**Visual Studio Code**

**MultiValue Extensions**

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](#_Toc13454178)

[1 Introduction 4](#_Toc13454179)

[2 Prerequisites 5](#_Toc13454180)

[3 Installing Visual Studio Code 6](#_Toc13454181)

[4 Configuring Visual Studio Code for MVON#. 9](#_Toc13454182)

[5 Connecting to a MVON# Server 11](#_Toc13454183)

[5.1 Testing the connection 14](#_Toc13454184)

[5.2 Associating Programs with MVON# 16](#_Toc13454185)

[5.3 Additional MV# Developer Settings 16](#_Toc13454186)

[6 MV# Developer Features 17](#_Toc13454187)

[6.1 Syntax Highlighting 17](#_Toc13454188)

[6.2 Intellisense 18](#_Toc13454189)

[6.3 Find All References 18](#_Toc13454190)

[6.4 Goto/Peek Definition. 19](#_Toc13454191)

[6.5 Internal Subroutine lookup 20](#_Toc13454192)

[6.6 Compiling and Cataloging your programs. 21](#_Toc13454193)

[6.7 Formatting Programs 21](#_Toc13454194)

[7 Connecting to other MV Platforms 22](#_Toc13454195)

[7.1 Universe 23](#_Toc13454196)

[7.2 Unidata 24](#_Toc13454197)

[7.3 OpenQM 25](#_Toc13454198)

[7.4 jBASE 26](#_Toc13454199)

[7.5 D3 27](#_Toc13454200)

[7.6 Testing remote connectivity 28](#_Toc13454201)

[7.7 Debugging remote connection issues 29](#_Toc13454202)

[8 MV# Debugger Extension 30](#_Toc13454203)

[8.1 Starting the debugger 31](#_Toc13454204)

[8.2 Debugging features 33](#_Toc13454205)

[8.3 Call Stack 35](#_Toc13454206)

[9 MV# TCL Extension 37](#_Toc13454207)

[9.1 Establishing a TCL session. 38](#_Toc13454208)

[9.2 TCL Features 40](#_Toc13454209)

[9.2.1 Intellisense 40](#_Toc13454210)

[9.2.2 Automatically execute script lines 42](#_Toc13454211)

[9.2.3 Error highlighting 43](#_Toc13454212)

[9.2.4 Dictionary details display 44](#_Toc13454213)

Error! Hyperlink reference not valid.Error! Hyperlink reference not valid.Error! Hyperlink reference not valid.Error! Hyperlink reference not valid.Error! Hyperlink reference not valid.Error! Hyperlink reference not valid.**Error! Hyperlink reference not valid.Error! Hyperlink reference not valid.Error! Hyperlink reference not valid.**Error! Hyperlink reference not valid.**Error! Hyperlink reference not valid.Error! Hyperlink reference not valid.Error! Hyperlink reference not valid.Error! Hyperlink reference not valid.Error! Hyperlink reference not valid.Error! Hyperlink reference not valid.Error! Hyperlink reference not valid.**Error! Hyperlink reference not valid.**Error! Hyperlink reference not valid.Error! Hyperlink reference not valid.Error! Hyperlink reference not valid.Error! Hyperlink reference not valid.Error! Hyperlink reference not valid.Error! Hyperlink reference not valid.Error! Hyperlink reference not valid.**Error! Hyperlink reference not valid.**Error! Hyperlink reference not valid.Error! Hyperlink reference not valid.Error! Hyperlink reference not valid.**Error! Hyperlink reference not valid.**Error! Hyperlink reference not valid.Error! Hyperlink reference not valid.Error! Hyperlink reference not valid.Error! Hyperlink reference not valid.Error! Hyperlink reference not valid.Error! Hyperlink reference not valid.**

Preface

Purpose of this guide

This document describes how to use the MVextensions Visual Studio Code extension as a MultiValue Development Environment. For purposes of this guide MV refers to Pick-style application and database environments mostly known currently as MultiValue.

