



Visual Studio Code MV# Extensions



ONgroup Intl

■ 707B E Cervantes St ■ Pensacola FL 32501 ■ 800-573-0300 ■ www.ONgroup.com

Copyright © 2018 ONgroup Intl

All rights reserved.

ONgroup Intl make no representations that the use of its products in the manner described in this publication will not infringe on existing or future patent rights, nor do the descriptions contained in this publication imply the granting of licenses to make, use, or sell equipment or software in accordance with the description.

Possession, use, or copying of the software described in this publication is authorized only pursuant to a valid written license from ONgroup Intl or an authorised sub licensor.

Neither ONgroup Intl nor its employees are responsible for any errors that may appear in this publication. The information in this publication is subject to change without notice.

All other trademarks and service marks are property of their respective holders.

Contents

Preface	3
1 Introduction	4
2 Pre Requisites	5
3 Installing Visual Studio Code	6
4 Configuring Visual Studio Code for MVON#	9
5 Connecting to a MVON# Server	11
5.1 Testing the connection	14
5.2 Associating Programs with MVON#.....	16
5.3 Additional MV# Developer Settings	16
6 MV# Developer Features	17
6.1 Syntax Highlighting	17
6.2 Intellisense	18
6.3 Find All References.....	18
6.4 Goto/Peek Definition.	19
6.5 Internal Subroutine lookup	20
6.6 Compiling and Cataloging your programs.....	21
6.7 Formatting Programs	21
7 Connecting to other MV Platforms	22
7.1 Universe	23
7.2 Unidata	24
7.3 OpenQM.....	25
7.4 jBASE	26
7.5 D3.....	27
7.6 Testing remote connectivity	28
7.7 Debugging remote connection issues	29
8 MV# Debugger Extension	30
8.1 Starting the debugger.....	31
8.2 Debugging features.....	33
8.3 Call Stack	35
9 MV# TCL Extension	37
9.1 Establishing a TCL session.....	38
9.2 TCL Features	40
9.2.1 Intellisense	40
9.2.2 Automatically execute script lines.....	42
9.2.3 Error highlighting.....	43
9.2.4 Dictionary details display	44

PREFACE

Purpose of this guide

This document describes how to use Visual Studio Code as the MVON# Development Environment.

1 INTRODUCTION

Visual Studio code is a feature rich IDE that allows programmers to develop and debug code in various languages. MVON# provides developers with the ability to program their MV applications with a variety of program languages including BASIC, C#, Python, JavaScript and Typescript. This makes Visual Studio Code an ideal IDE as it supports all the above languages.

In order to fully utilize the power of VSCODE, ONGroup has built a set of extensions to cater for MVON# BASIC language. There are also extensions to support all the MVON# supported languages.

This extension includes the following features

1. Code highlighting for MV# BASIC Programs
2. Intellisense for the MV# BASIC Statements and Functions
3. Code folding
4. Code formatting
5. Goto/Peek Definition. Automatically jump to and peek internal subroutines
6. Goto/Peek Definition. Automatically peek/load CALL, CHAIN and INCLUDE routines
7. Syntax checking for GOTO/GOSUB's, LOOPS, CASE STATEMENTS and IF THE/ELSE statements
8. Access your remote MVON# files and programs
9. Find all References of a word in current program
10. A visual debugger

Visual Studio Code is available on Windows, Linux and Mac OSX.

2 PRE REQUISITES

The following environment is required in order to use Visual Studio Code.

1. Windows, Linux or Mac OSX machine.

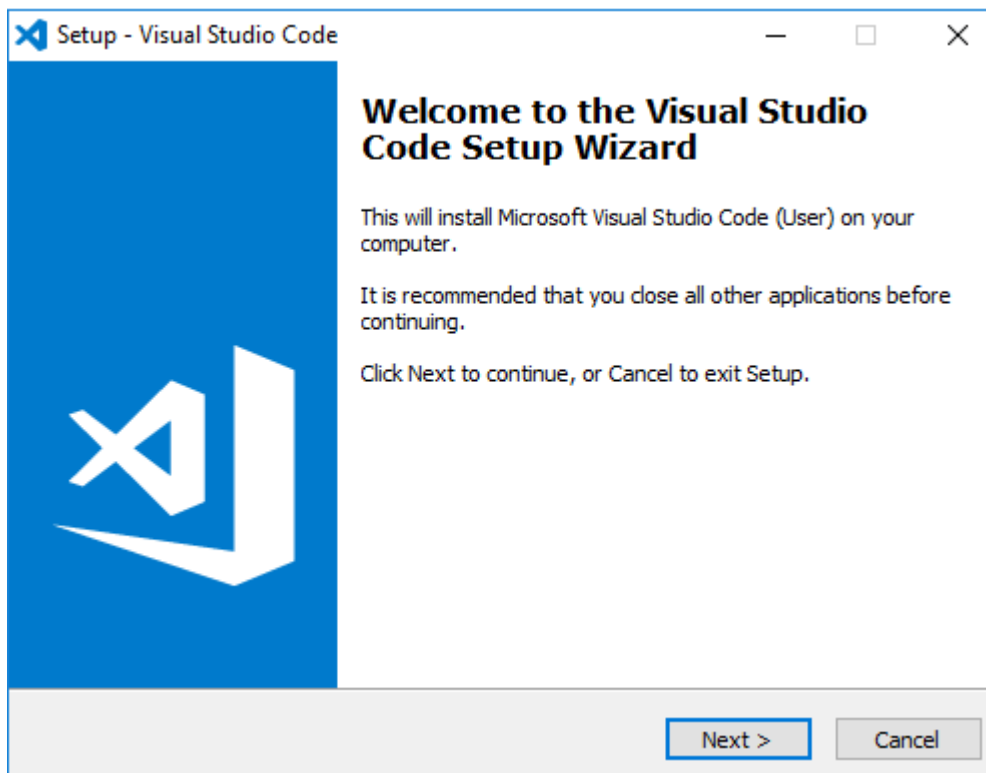
3 INSTALLING VISUAL STUDIO CODE

Visual Studio Code can be downloaded from the following link:

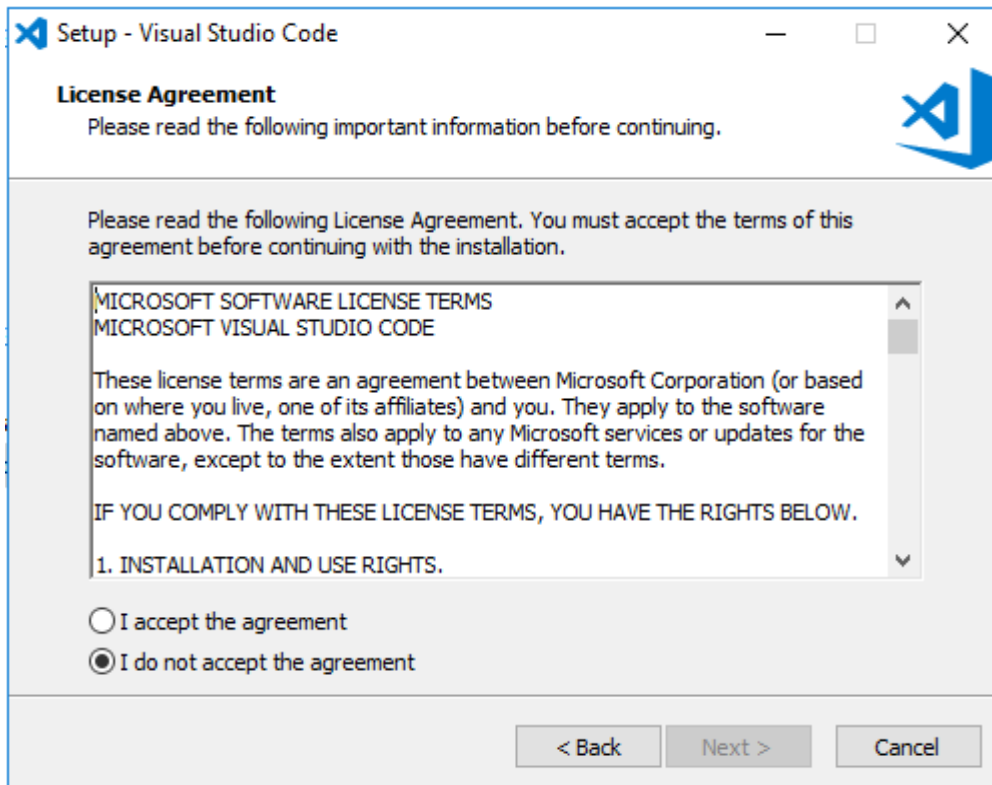
<https://code.visualstudio.com/Download>

You can select the version for operating system. This guide describes how to install the Windows version of Visual Studio Code.

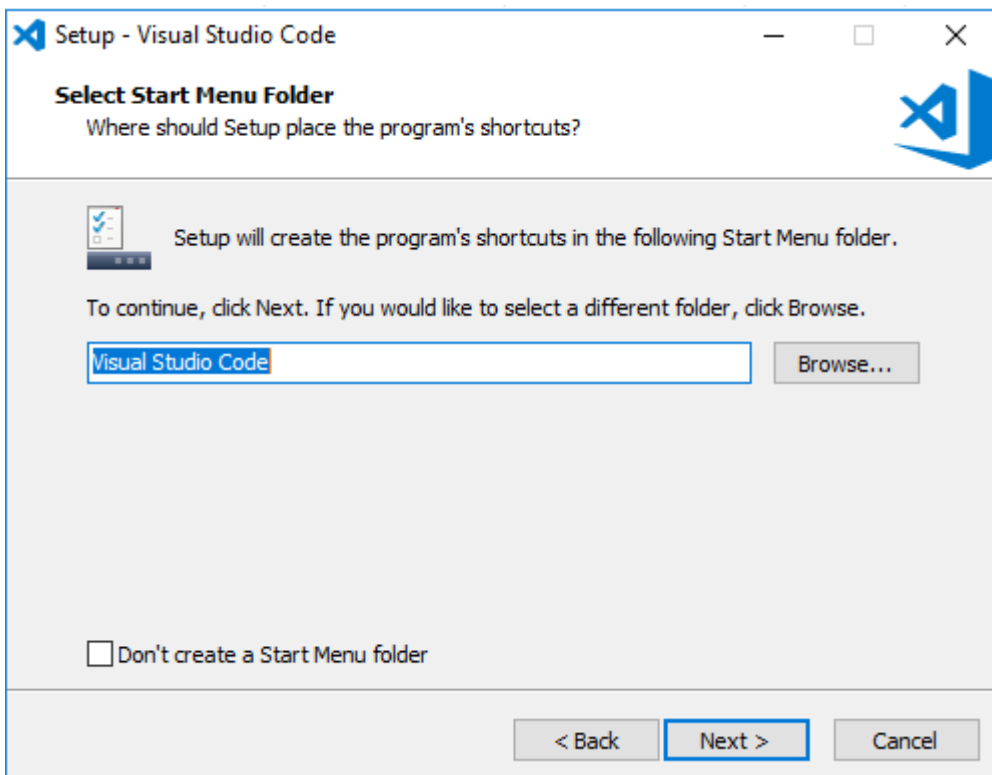
Depending on your Windows operating system, run either the 32 bit **VSCodeUserSetup-ia32-1.26.1.exe** or the 64 bit **VSCodeUserSetup-x64-1.26.1.exe**



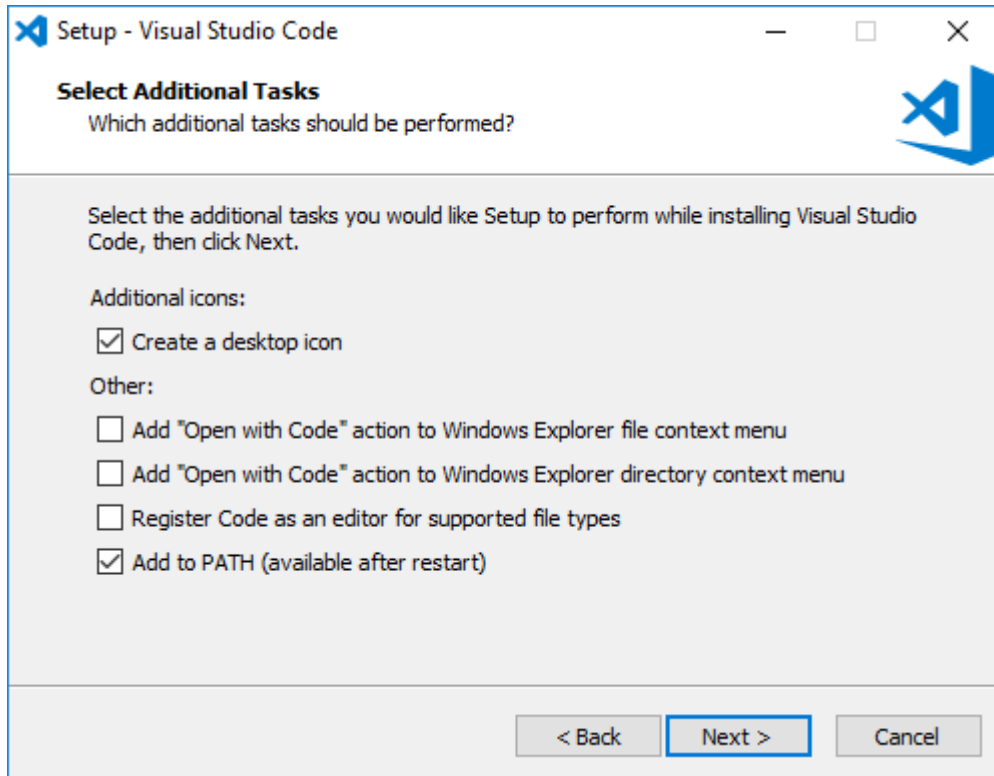
Select Next



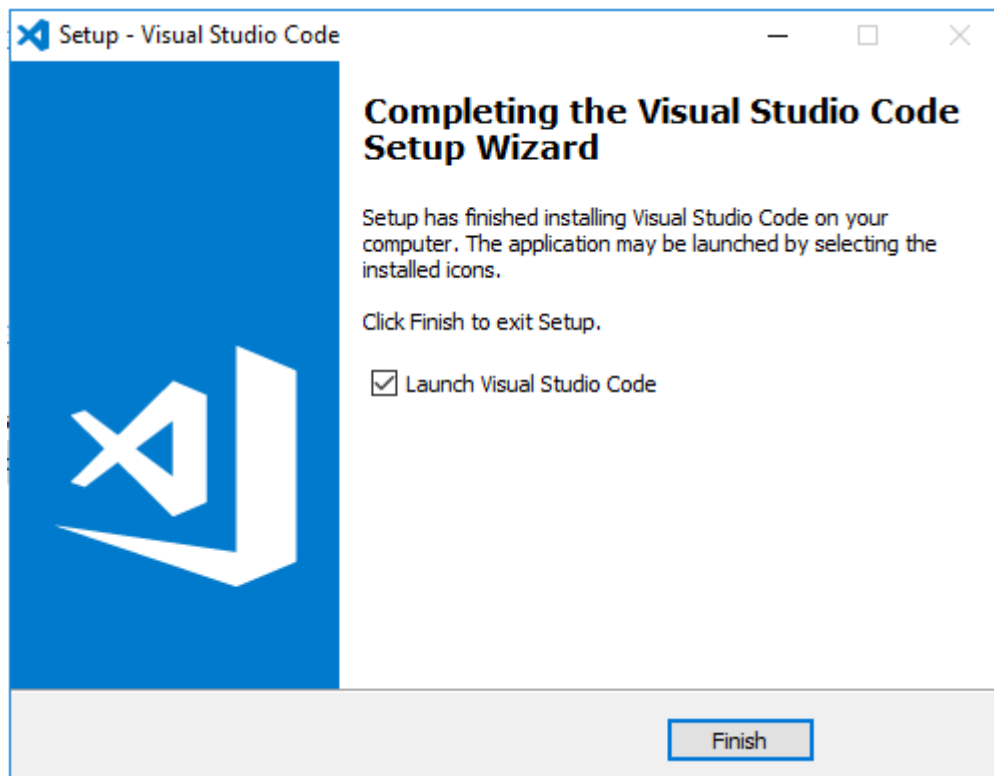
Accept the License Agreement and select **Next**



