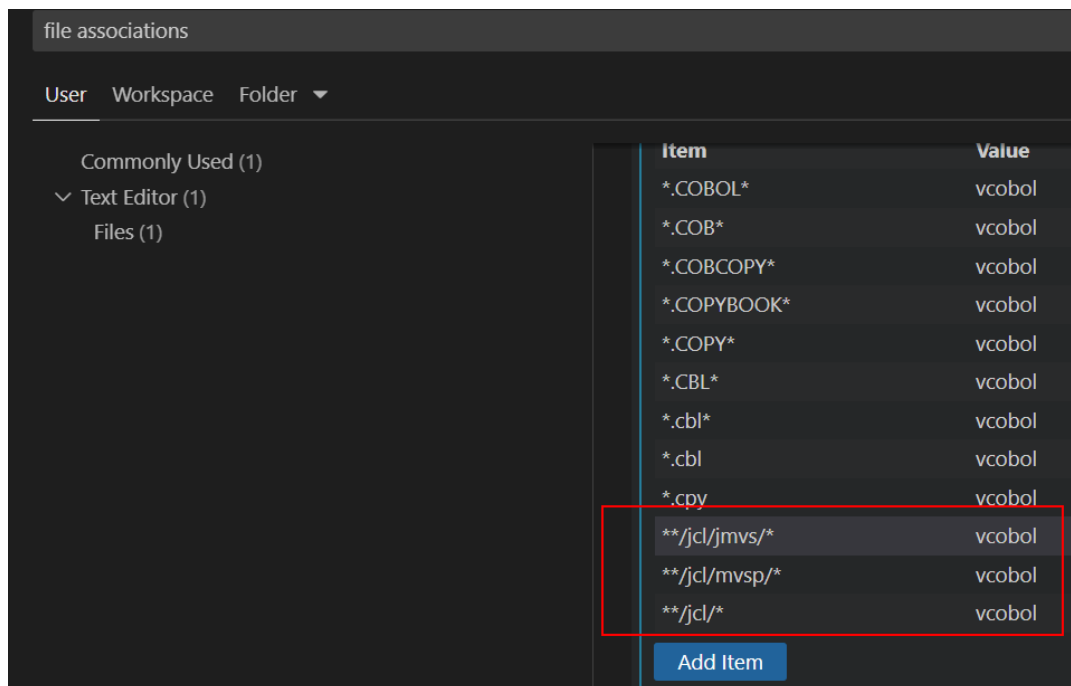
You can specify the specific editor to use for extensionless files using the following syntax in your settings:

****/<path>/***

Where path is a partial path to the extensionless file and specifying the editor (Value) as vcobol.

To access this setting, choose the VS Code Settings icon and search for “file associations”. In the screen capture below, three settings have been added for extensionless JCL files and have been assigned to our vcobol extension.

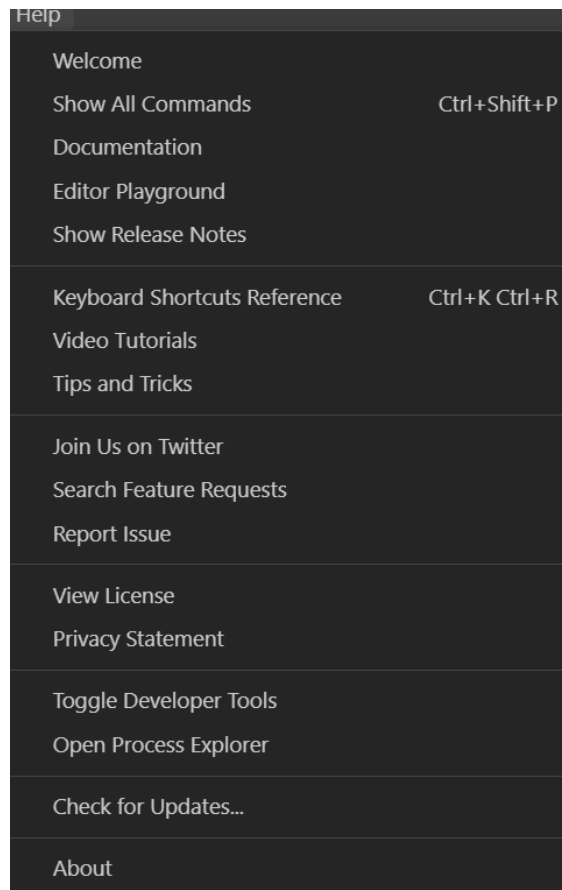
Note that if you open a file with no extension, that that file path contains the text “jcl”, the file is considered to be a BATCH script file.



Additional Resources

VS Code has some excellent resources right in its Help menu. In particular:

Editor Playground:	An interactive document for practicing some of the VS Code features.
Video Tutorials:	Links to short, 2-to-5-minute tutorials on editing, customizing and personalizing VS Code.
Tips and Tricks:	A document detailing tips and tricks with VS Code.
YouTube:	Visit the https://www.youtube.com/@code/videos channel for tons of videos.



Known Issues

- Need to document color schemes (for theme authors).
- Copy and paste files from Windows to Remote not working using menu options or CTRL-V. You can however, drag and drop files from Windows Explorer to the VS Code Explorer pane.
- Git Integration. Currently, VS Code tries to launch git on your Windows workstation and not on the Linux server. This will be corrected in a future release.
- VS Code Terminal menu option attempts to navigate to the Linux server path on your Windows workstation instead of the Linux server. Please use the right click menu option Launch Terminal instead.
- More testing for Change All Occurrences and Rename (with copybooks) is needed.
- Not all build/compile errors are reported in the PROBLEMS view. In particular, TPE/BMS compiler errors, VCPREP errors. This will be corrected.
- Current version of the extension is limited to one host server at a time. You cannot connect to multiple servers simultaneously.
- If you use the Change Directory and change to the root folder, /, VS Code will display the folder name as the name of your Connection. Apparently, VS Code wants to have letters in the folder name. The subfolders of root, /, display properly.
- WARNING: If you use another extension designed for COBOL programs your results will be unpredictable and are not be supported. Please do not install another extension for handling COBOL, JCL and BMS files.
- If you change the name of a Program Folder, VS Code may not re-sort the files in the Explorer View. To get the proper sorting, just re-connect to the same server using the server icon in the activity bar.