# 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, this extension was built for the MV BASIC variants of 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 MV files and programs
9. Find all References of a word in current program

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

# Prerequisites

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

1. Windows, Linux or Mac OSX machine.

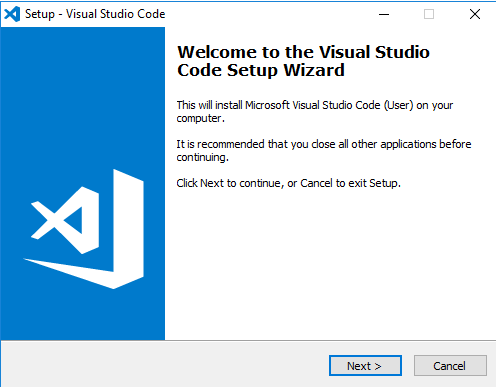
# Installing Visual Studio Code

Visual Sudio Code can be downloaded from the following link:

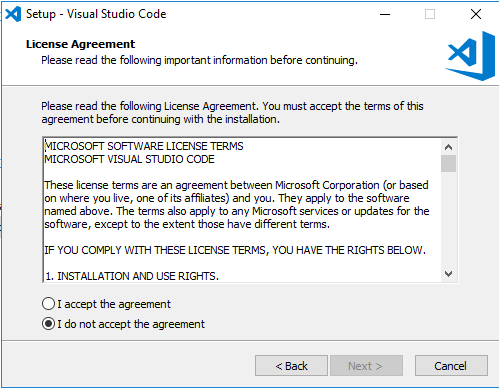
<https://code.visualstudio.com/Download>

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

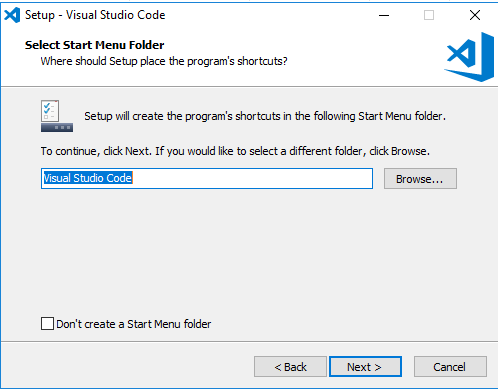
Depending on your Window operating system, run either the 32 bit **VSCodeUserSetup-ia32-1.26.1.exe** or the 64 bit **VSCodeUserSetup-x64-1.26.1.exe**



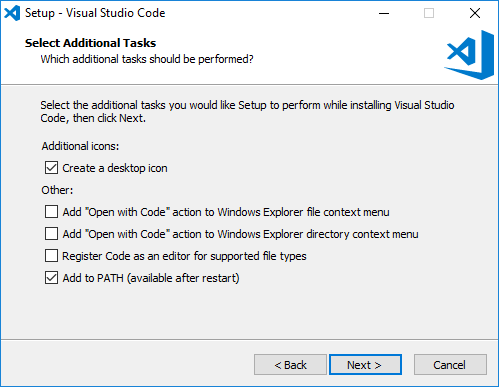
Select Next



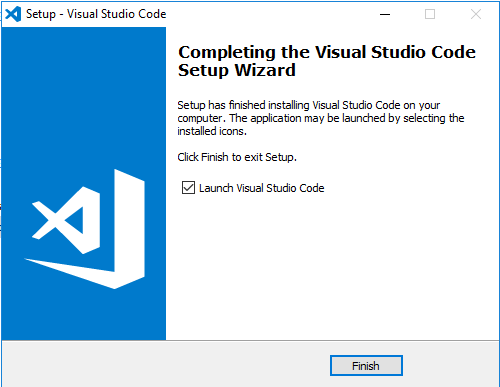
Accept the License Agreement and select **Next**



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



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

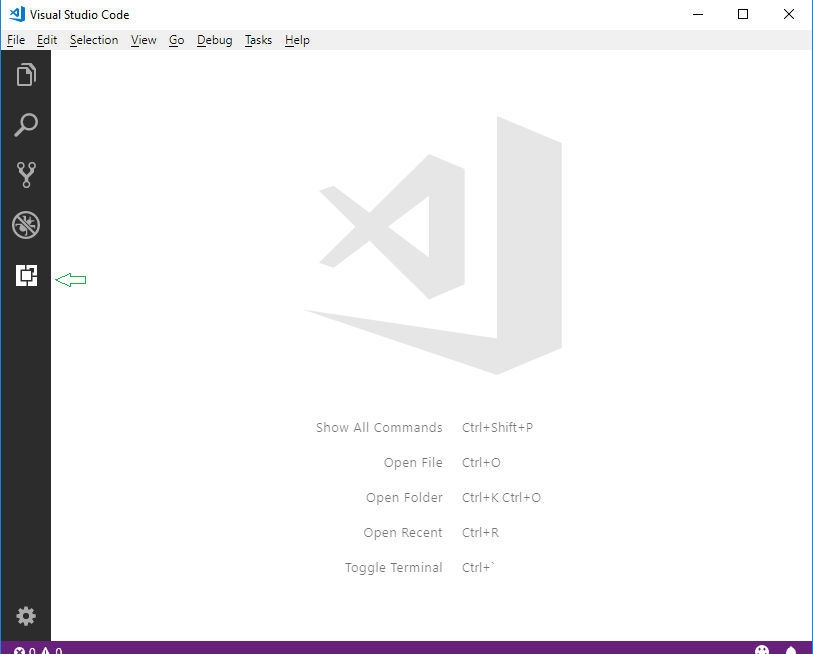


Visual Studio Code is now installed.