Accept the defaults or specify your folder and select **Next**



Select the options you would like to include in the install and select **Next**

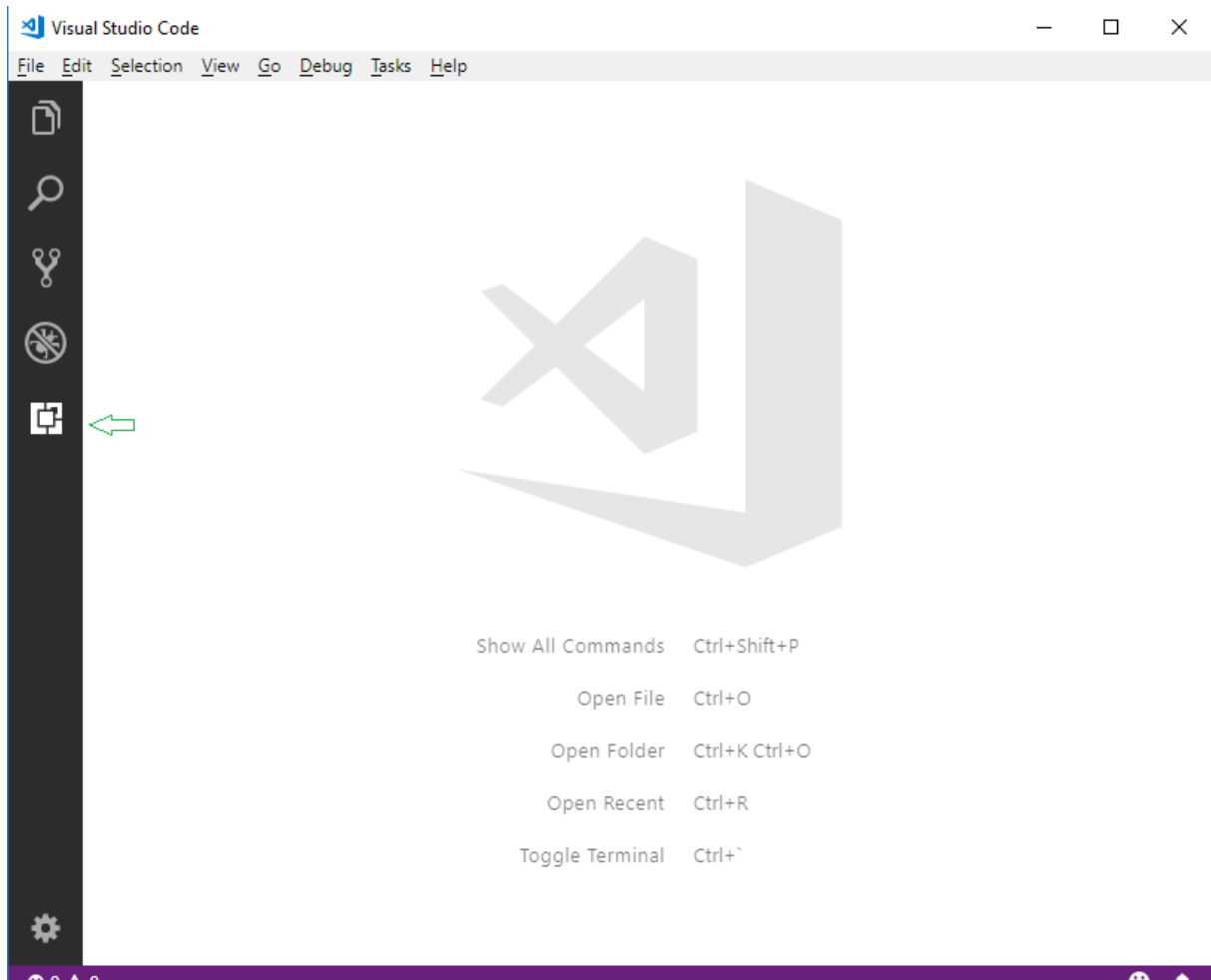


Visual Studio Code is now installed.

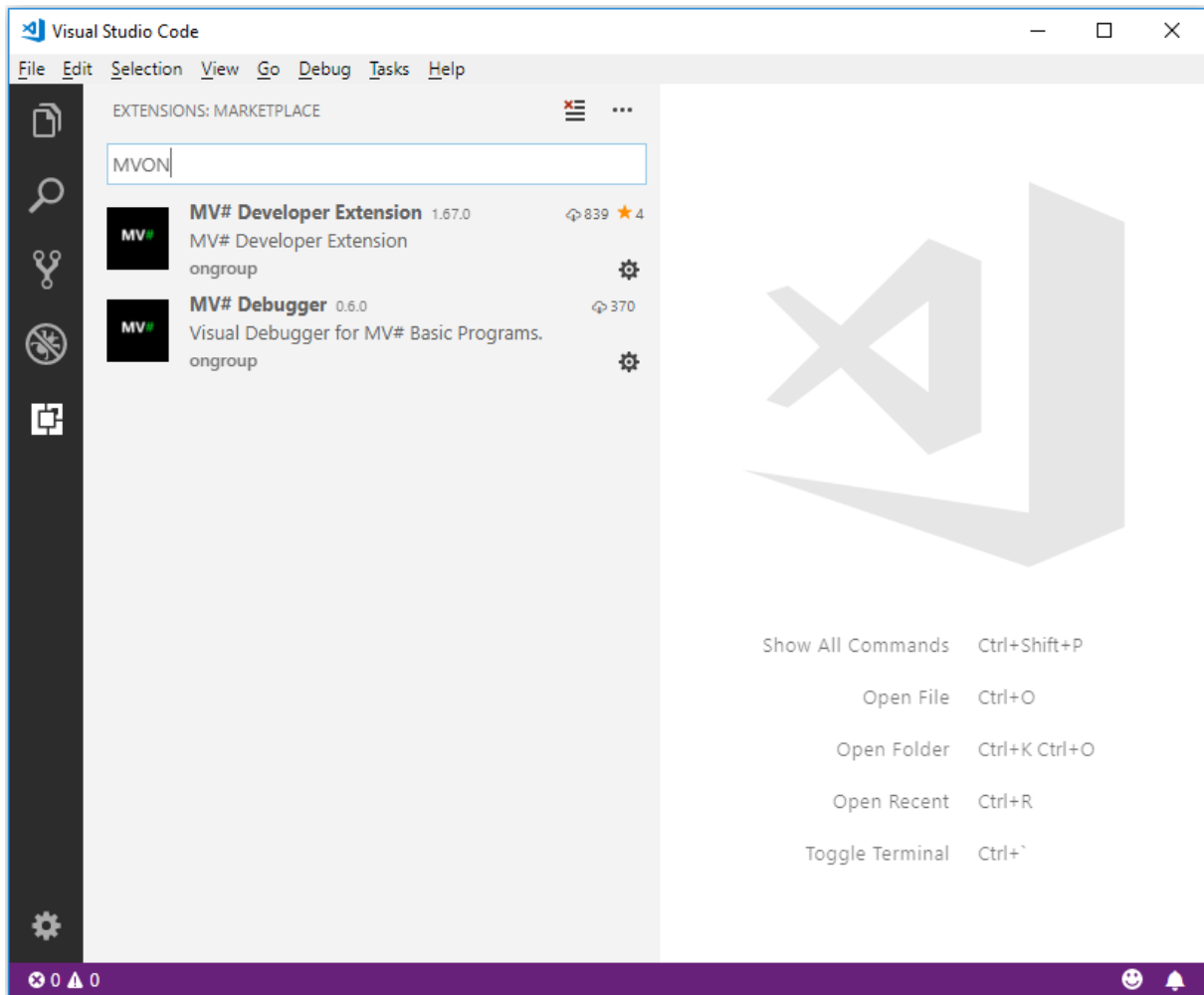
4 CONFIGURING VISUAL STUDIO CODE FOR MVON#.

Before we can start using the MVON# features for Visual Studio Code, we need to install the extensions. VSCODE has a automated download and installation process for extensions.

Start VSCODE and select the Extensions Button



In the search box, type MVON and press enter:



You can select the Developer Extension and the Debugger extension by selecting the download image.

Once the extensions are installed we are ready to start accessing our MVON# server.

5 CONNECTING TO A MVON# SERVER

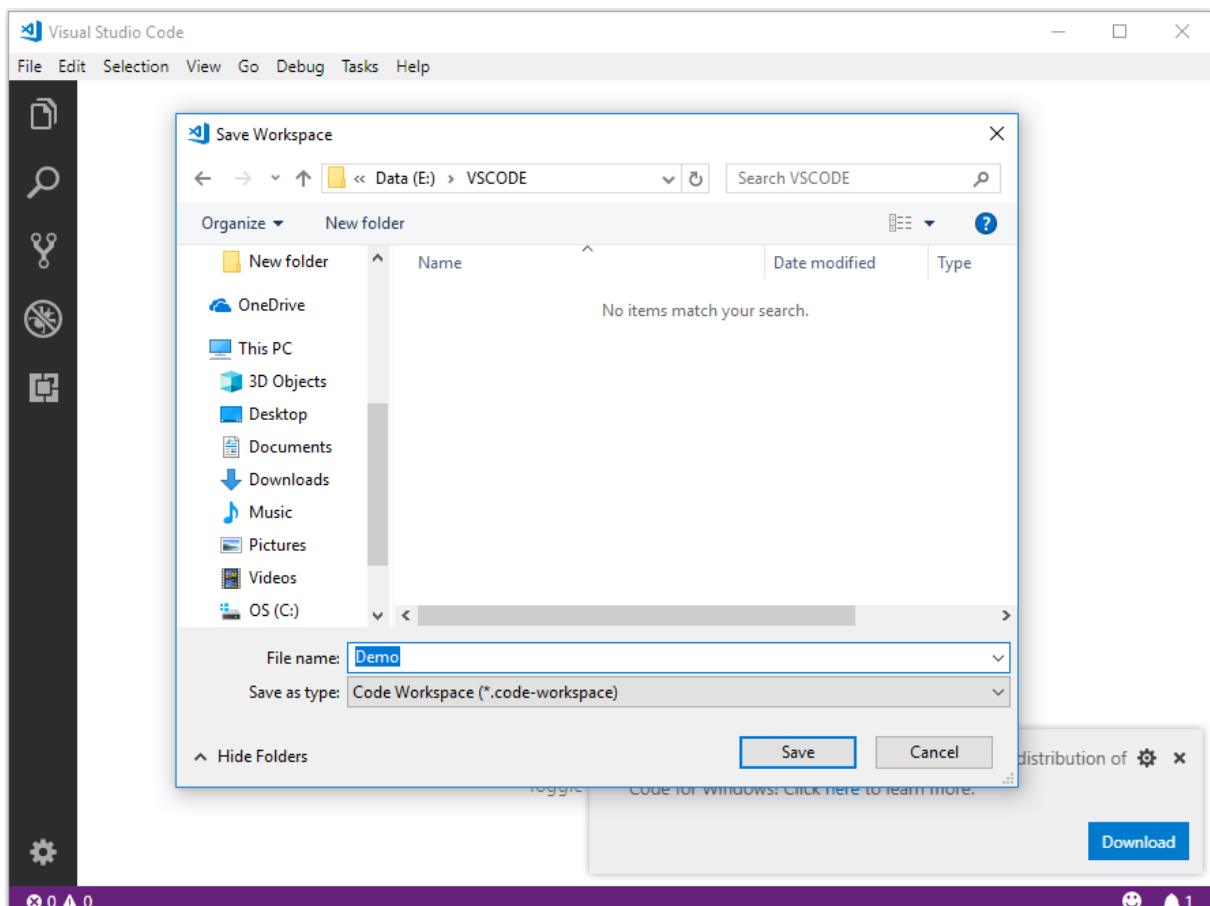
The extension allows us to connect to remote MVON# servers and edit BASIC programs. We need to configure a **Workspace** that will contain all the parameters required to connect and login to the remote MVON# Server.

The simplest method is to create a directory on your machine where we will save the Workspace definitions. If we have multiple server and multiple accounts on each server, we can create multiple Workspace's that points to a particular server and account.