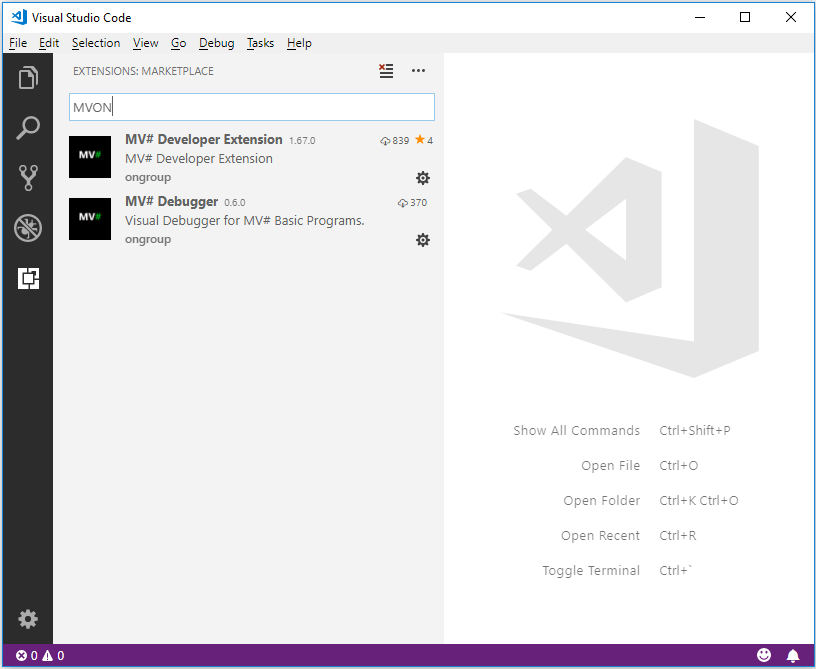
# Configuring Visual Studio Code for MV.

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

Start VSCODE and select the Extensions Button



In the search box, type MV# and press enter:



You can select the Developer Extension by clicking on the download image.

Once the extenion is installed we are ready to start accessing our MV server.

# Connecting to a MultiValue Server

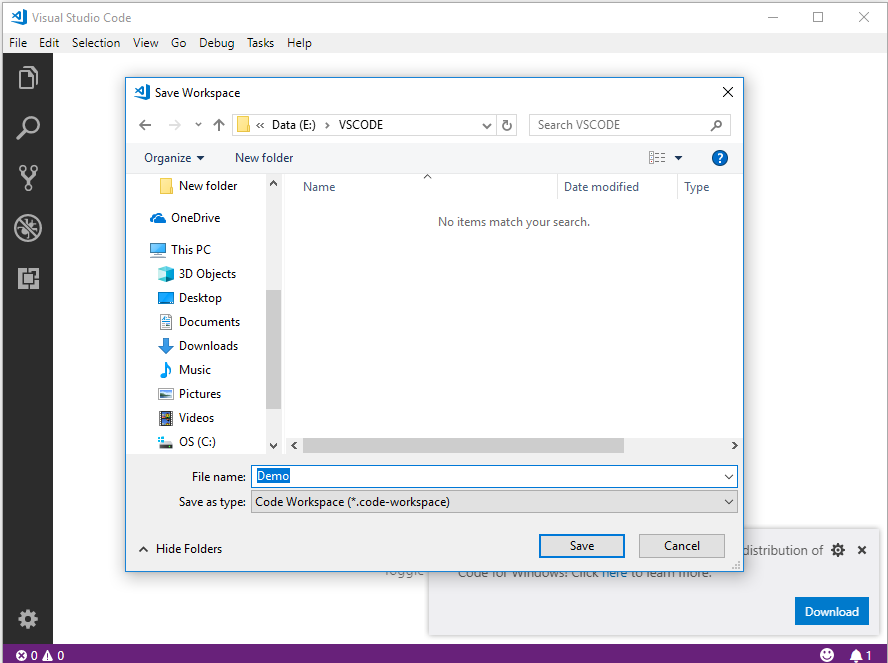
The extenion allows us to connect to MV servers and edit, compile and catalog BASIC programs. In order to do that we first need to configure a VS Code **Workspace** that will contain all the paramters required to connect and login to the remote MV Server.