In order to connect to a MVON# Server, we required the following information:

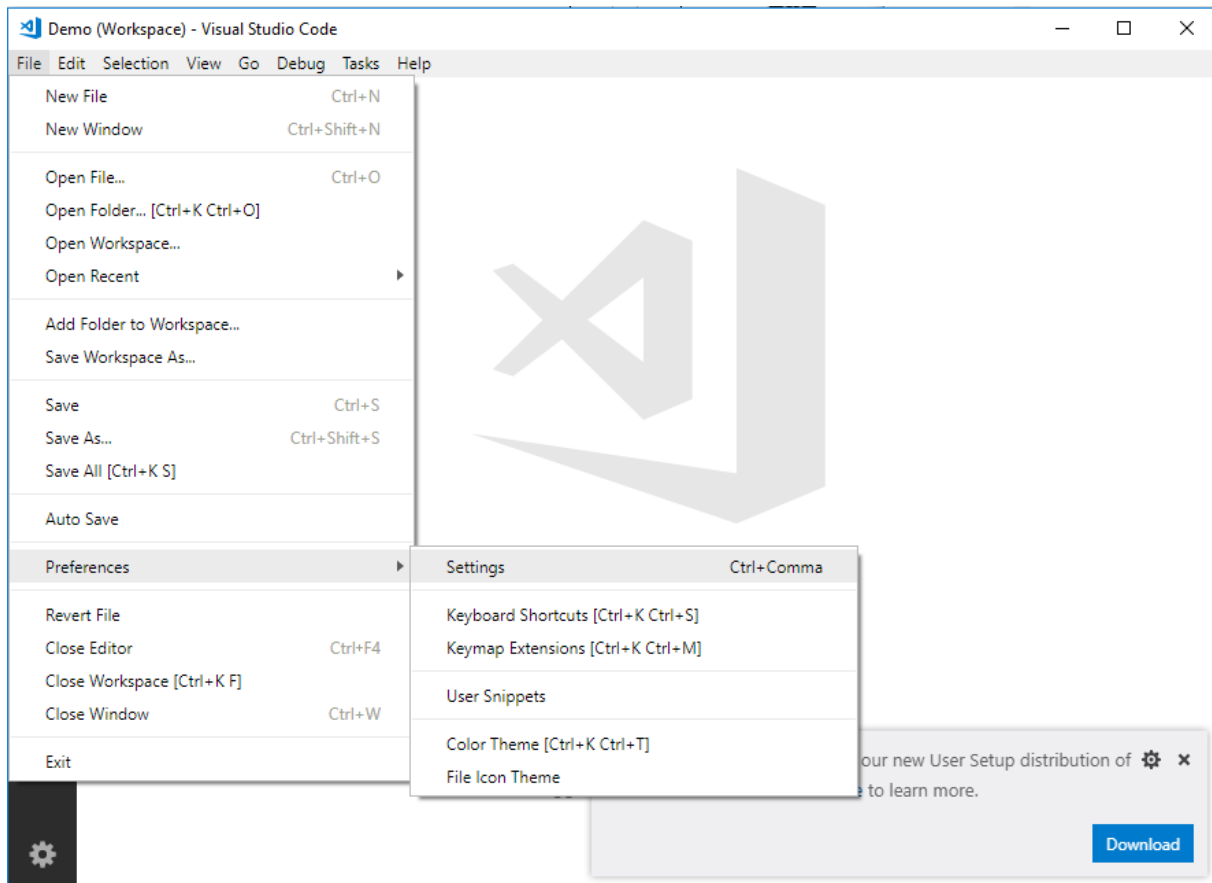
1. Hostname or IP Address of the MVON# server.
2. User name to login into the server
3. Password for the user above.
4. Account name to connect to on the MVON# Server

To create a new Workspace, select **"Save Workspace As"** from **File** Menu. I have created a Folder called VSCODE on the E: drive where I store all my Workspace definitions.



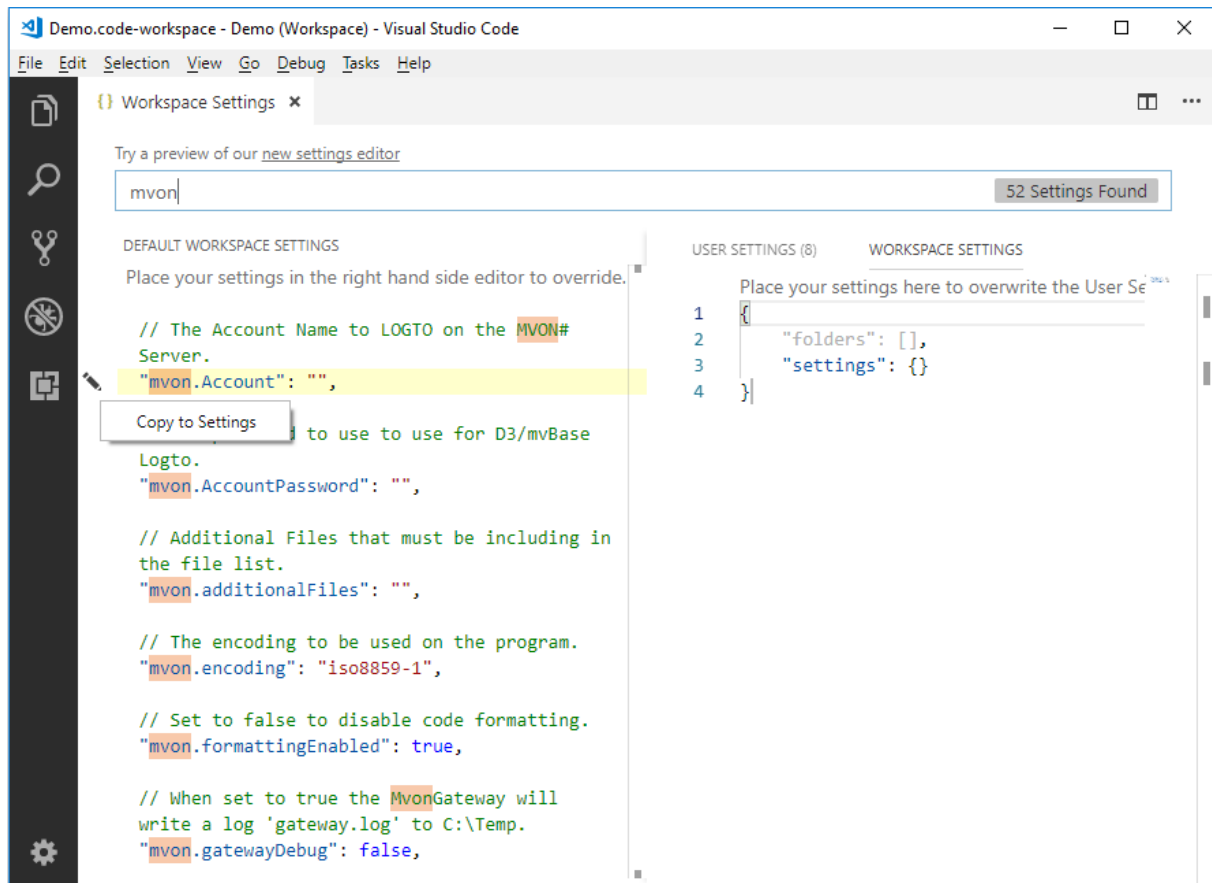
The will create a blank Workspace called Demo that we can now configure to point to our MVON# server.

To configure the connection parameters , select **File, Preference, Setting** from the menu.



This will bring up the Settings pane in VSCODE, make sure you select the **Workspace Tab**, and then type **mvon** in the search box. This will display a list off all the parameters that can be set for the MV# Developer extension.

Hover your mouse over a parameter and you will be given the option to **Edit** and **Copy to Settings**.



After adding all the parameters to the workspace, your settings should be like this:

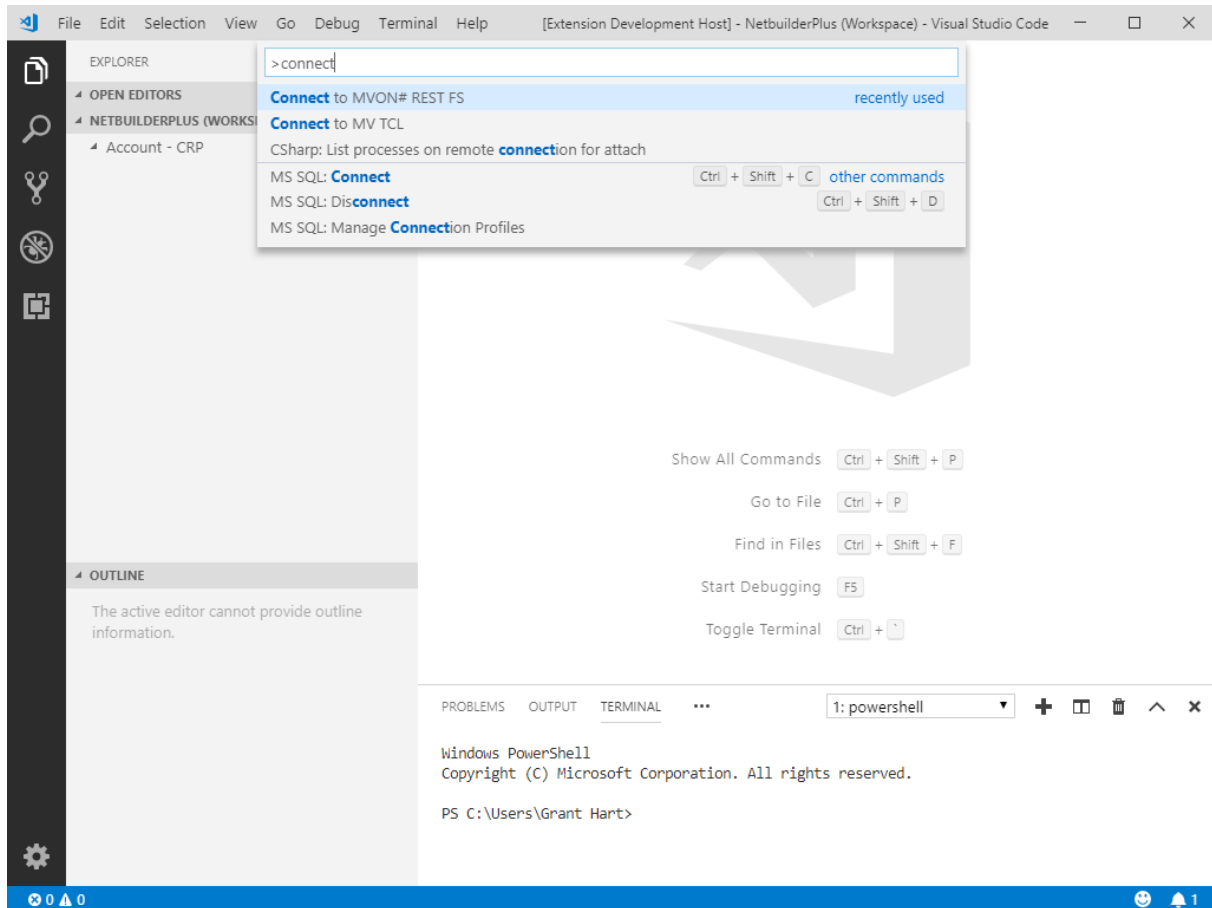
```
{
  "folders": [
    {
      "uri": "RestFS:/",
      "name": "Account - DEMO",
    }
  ],
  "settings": {
    "mvon.RestPath": "http://localhost/mvonrest/",
  }
}
```

(You can copy and paste the above and make the necessary changes for your system)

This is the base settings required to connect to your MVON# Server. Press Ctrl-S to save your settings.

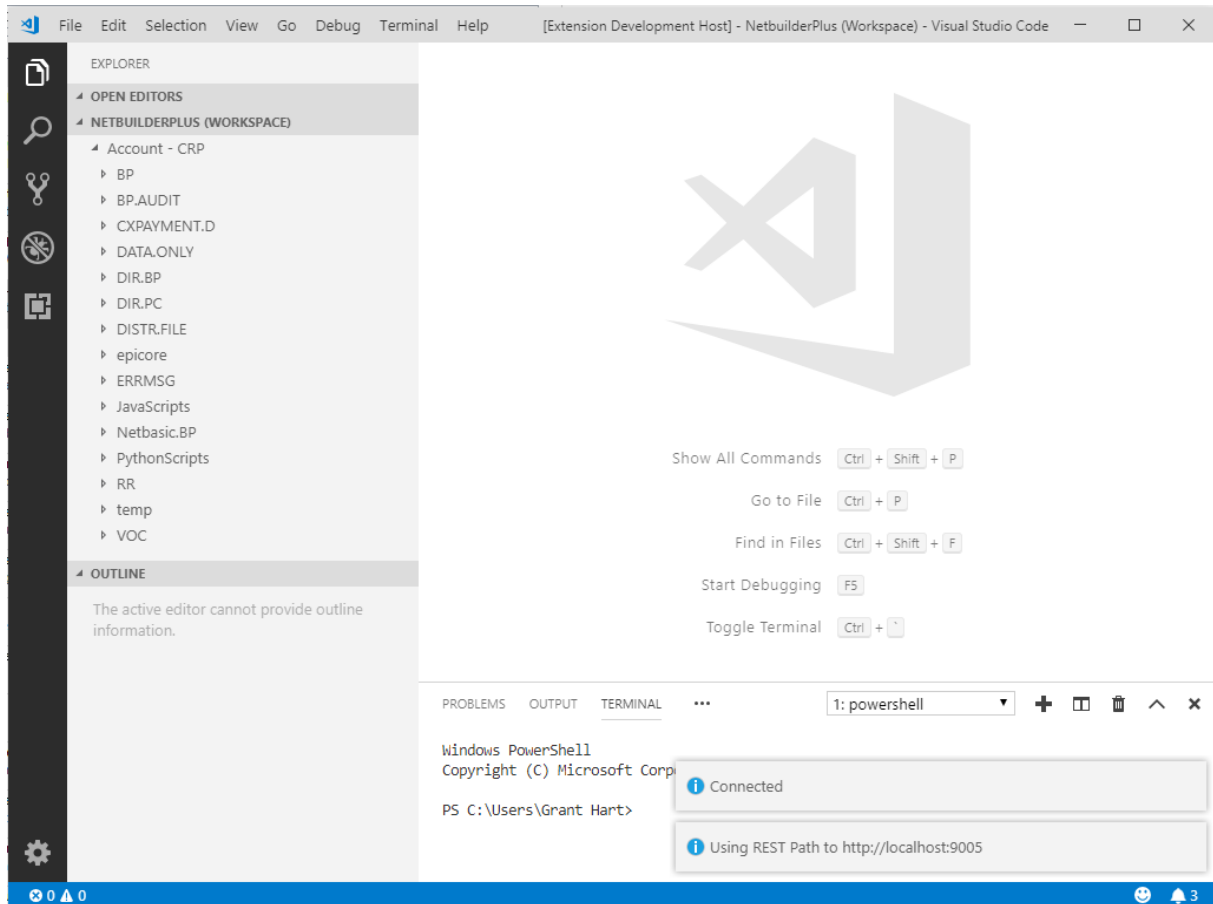
5.1 Testing the connection

We can test to if our connection to MVON# works by Pressing **F1**. VSCODE will prompt you for the command to run. Type **Connect** to display all commands with Connect in it and is displays:



Select **Connect to MVON# REST FS** and the extension will connect to the server and retrieve a list of Directory files from the server.

If the connection is successful, the following 3 messages will appear at the bottom left of the screen.



5.2 Associating Programs with MVON#

Most programming languages have an extension that say what language it is. Python is .py, C# is .cs etc and MV BASIC does not follow this concept.

In order to know that we are editing a BASIC program to enable Syntax highlighting, Intellisense and Linting, we need to tell VSCODE that files in the Workspace are linked to MVON#. This is achieved by adding the following setting to your Workspace settings.

```
{
  "folders":[
    {
      "uri": "RestFS:/",
      "name": "Account - DEMO",
    }
  ],
  "settings": {
    "mvon.RestPath": "http://localhost/mvonrest",
    "files.associations": {"*":"mvon"}
  }
}
```

5.3 Additional MV# Developer Settings

The following settings are available to customise your VSCODE MV# Developer experience.

Setting	Description
mvon.margin	The number of characters to use as a margin when formatting.
mvon.indent	The number of characters to use when indenting code blocks.
mvon.useCamelCase	Use Camelcase for Intellisense keywords.
mvon.ignoreGotoScope	The linter will not highlight goto that jump into the middle of loops.
mvon.formattingEnabled	Set to false to disable code formatting.

6 MV# DEVELOPER FEATURES

The following is a list of features that the extensions offer MV Developers when using VSCODE.

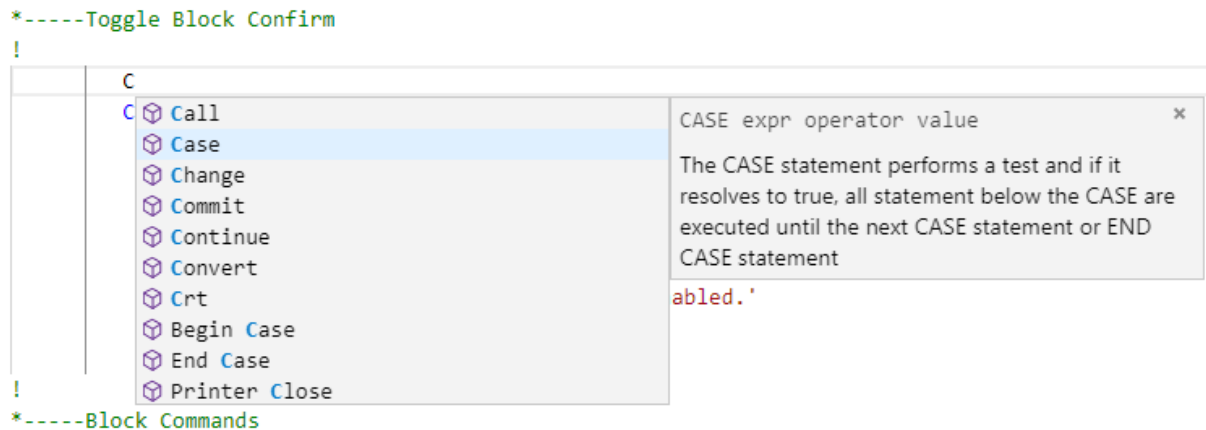
6.1 Syntax Highlighting

```
!
*-----Insert Text
!
      CASE UPCASE(ANS) = 'I' OR UPCASE(ANS) = 'IB' OR UPCASE(ANS) MATCHES
"'I '0X" OR UPCASE(ANS) MATCHES "'IB '0X"
      GOSUB 1030
!
*-----Toggle Block Confirm
!
      CASE UPCASE(ANS) = 'BLOCK'
      IF BLOCK THEN
        BLOCK = FALSE
        CRT 'BLOCK operation verification = disabled.'
      END ELSE
        BLOCK = TRUE
        CRT 'BLOCK operation verification = enabled.'
      END
      GOSUB 1000
```

Code is highlighted based on the current theme selected for VSCODE.

6.2 Intellisense

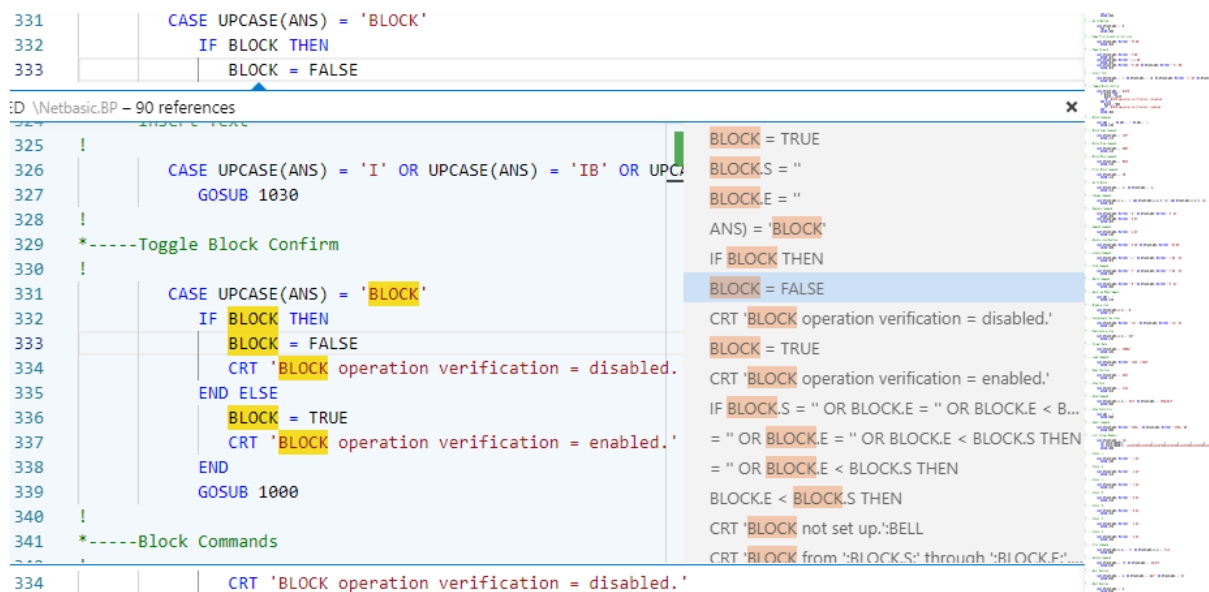
As you type your program, you will be prompted with available statements and functions including the syntax and description.



6.3 Find All References

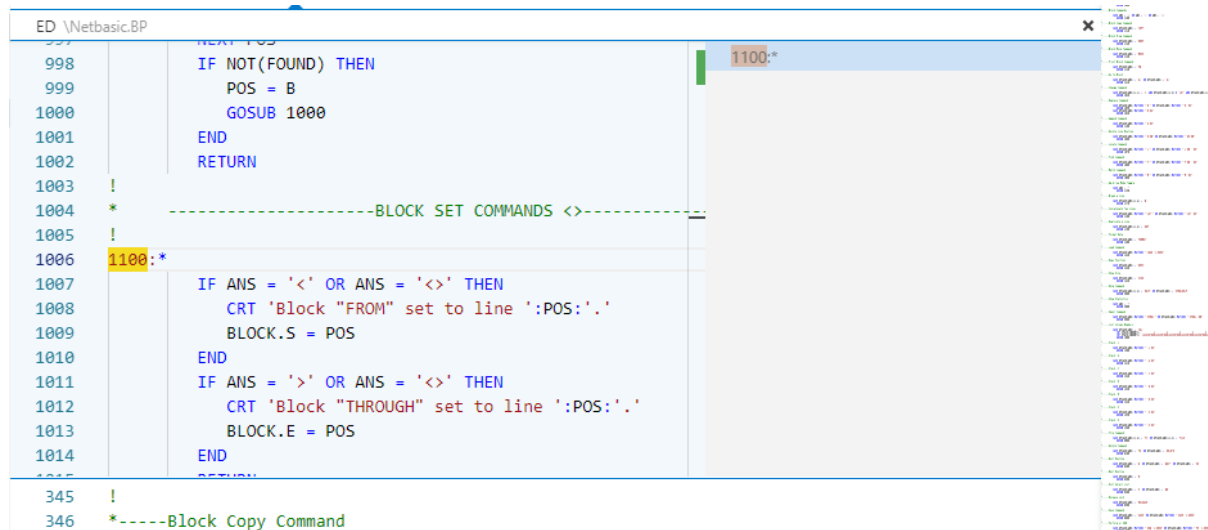
You can find all references to a word in your program by **right clicking** on a word and selecting **Find All References** from the menu.

The display consists of 2 panels, the right containing the line that the word is in and the actual code block is in the left. Clicking on a line in the right panel will take you to the code block.



6.4 Goto/Peek Definition.

If you **right click** on a internal or external subroutine name and select **Peek Definition**, a window appears showing the internal or external subroutine.



If you select **Goto Definition**, the cursor is moved to start of the subroutine.

6.5 Internal Subroutine lookup

Pressing “**Ctrl-space**” after the word GOTO, GOSUB or GO TO, will allow you to select from defined internal subroutines in your program.

```

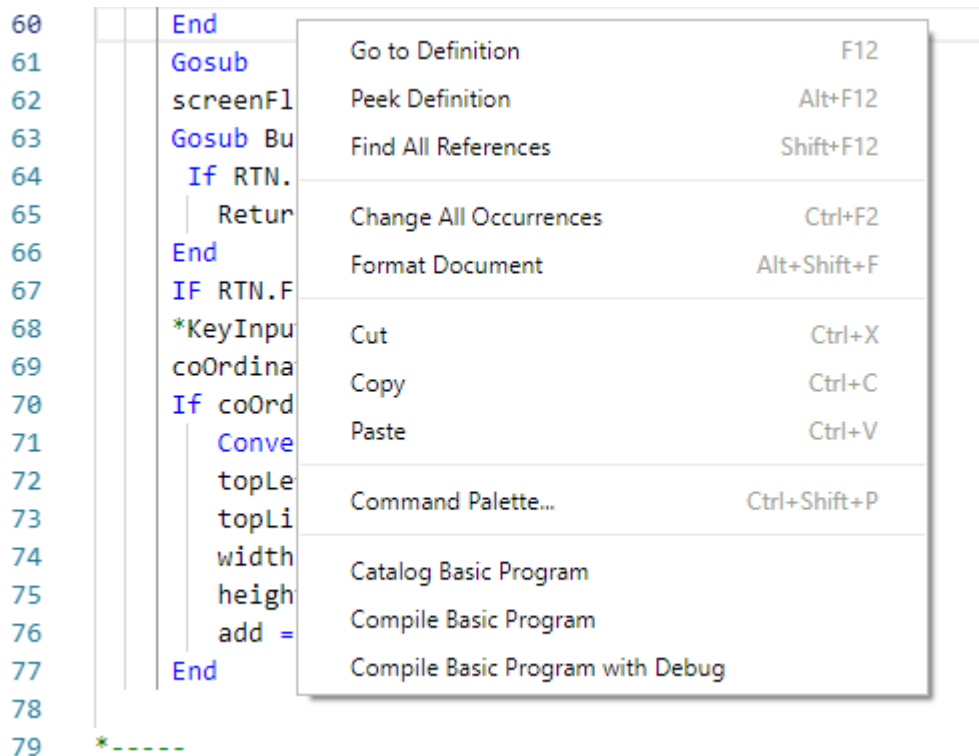
56      If ExitFlag Then
57      |   Answer = "Esc"
58      |   Return
59      |   End
60      End
61      Gosub |
62      screen BuildDefaults
63      Gosub BuildInpValues
64      If RT BuildScreenInput
65      |   Ret CheckMandatory
66      End ClearMvWindow
67      IF RTN DeleteMv
68      *KeyIn DisplayFunctionKeys
69      coOrdi DisplayMvPage
70      If co0 DisplayMvWindow
71      |   Con DisplayScreen
72      |   top EnquiryDisplay
73      |   top FieldsAffectingOthersLab
74      width = coOrdinates<3>
75      height = coOrdinates<4>
76      add = 1
77      End

```

6.6 Compiling and Cataloging your programs.

Right Clicking inside the code window allows you to select 3 options:

1. Catalog Basic Program – catalogs the BASIC program
2. Compile Basic Program – compiles the basic program.
3. Compile Basic Program with Debug – compiles with the debug flag set.



After the option is selected, the results will be displayed in message box at the bottom of the screen. If an error is detected, the editor will place the cursor on the line where error occurs.

6.7 Formatting Programs

Right Clicking and selecting **Format Document**, will format your BASIC program. The formatting is based on the 2 settings, **mvon.indent** and **mvon.margin** that have default values of 3 and 5.

7 CONNECTING TO OTHER MV PLATFORMS

The VSCODE MV# Developer extensions allow you to connect to the most MV platforms and provides all the features described above. The MvonGateway is a windows service that acts as a router to access each of the different MV platforms.

As each platform might require different parameters, a Workspace configuration example is provided for each of the following MV platforms.

1. Universe
2. Unidata
3. OpenQM
4. jBASE
5. D3
6. MvBase

The path to the Gateway Installation media is:

C:\Users\{User Name}\.vscode\extensions\ongroup.mvon-1.85.0\Gateway

It is a standard Windows installer module. Copy the installer to the machine that is going to run the Gateway and install. The Gateway exposes a REST File System that can be accessed via VSCODE. The port that REST pipeline is listening on is defaulted to 9005.

7.1 Universe

```
{
  "folders":[
    {
      "uri": "RestFS:/",
      "name": "Account - Universe",
    }
  ],

  "settings": {
    "mvon.RestPath": "http://localhost:9005/",
    "mvon.UseGateway": true,
    "mvon.RemoteHost": "192.168.137.102",
    "mvon.GatewayType": "Universe",
    "mvon.UserName": "mvon",
    "mvon.Password": "mvon#",
    "mvon.Account": "SUPER-GROUP",
    "files.associations": {"*":"mvon"}
  }
}
```

Setting		Description
mvon.useGateway	true	Indicate that the gateway must be used.
mvon.RestPath	http://localhost:9005	Path to Gateway
mvon.remoteHost	192.168.137.102	The servers IP/Host name that is running the Universe Database.
mvon.gatewayType	Universe	Connecting to a Universe server
mvon.UserName	mvon	The Windows/UNIX user id to log into the server.
mvon.Password	mvon#	The password for the user above.
mvon.Account	SUPER-GROUP	The account name on Universe to connect to. This must be defined in the UV.ACCOUNT file in the UV account.