The simplist method is to create a directory on your machine where you will save the Workspace definitions. If you have multiple servers and/or multiple accounts on each server, you will create multiple Workspace’s that points to a a particular server and account.

In order to connect to your MV Server, the following information is required:

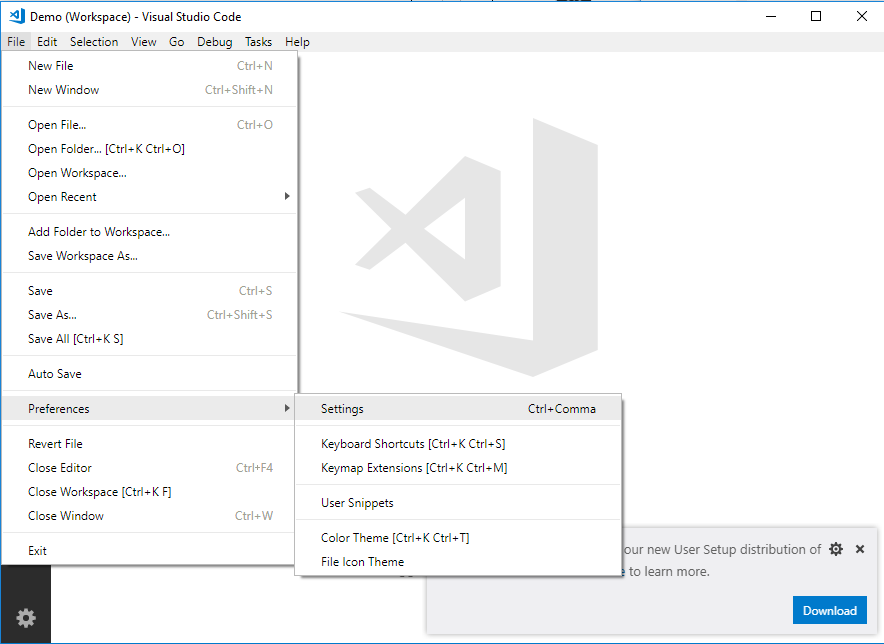
1. Hostname or IP Address of the MV server.
2. User name to login into the server
3. Password for the user above
4. Account name to connect to on the MV Server

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

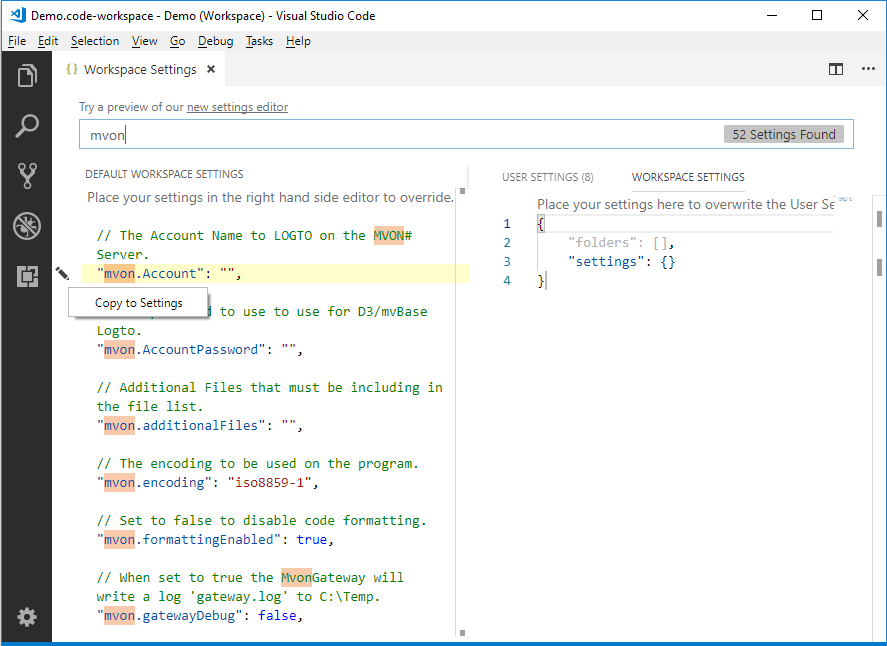


The will create a blank Workspace called Demo that we can now configure to point to our MV server.

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 **mv** in the search box. This will display a list off all the parameters that can be set for the MV Developer extension.