7.2 Unidata

```
{
  "folders":[
    {
      "uri": "RestFS:/",
      "name": "Account - SUPER-GROUP",
    }
  ],

  "settings": {
    "mvon.UseGateway": true,
    "mvon.RemoteHost": "192.168.137.102",
    "mvon.gatewayType": "Unidata",
    "mvon.UserName": "mvon",
    "mvon.Password": "mvon#",
    "mvon.Account": "SUPER-GROUP",
    "mvon.AccountPath": "/usr/data/SUPER-GROUP",
    "files.associations": {"*":"mvon"}
  }
}
```

Setting		Description
Mvon.UseGateway	true	Indicate that the gateway must be used.
mvon.RestPath	http://localhost:9005	Path to Gateway
mvon.RemoteHost	192.168.137.102	The servers IP/Host name that is running the Unidata Database.
mvon.GatewayType	Unidata	Connecting to a Unidata server
mvon.Password	mvon#	The password for the user above.
Mvon.Account	SUPER-GROUP	A name for this account.
Mvon.AccountPath	/usr/SUPER-GROUP	The path on the Unidata machine to the Unidata account.

7.3 OpenQM

```
{
  "folders":[
    {
      "uri": "RestFS:/",
      "name": "Account - PRC",
    }
  ],

  "settings": {
    "mvon.UseGateway": true,
    "mvon.RemoteHost": "192.168.137.102",
    "mvon.gatewayType": "QM",
    "mvon.UserName": "mvon",
    "mvon.Password": "mvon#",
    "mvon.Account": "PRC",
    "files.associations": {"*":"mvon"}
  }
}
```

Setting		Description
Mvon.useGateway	true	Indicate that the gateway must be used.
mvon.remoteHost	192.168.137.102	The servers IP/Host name that is running the OpenQM Database.
mvon.gatewayType	QM	Connecting to a OpenQM server
mvon.UserName	mvon	The Windows/UNIX user id to log into the server.
mvon.Password	mvon#	The password for the user above.
Mvon.Account	PRC	The account name on the QM server to connect to. This must be defined in the ACCOUNTS file in the QMSYS account.

7.4 jBASE

```
{
  "folders":[
    {
      "uri": "GatewayFS:/",
      "name": "Account - PRC",
    }
  ],

  "settings": {
    "mvon.useGateway": true,
    "mvon.remoteHost": "192.168.137.102",
    "mvon.gatewayType": "jBASE",
    "mvon.gatewayPort": 9004,
    "mvon.gatewayHost": "154.73.73.6",
    "mvon.UserName": "mvon",
    "mvon.Password": "mvon#",
    "mvon.Account": "",
    "files.associations": {"*":"mvon"}
  }
}
```

Setting		Description
Mvon.useGateway	true	Indicate that the gateway must be used.
mvon.remoteHost	192.168.137.102	The servers IP name that is running the jBASE Database.
mvon.gatewayType	jBASE	Connecting to a jBASE server
mvon.gatewayPort	9004	The default port number that the Gateway is listening for connections on.
mvon.UserName	mvon	The Windows/UNIX user id to log into the server.
mvon.Password	mvon#	The password for the user above.
Mvon.Account		This is blank, jBASE uses the default path of the user for the account.

A record in the **MD** called **MVONFILES** can used as a list of available files, alternatively all files are displayed.

7.5 D3

```
{
  "folders":[
    {
      "uri": "GatewayFS:/",
      "name": "Account - DM",
    }
  ],

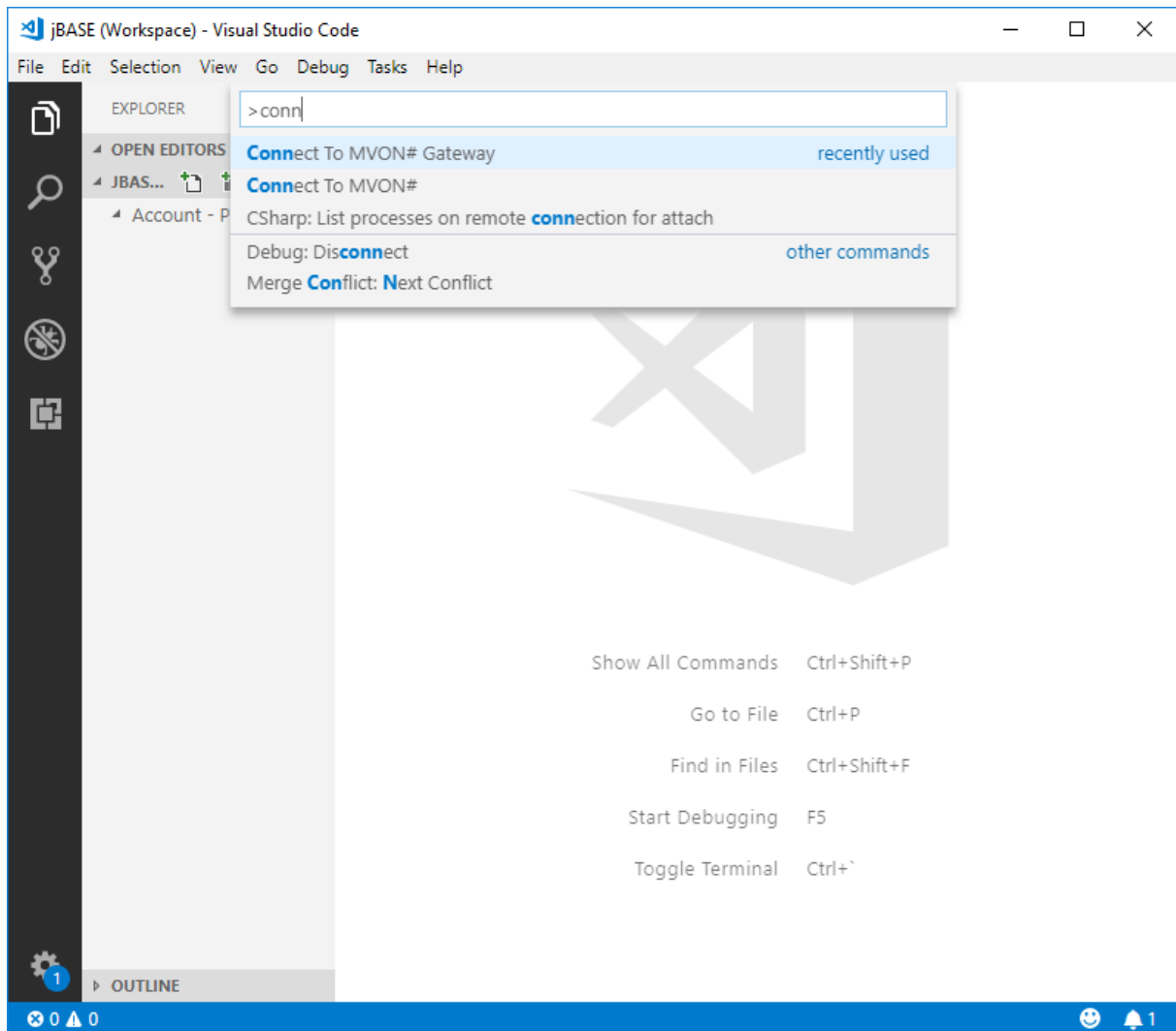
  "settings": {
    "mvon.useGateway": true,
    "mvon.remoteHost": "192.168.137.102",
    "mvon.gatewayType": "D3",
    "mvon.gatewayPort": 9004,
    "mvon.gatewayHost": "154.73.73.6",
    "mvon.UserName": "dm",
    "mvon.AccountPassword": "",
    "mvon.Account": "dm",
    "files.associations": {"*":"mvon"}
  }
}
```

Setting		Description
Mvon.useGateway	true	Indicate that the gateway must be used.
mvon.remoteHost	192.168.137.102	The servers IP name that is running the D3 Database.
mvon.gatewayType	D3	Connecting to a D3 server
mvon.gatewayPort	9004	The default port number that the Gateway is listening for connections on.
mvon.UserName	dm	The D3 User name to log in with
mvon.AccountPassword		Specify the account password if a password is set on the account.
Mvon.Account	dm	The D3 account to connect to.

MSVP must be configured for the above account and the user must have MSVP access. A record in the **MD** called **VSCODEFILES** can be used as a list of available files, alternatively all files are displayed.

7.6 Testing remote connectivity

Once your Workspace is configured for your MV platform, you can connect to your MV Platform by pressing **F1** in VSCODE and type Connect in the search field.



Select the **Connect to MVON# Gateway** option. Once the connection is successful, a list of files will be displayed in the Files pane.

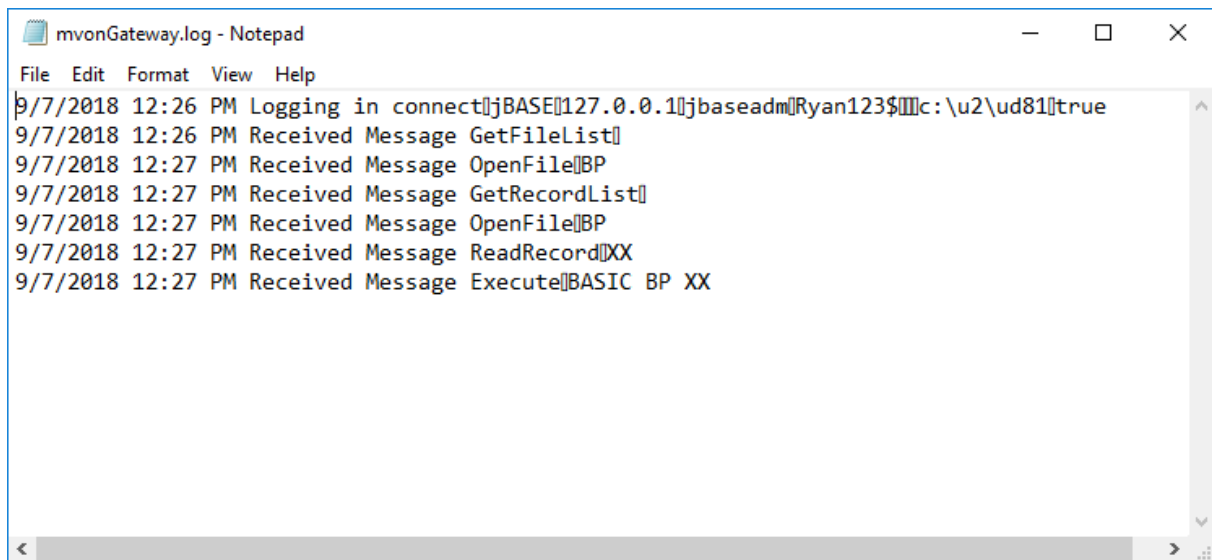
7.7 Debugging remote connection issues

There is an additional parameter that can be specified in your Workspace

`"mvon.gatewayDebug": true`

When this is specified, the MVON# Gateway will write a log of any issues encountered while connecting to your remote MV platform. This can be used to identify any setup issues:

The log file is created in `c:\temp` called **mvonGateway.log**



```
mvonGateway.log - Notepad
File Edit Format View Help
9/7/2018 12:26 PM Logging in connect[]jBASE[]127.0.0.1[]jbaseadm[]Ryan123$[]c:\u2\ud81[]true
9/7/2018 12:26 PM Received Message GetFileList[]
9/7/2018 12:27 PM Received Message OpenFile[]BP
9/7/2018 12:27 PM Received Message GetRecordList[]
9/7/2018 12:27 PM Received Message OpenFile[]BP
9/7/2018 12:27 PM Received Message ReadRecord[]XX
9/7/2018 12:27 PM Received Message Execute[]BASIC BP XX
```

8 MV# DEBUGGER EXTENSION

The MVON# Debugger extension enables powerful visual debugging of your MVON# BASIC programs. This feature is ONLY available for MVON# as MVON# supports the real time debugging protocol required by VSCODE.

In order to debug BASIC programs in VSCODE, they programs need to be compiled using the debug option. You can select **Compile Basic program with Debug** by right clicking in the code editor or alternatively you can compile from the command line using:

BASIC BP XX (D

To enable debugging in VSCODE when need to tell our MVON# environment that the VSCODE debugging is available. An entry VSCODEDEBUG must added to the MVON.CONFIG if the VOC