After adding all the parameters to the workspace, you setting should be like this:

{

    "folders":[

        {

            "uri": "RestFS:/",

            "name": "Account - DEMO",

        }

    ],

    "settings": {

        "mvon.RestPath": "http://localhost/mvonrest/",

    }

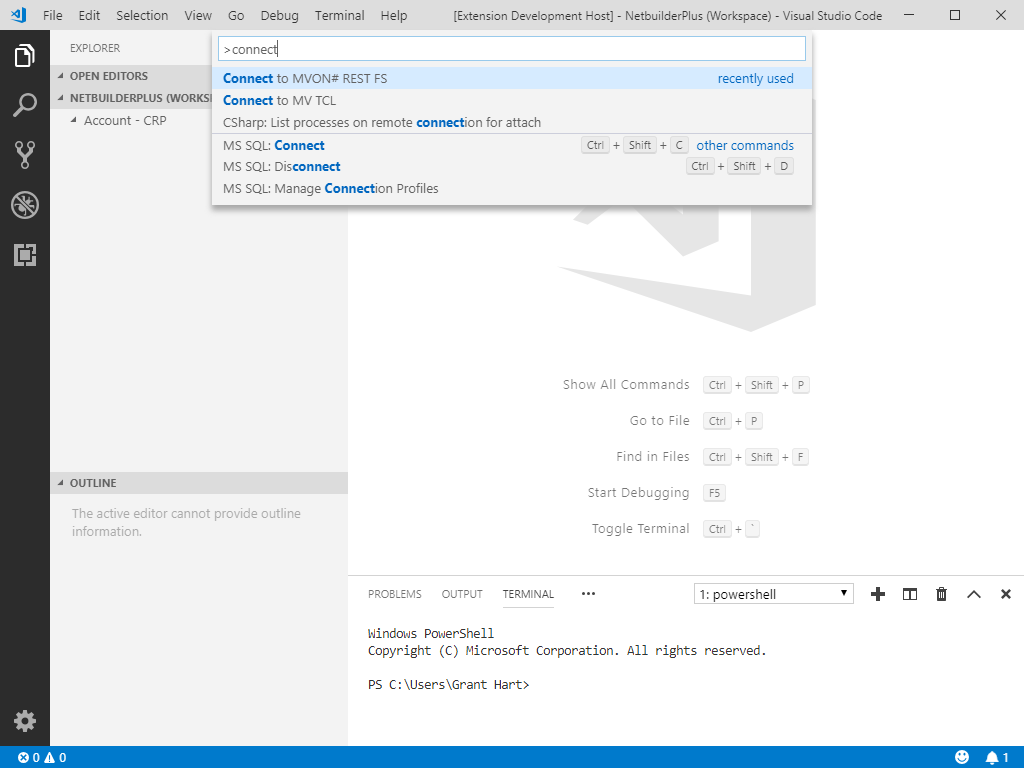
}

(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 Ctl-S to save your settings.

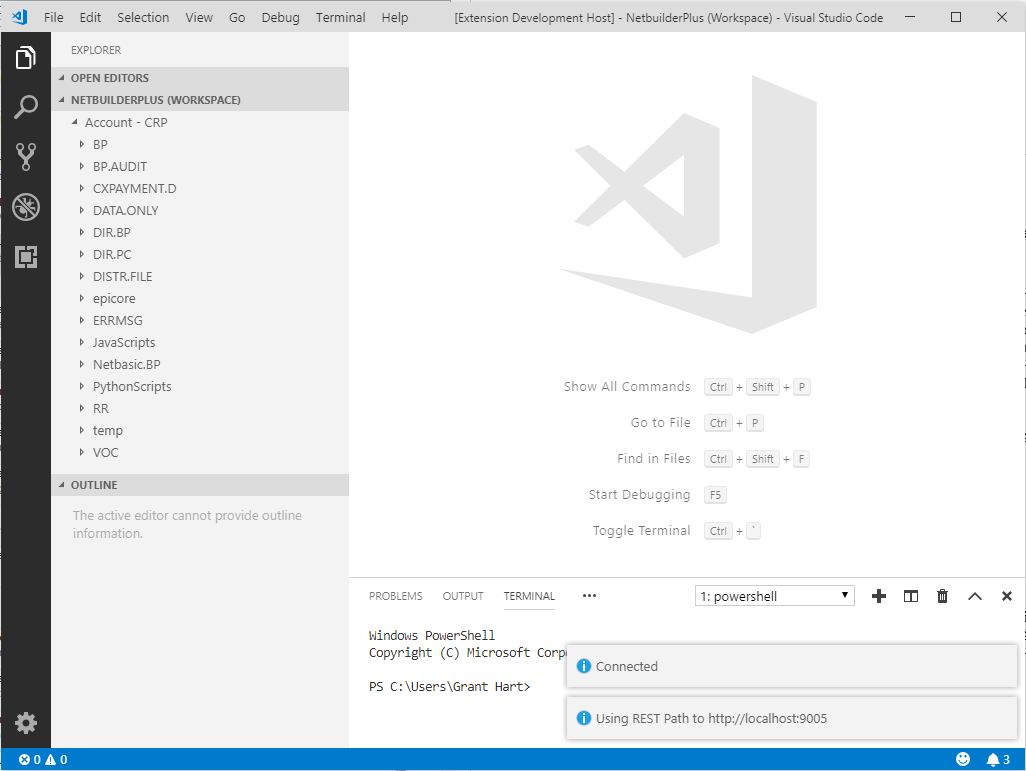
## Testing the connection

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 sucessful, the following 3 messages will appear at the bottom left of the screen.