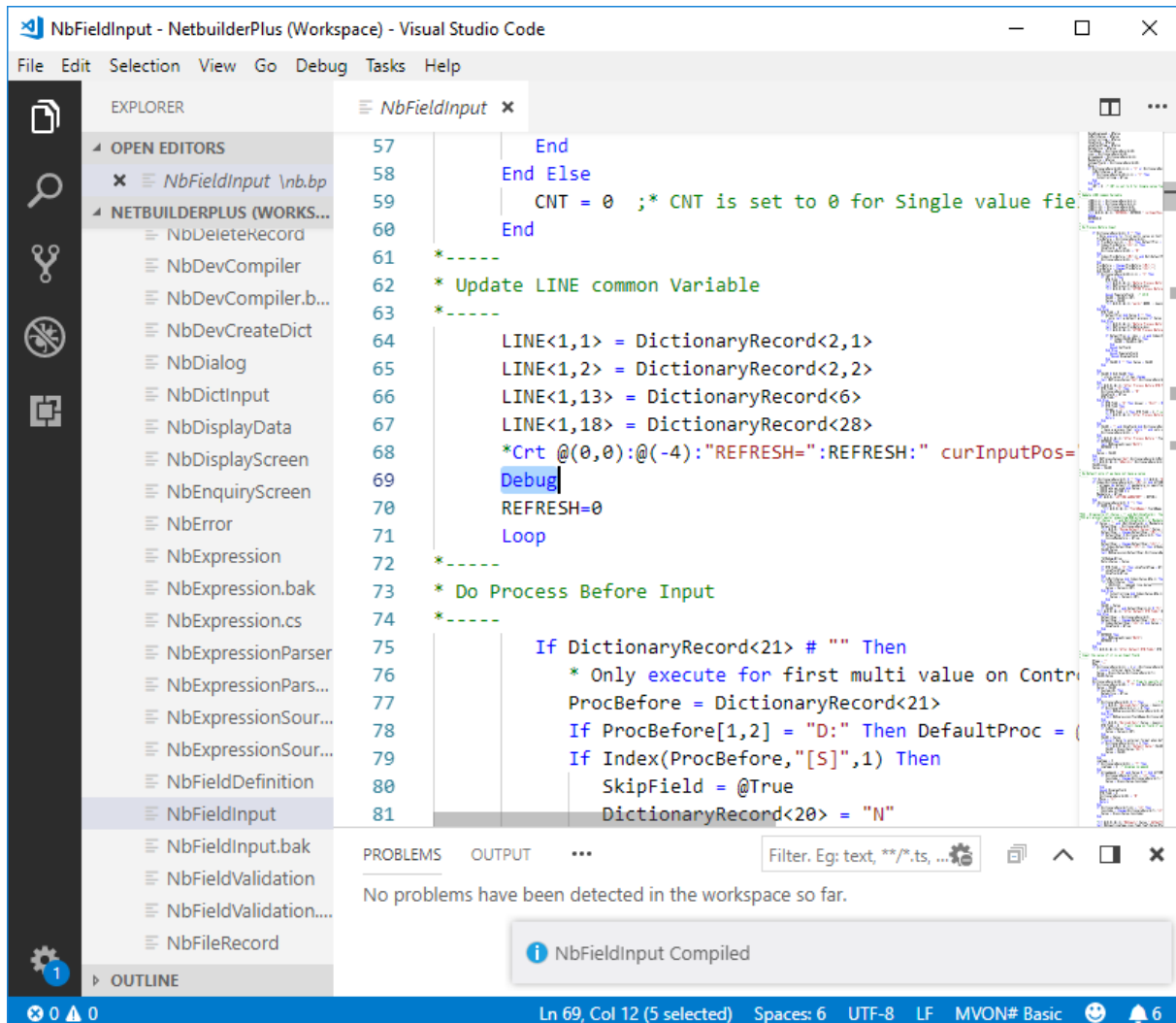
>CT VOC MVON.CONFIG

MVON.CONFIG

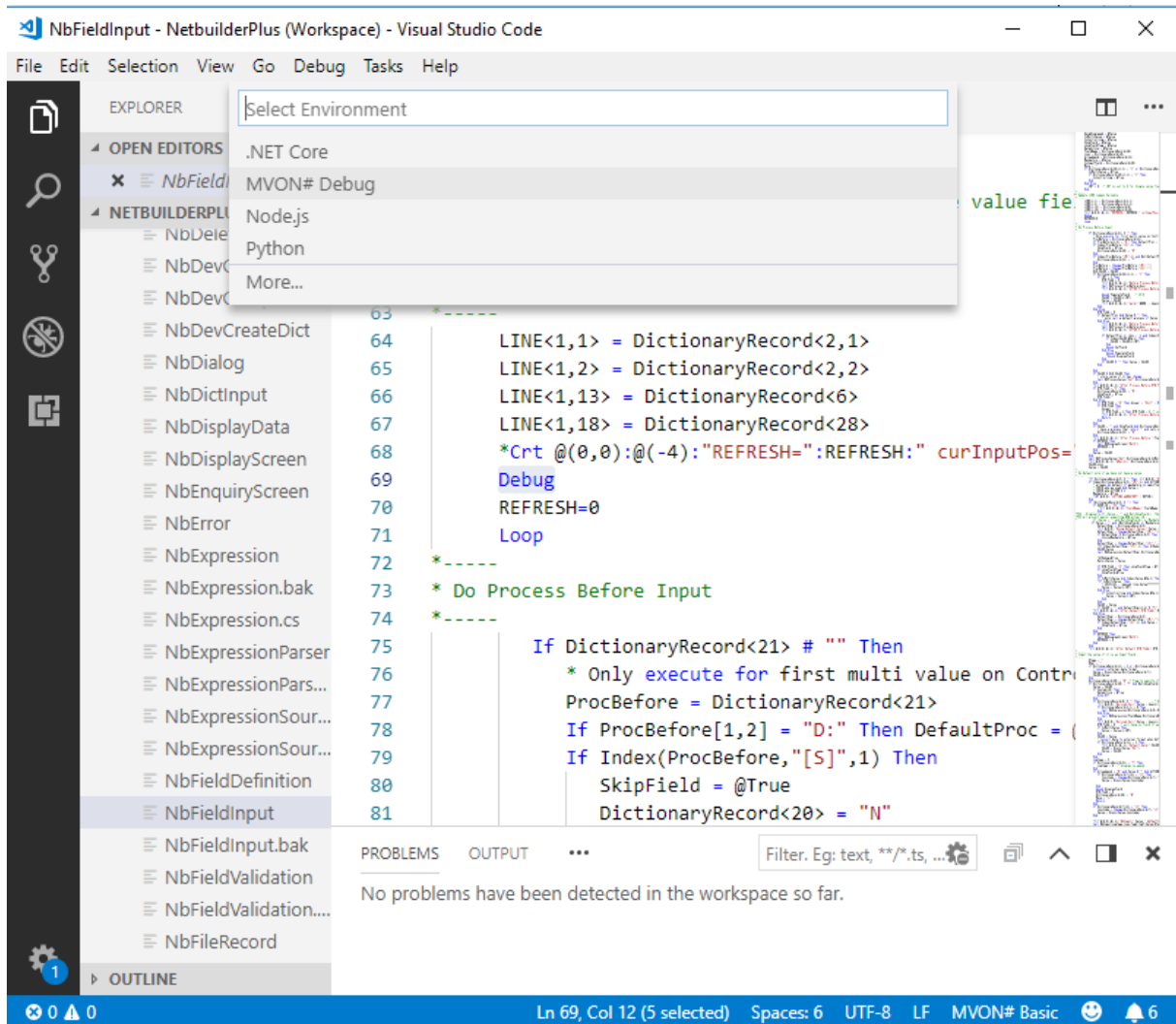
0001 VSCODEDEBUG

8.1 Starting the debugger

You need to first load the subroutine that you wish to debug into the code editor. Place a DEBUG statement just prior to where you want the debugging to start and compile the program with debug.



We start the debugger by pressing F5 in the editor window and selecting MVON# Debug from the drop down list:

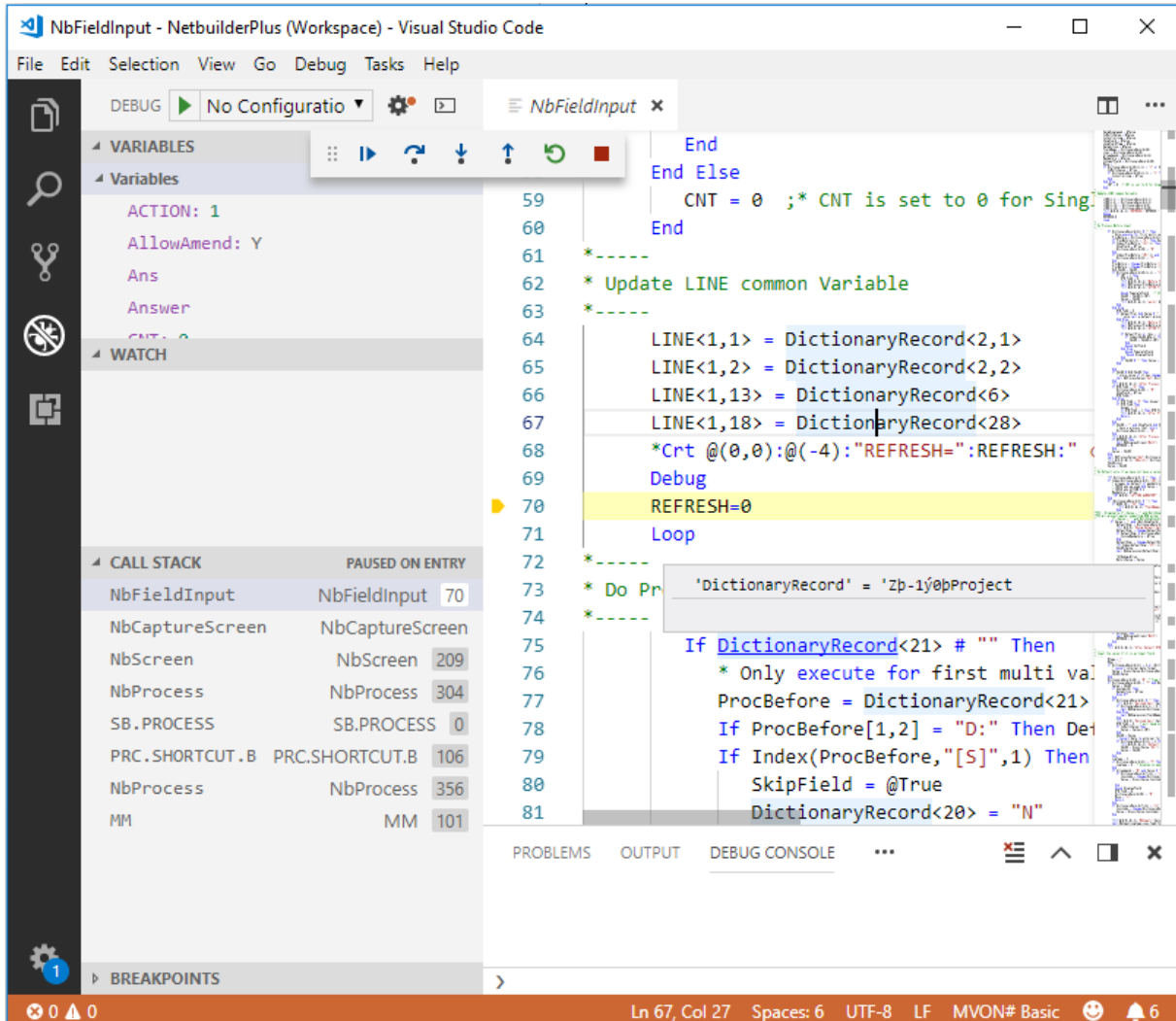


The debug menu bar will appear to indicate the the debugger is now active.

You can now run you BASIC program and when the program gets the DEBUG point , VSCODE will automatically display the debug panel and the current line of code will be highlighted .

8.2 Debugging features

You can step through your code by pressing **F10**, the debugger will move to the next line to be executed. Pressing **F5**, will continue with the program until the next DEBUG statement, breakpoint or the program terminates



If you **hover** your cursor above a variable, the contents of the variable are displayed in a panel.

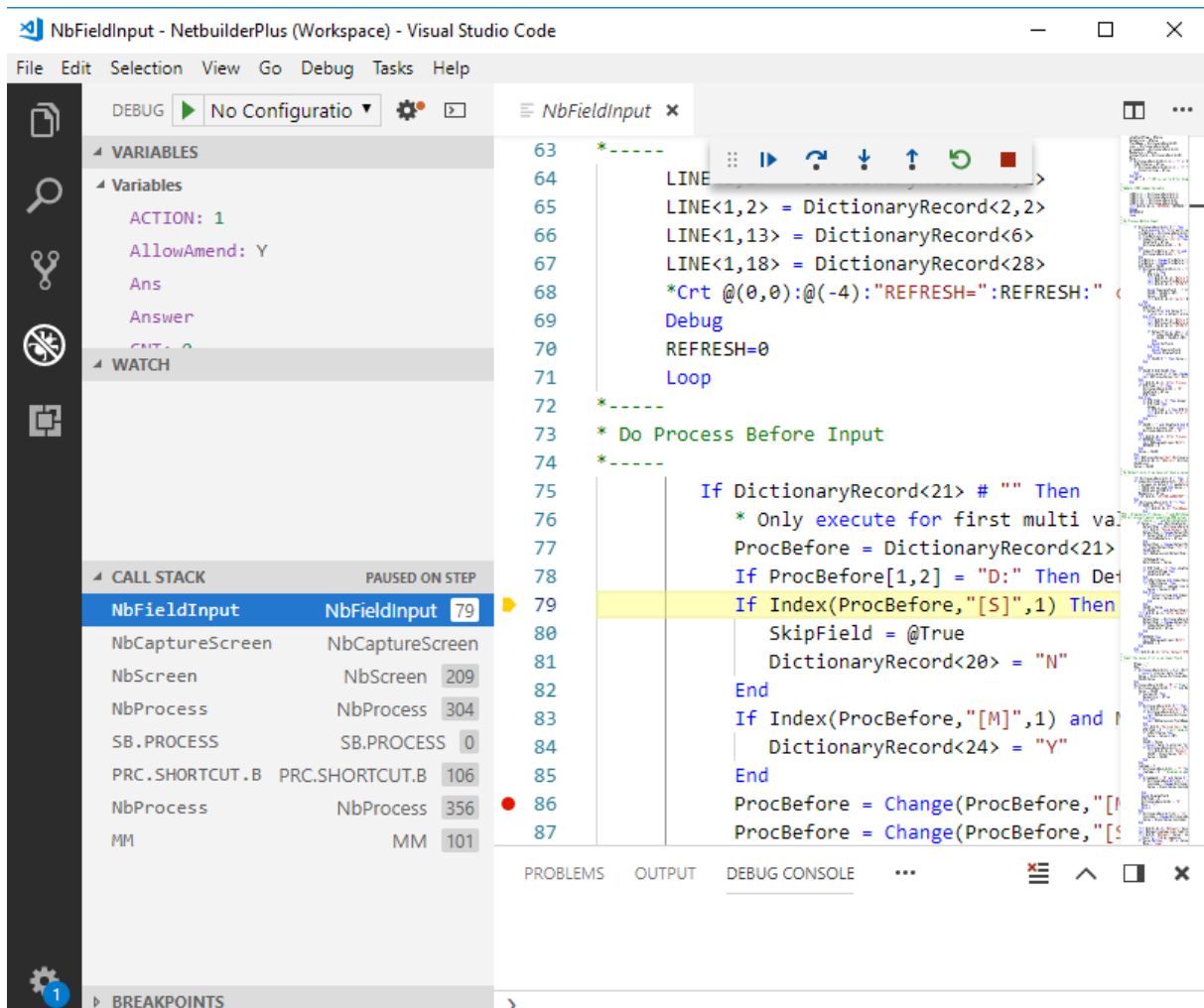
A list of all variables and their contents are displayed in the Variables pane on the left hand side:



These values are automatically updated as you step through your code.

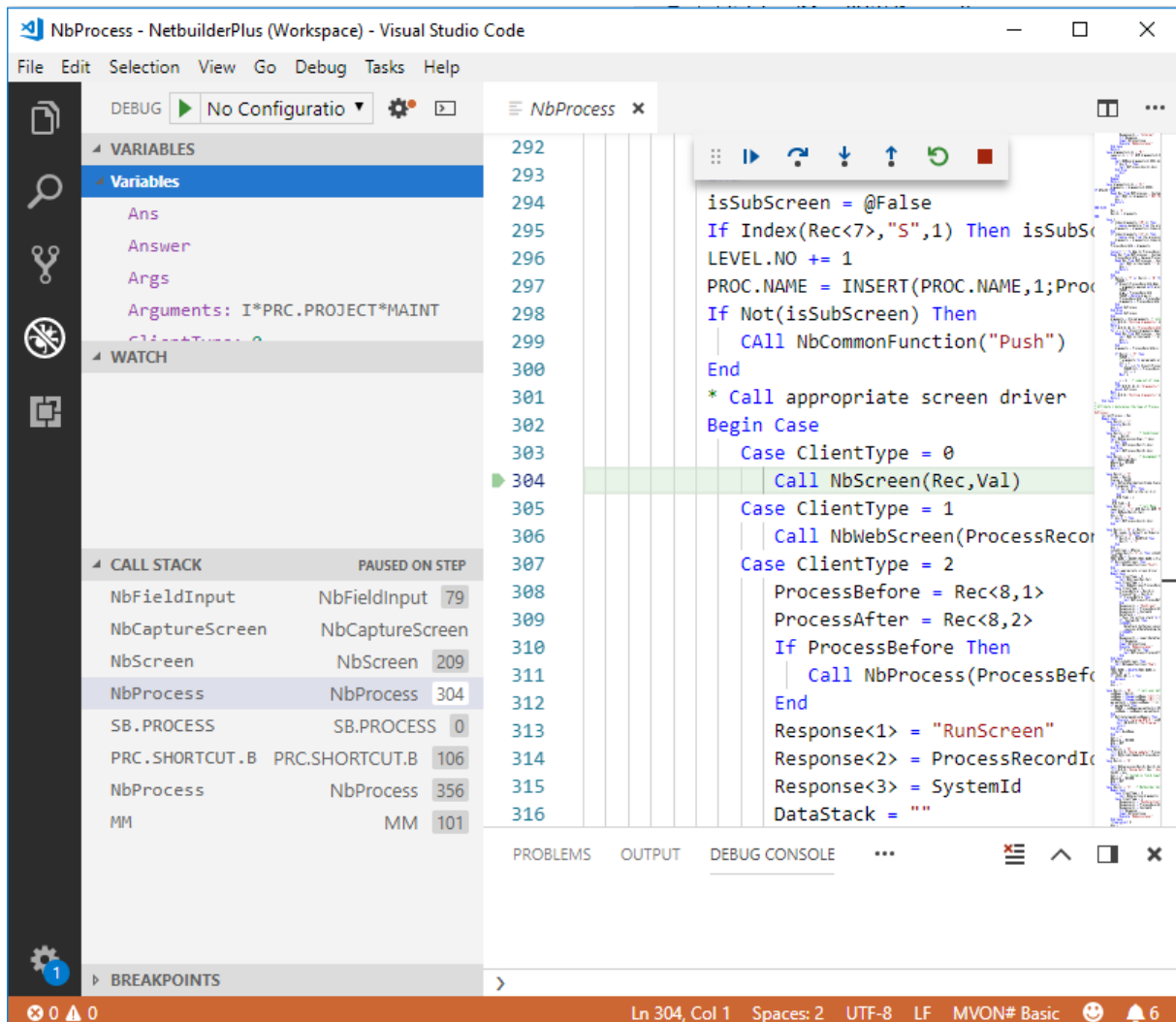
8.3 Call Stack

The call stack is powerful feature of the MVON# debugger. It shows the trace of all the programs called and the line number that they were called on>



If no line number is displayed, the program was not compiled with the DEBUG flag.

The call stack is interactive and if you select an entry from the call stack, the editor automatically loads the program and shows you the line where the program was called from



All variable information for the entire call stack is passed to the debugger so you can interrogate any variable in any of the programs in the call stack.

9 MV# TCL EXTENSION

The MV# TCL Extension gives intellisense, syntax highlighting and syntax checking to a TCL session. It also starts an interactive terminal session where you can execute and see the results of your TCL statements.

Currently the TCL language support is based on MVON#, other MV dialects will be included in the future.

You must have already created the correct settings to connect to your MV platform as described in the above section of the document.

A TCL session is created by opening or creating a document that has a .tcl suffix.

If your target MV platform is not MVON#, you will also need to install the MvonGateway as described in the above section. The Gateway must be the latest that can be found in the path:

C:\Users\{User Name}\.vscode\extensions\ongroup.tcl-0.1.0\Gateway

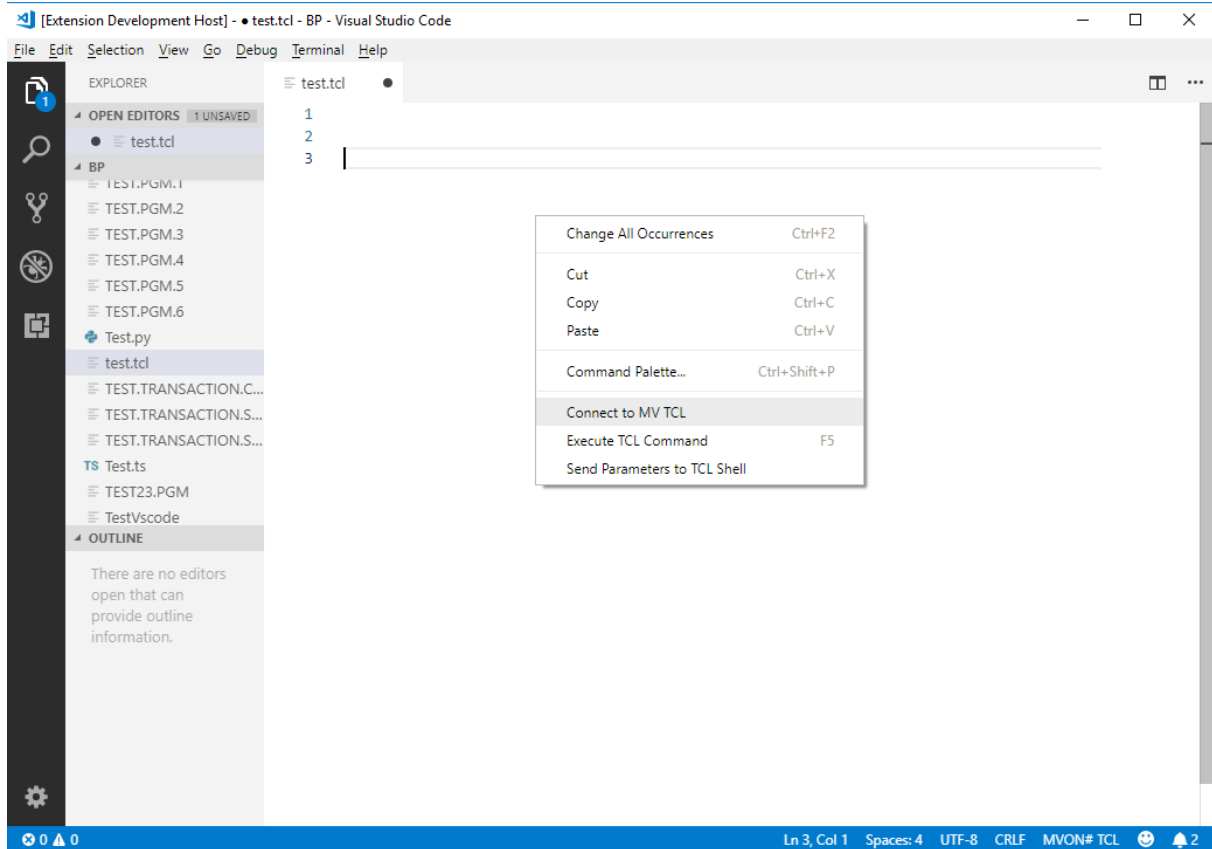
The following additional setting need to be configured for the TCL extension

Setting	Description
tcl.sshCommand	The command to invoke a session on your MV platform. This can be telnet or ssh if your platform supports it.
tcl.parameters	Parameters to pass to the session once it has been established. This could include login credentials and a LOGTO the account you wish the TCL session to be active in.

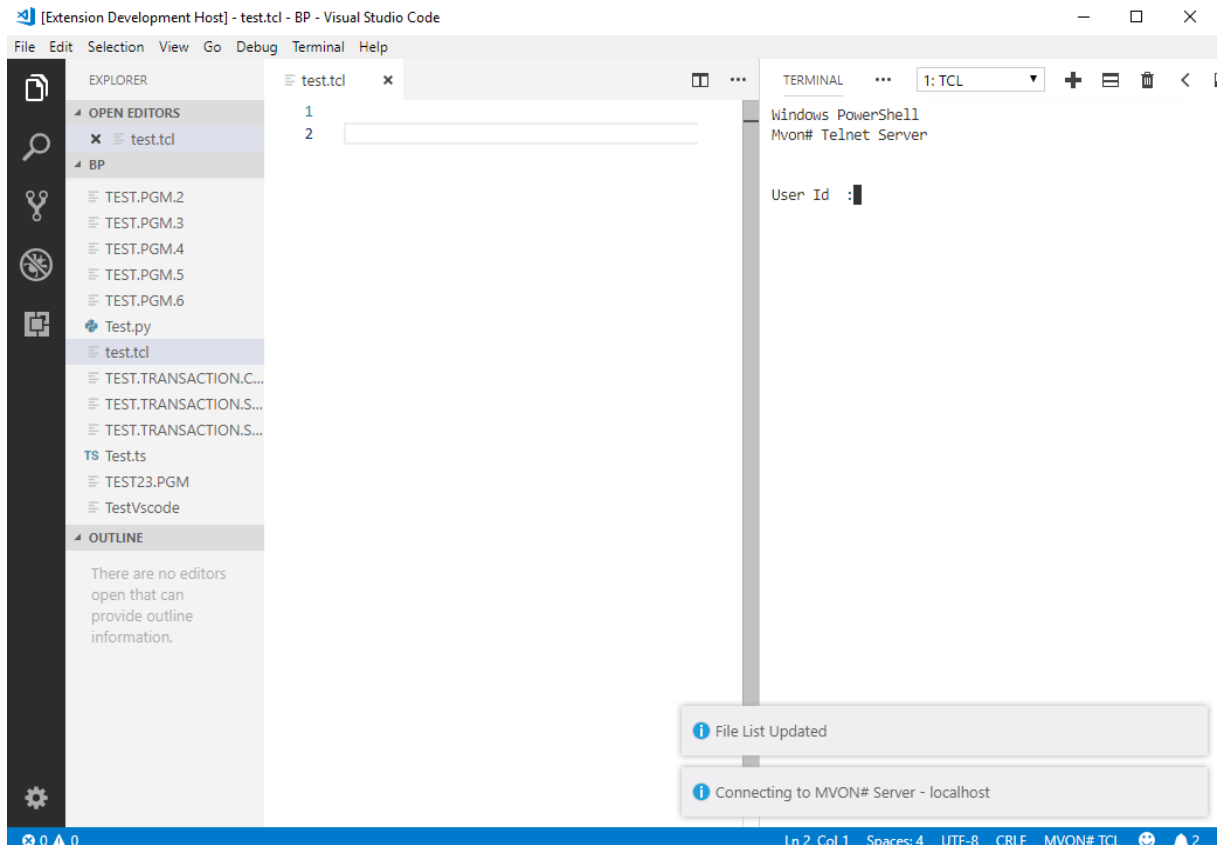
```
"tcl.parameters": [
  "Grant Hart",
  "xxxxxxx$",
  "LOGTO PRC"
],
"tcl.sshCommand": "telnet localhost 2023",
```

9.1 Establishing a TCL session.

Open a document with the suffix of `.tcl`. and right click in the editor window:



Select the **Connect to MV TCL** option from the menu bar. This will open a new terminal session and execute the command you configured in the tcl settings.

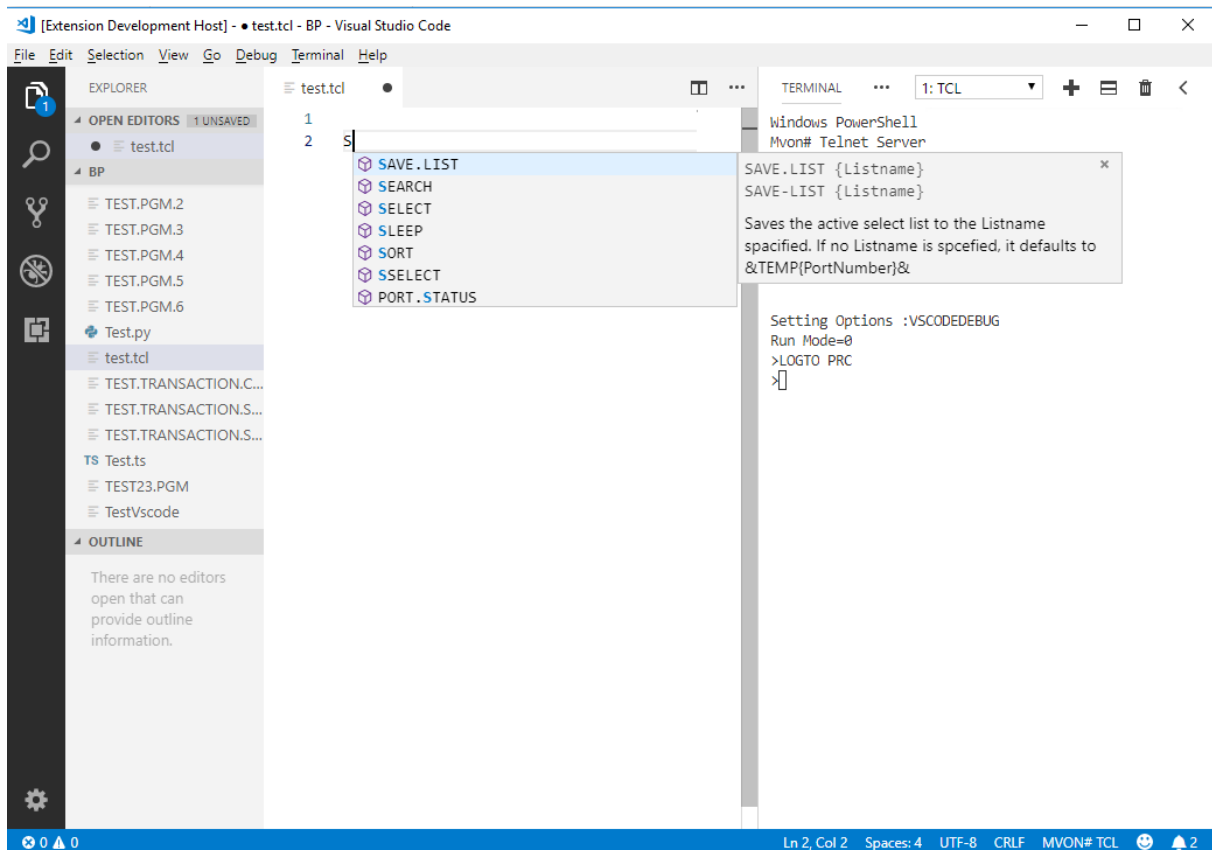


You can right click on the editor window and select Send Parameters to TCL shell from the menu bar to pass the parameters to the session. This automatically connects to you MV platform and loads a list of files that exist in the specified account.