## Associating Programs with the MVextension

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"}

    }

}

## Additional MV# Developer Settings

The following settings are availabel 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. |
|  |  |

# MV Developer Features

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

## 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.'

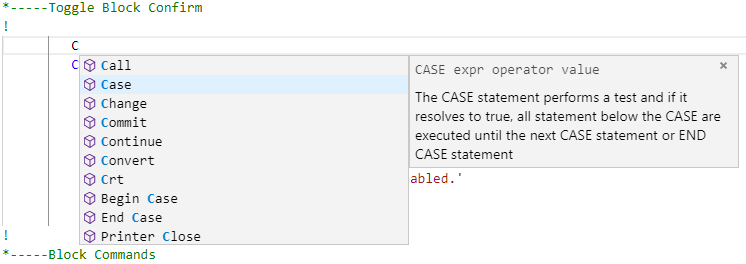
END

GOSUB 1000

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

## Intellisense

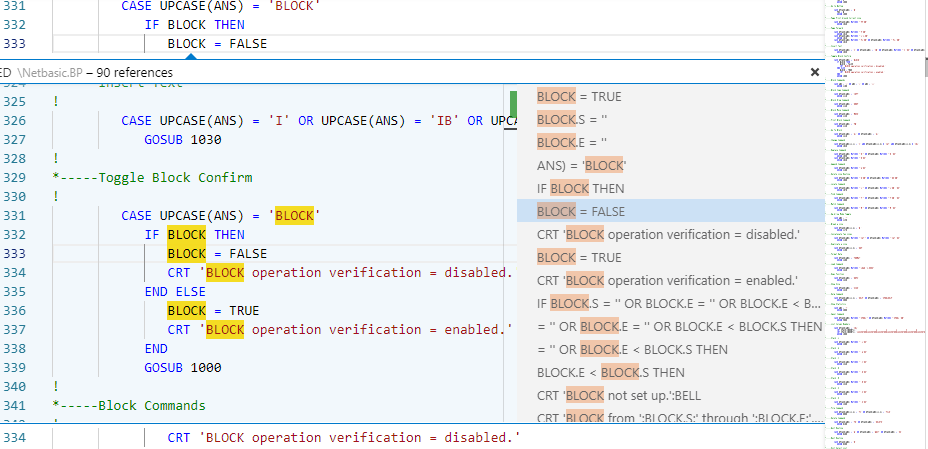
As you type your program, you will be prompted with available statements and functions including the sytax and description.



## Find All References

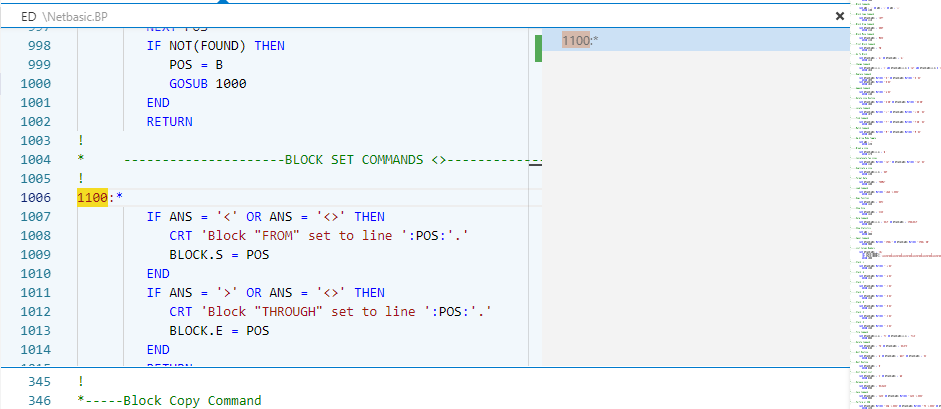
You can find all references to a word in your program by **right clicking** on a word and selecting **Find All Refrences** from the menu.

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.



## Goto/Peek Definition.

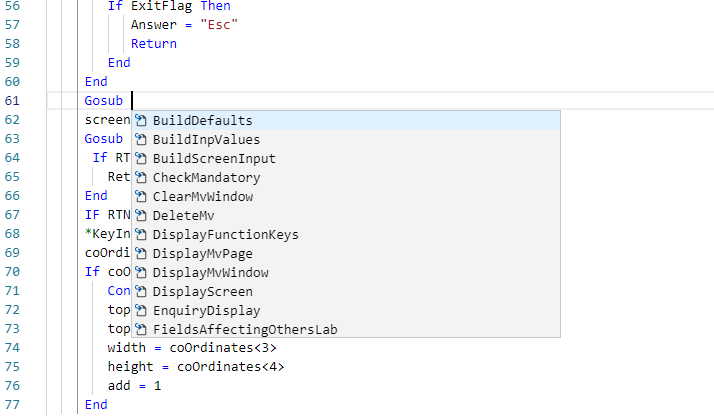
If you **right click** on a internal or external subroutine name and select **Peek Definition**, a window appears showing the internal or external subroutine.



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

## Internal Subroutine lookup

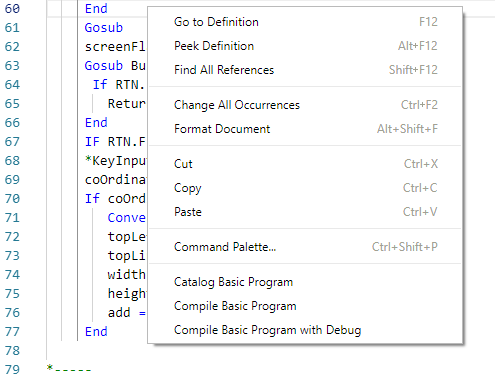
Pressing **“Ctl-space”** after the word GOTO, GOSUB or GO TO, will allow you to select from defined internal subroutines in your program.



## 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 occus.

## 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.

# Connecting to other MV Platforms

The VSCODE MV# Developer extensions all you to connect to the most MV platforms and provides all the features described above. The MvonGetway 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-{Verson Number}\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.

## Universe

{

    "folders":[

{

"uri": "RestFS:/",

"name": "Account - Universe",

}

],

    "settings": {

“mvon.RestPath": “http://localhost:9005/”,

        "mvon.UseGateway": true,

        "mvon.RemoteHost": "192.168.1.10",

        "mvon.GatewayType": "Universe",

"mvon.UserName": "MVextensions",

        "mvon.Password": "Password1",

        "mvon.Account": "MV",

"files.associations": {"\*":"mvon"}

    }

}

|  |  |  |
| --- | --- | --- |
| **Setting** |  | **Description** |
| **mvon.RestPath** | <http://localhost:9005>/ | Path to REST Gateway |
| **mvon.UseGateway** | true | Indicate that the gateway must be used. |
| **mvon.RemoteHost** | 192.168.1.10 | The servers IP/Host name that is running the Universe Database. |
| **mvon.GatewayType** | Universe | Connecting to a Universe server |
| **mvon.UserName** | MVextensions | The Windows/UNIX user id to log into the server. |
| **mvon.Password** | Password1 | The password for the user above. |
| **mvon.Account** | MV | The account name on Universe to connect to. This must be defined in the UV.ACCOUNT file in the UV account. |

## Unidata

{

    "folders":[

{

"uri": "RestFS:/",

"name": "Account – Unidata",

}

],

    "settings": {

“mvon.RestPath”: “http://localhost:9005/”,

        "mvon.UseGateway": true,

        "mvon.RemoteHost": "192.168.1.10",

        "mvon.GatewayType": "Unidata",

"mvon.UserName": "MVextensions",

        "mvon.Password": "Password1",

"mvon.Account": "MV",

        "mvon.AccountPath": "/usr/data/MV",

"files.associations": {"\*":"mvon"}

    }

}

|  |  |  |
| --- | --- | --- |
| **Setting** |  | **Description** |
| **mvon.RestPath** | <http://localhost:9005> | Path to REST Gateway |
| **Mvon.UseGateway** | true | Indicate that the gateway must be used. |
| **mvon.RemoteHost** | 192.168.1.10 | The servers IP/Host name that is running the Unidata Database. |
| **mvon.GatewayType** | Unidata | Connecting to a Unidata server |
| **Mvon.UserName** | MVextensions | The Windows/UNIX user id to log into the server. |
| **mvon.Password** | Password1 | The password for the user above. |
| **Mvon.Account** | MV | A name for this account. |
| **Mvon.AccountPath** | /usr/data/MV | The path on the Unidata machine to the Unidata account. |