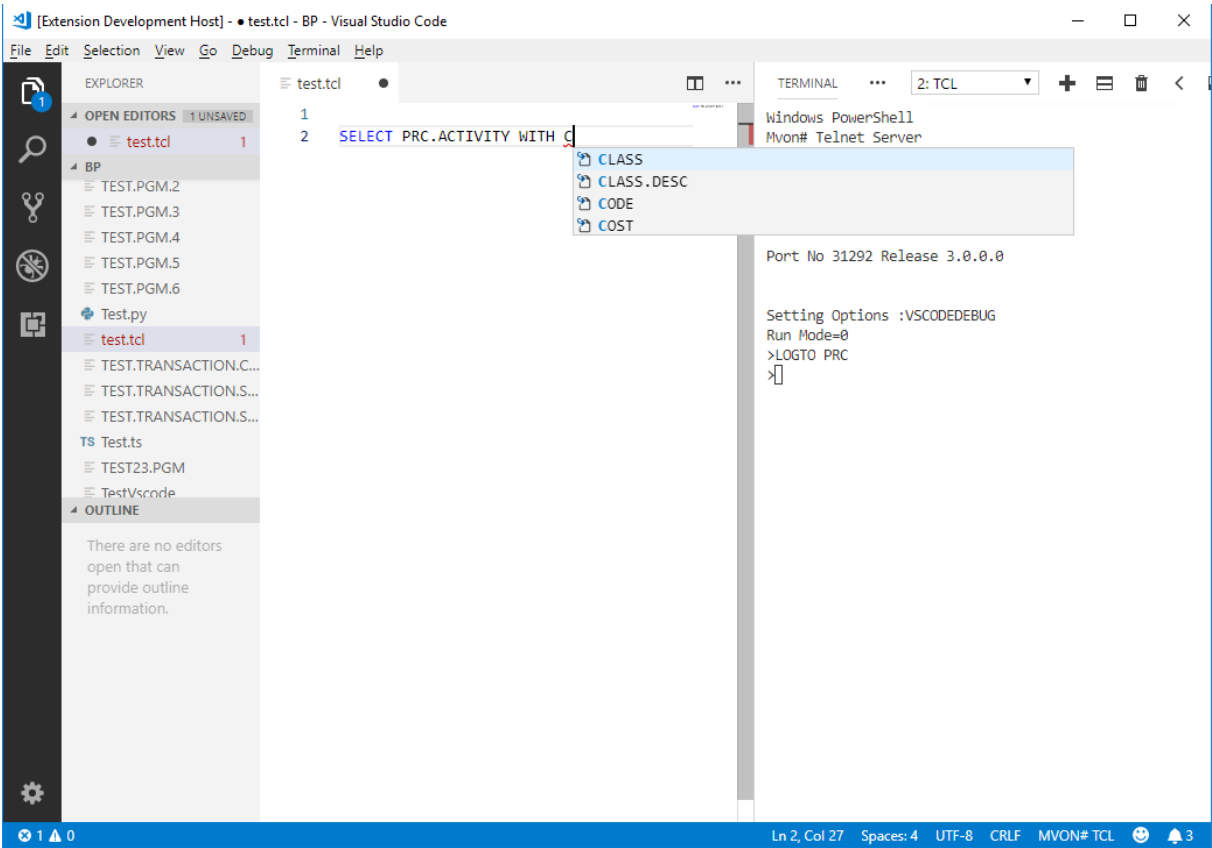
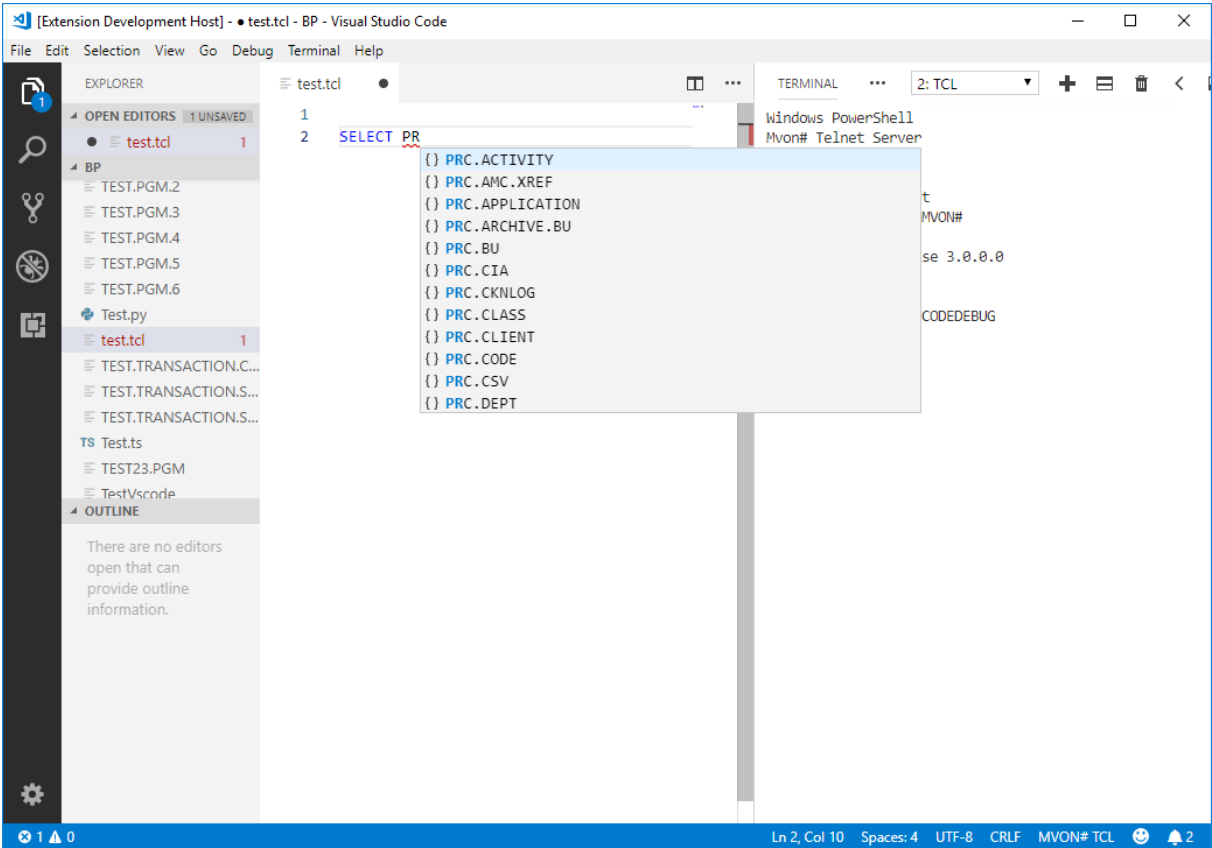
9.2 TCL Features

9.2.1 Intellisense

As you begin typing a list of available TCL commands are displayed for your selection. It also display the syntax that will be required.

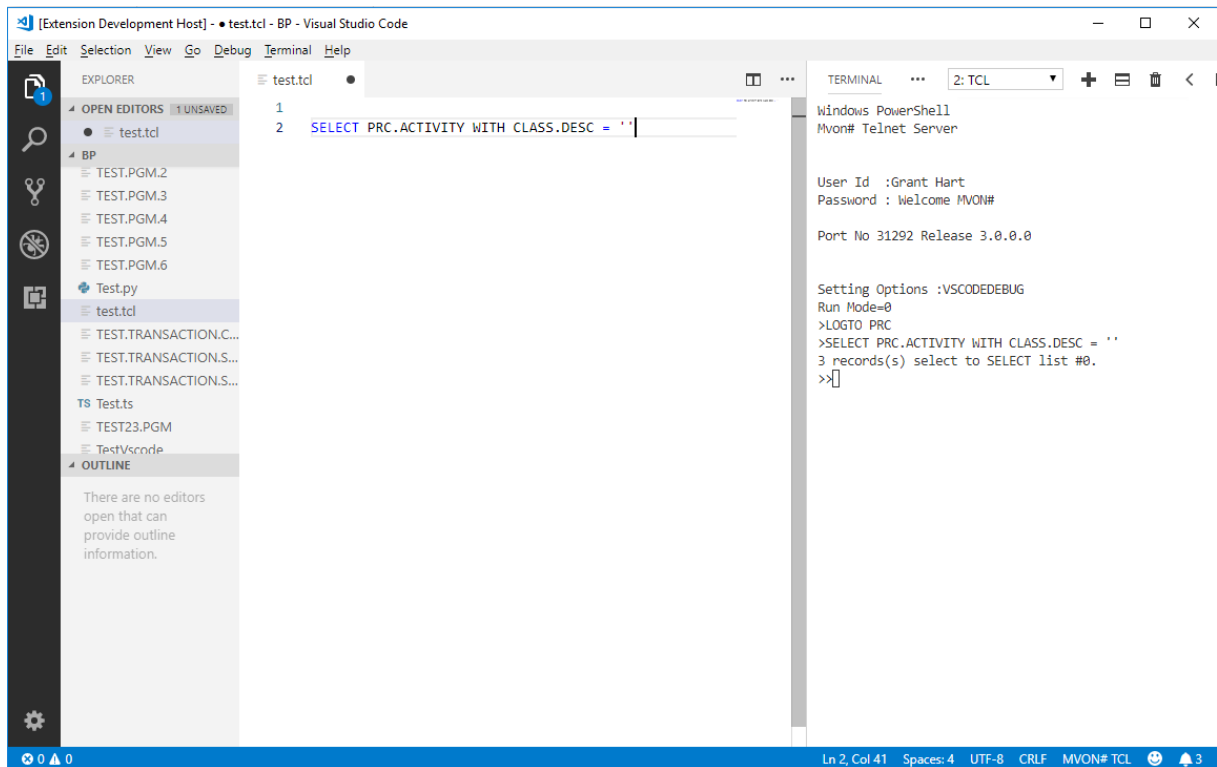


The intellisense takes cognisance of the type of TCL sentence you are creating and will display a list of filenames, dictionary names and item names if your sentence requires them.



9.2.2 Automatically execute script lines

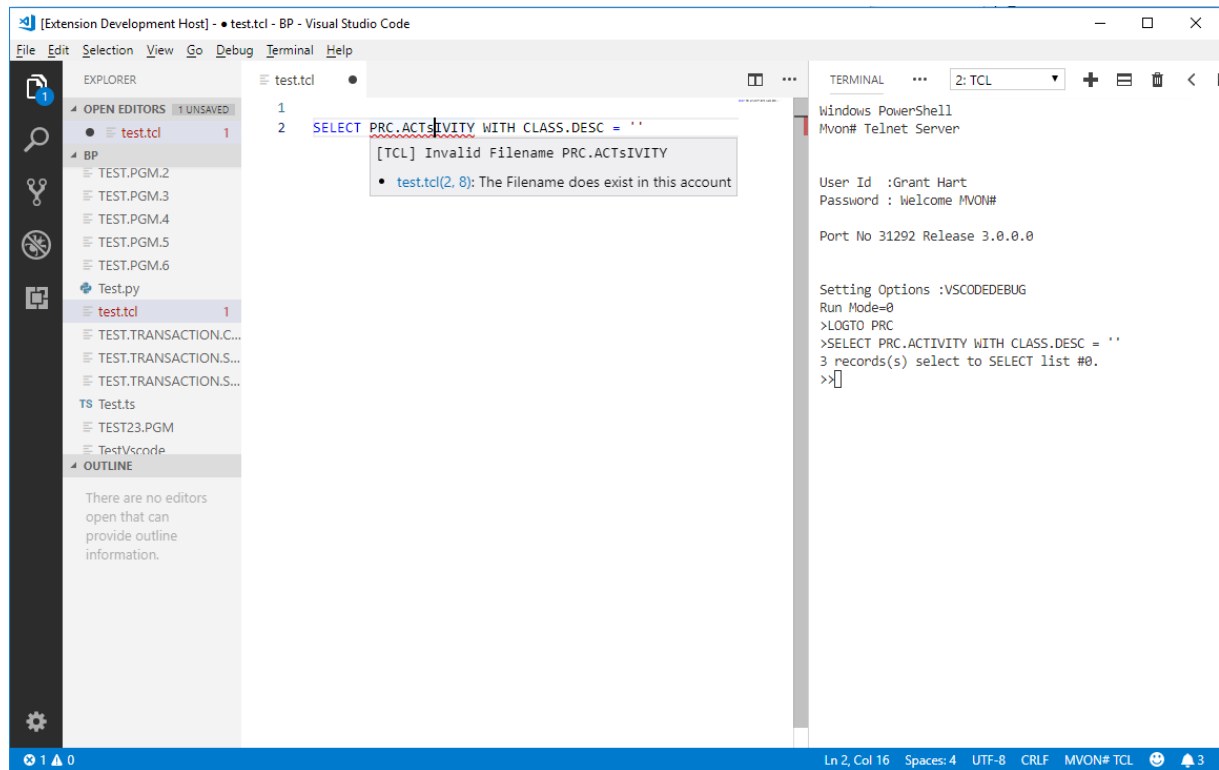
You can execute the current statement by **pressing F5** on the line you wish to execute:



Multiple statements can be executed by select the lines you wish to execute and then **pressing F5**.

9.2.3 Error highlighting

You extension evaluates you statement and highlights any errors it have found like invalid files names and invalid dictionary items.



9.2.4 Dictionary details display

If you hover your mouse over a dictionary item, the details of that dictionary item are displayed.