## OpenQM

{

    "folders":[

{

"uri": "RestFS:/",

"name": "Account - QM",

}

],

    "settings": {

“mvon.RestPath”: “http://localhost:9005/”,

        "mvon.UseGateway": true,

        "mvon.RemoteHost": "192.168.1.10",

        "mvon.GatewayType": "QM",

"mvon.UserName": "MVextensions",

        "mvon.Password": "Password1",

        "mvon.Account": "MV",

"files.associations": {"\*":"mvon"}

    }

}

|  |  |  |
| --- | --- | --- |
| **Setting** |  | **Description** |
| **mvon.RestPath** | <http://localhost:9005/> | Path to REST Gateway |
| **Mvon.UseGateway** | true | Indicate that the gateway must be used. |
| **mvon.remoteHost** | 192.168.1.10 | The servers IP/Host name that is running the OpenQM Database. |
| **mvon.gatewayType** | QM | Connecting to a OpenQM server |
| **mvon.UserName** | MVextensions | The Windows/UNIX user id to log into the server. |
| **mvon.Password** | Password1 | The password for the user above. |
| **Mvon.Account** | MV | The account name on the QM server to connect to. This must be defined in the ACCOUNTS file in the QMSYS account. |

## jBASE

{

    "folders":[

{

"uri": "GatewayFS:/",

"name": "Account - jBASE",

}

],

    "settings": {

“mvon.RestPath”: “http://localhost:9005/”,

        "mvon.UseGateway": true,

        "mvon.RemoteHost": "192.168.1.10",

        "mvon.GatewayType": "jBASE",

"mvon.UserName": "MVextensions",

        "mvon.Password": "Password1",

        "mvon.Account": "",

"files.associations": {"\*":"mvon"}

    }

}

|  |  |  |
| --- | --- | --- |
| **Setting** |  | **Description** |
| **mvon.RestPath** | <http://localhost:9005/> | Path to REST Gateway |
| **Mvon.UseGateway** | true | Indicates 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.UserName** | MVextensions | The Windows/UNIX user id to log into the server. |
| **mvon.Password** | Password1 | 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.

## D3

{

    "folders":[

{

"uri": "GatewayFS:/",

"name": "Account – D3",

}

],

    "settings": {

“mvon.RestPath”: “http://localhost:9005/”,

        "mvon.UseGateway": true,

        "mvon.RemoteHost": "192.168.1.10",

        "mvon.GatewayType": "D3",

"mvon.UserName": "dm",

        "mvon.AccountPassword": "",

        "mvon.Account": "dm",

"files.associations": {"\*":"mvon"}

    }

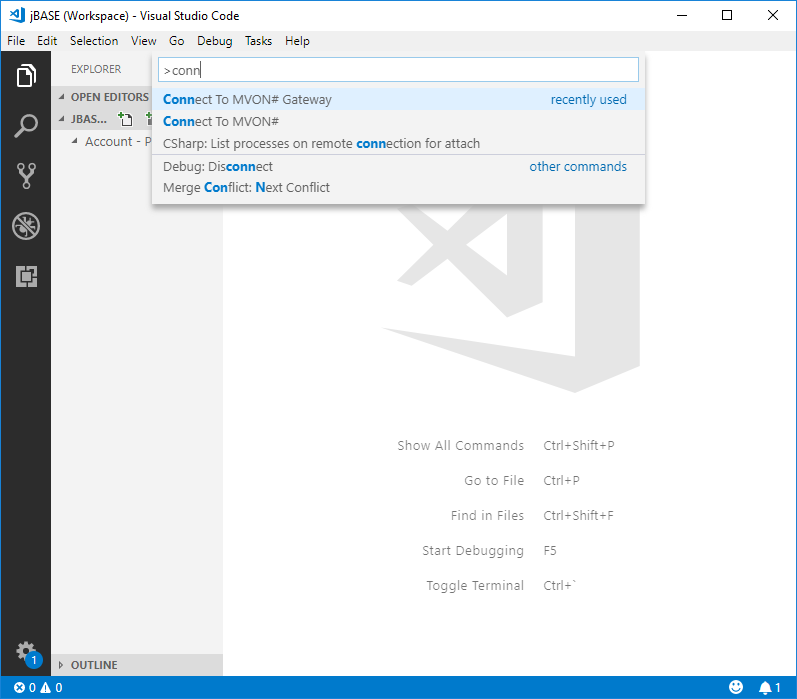
}

|  |  |  |
| --- | --- | --- |
| **Setting** |  | **Description** |
| **mvon.RestPath** | http://localhost:9005/ | Path to the REST Gateway |
| **mvon.UseGateway** | true | Indicates 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.

## 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.

## 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**

