Announcement
Dmitry Maslennikov · Sep 9, 2019

Open Exchange

VSCode ObjectScript release 0.7.13

Hi all, it's finally time for the next release of VSCode ObjectScript extension. So what's new in this release.

- Debugging support, for classes, routines and attach to a running process
- Files in Server Explorer now can be edited
- Added more details about connection errors
- Improvements in Server Explorer build tree
- Fixed memory leak when exporting large amount of files
- Server view can be opened in explorer as virtual file system with schema ìsfs://`
- Option to suppress popup information message about successful compile, ("objectscript.suppressCompileMessages": true)
- Export, addCategory setting have more flexibility in naming category for exported items
- Formatting for commands and functions, as Word, UPPER or lower
- Some improvements in syntax highlighting
- Some other small fixes

Well, let's go to details.

Debugging

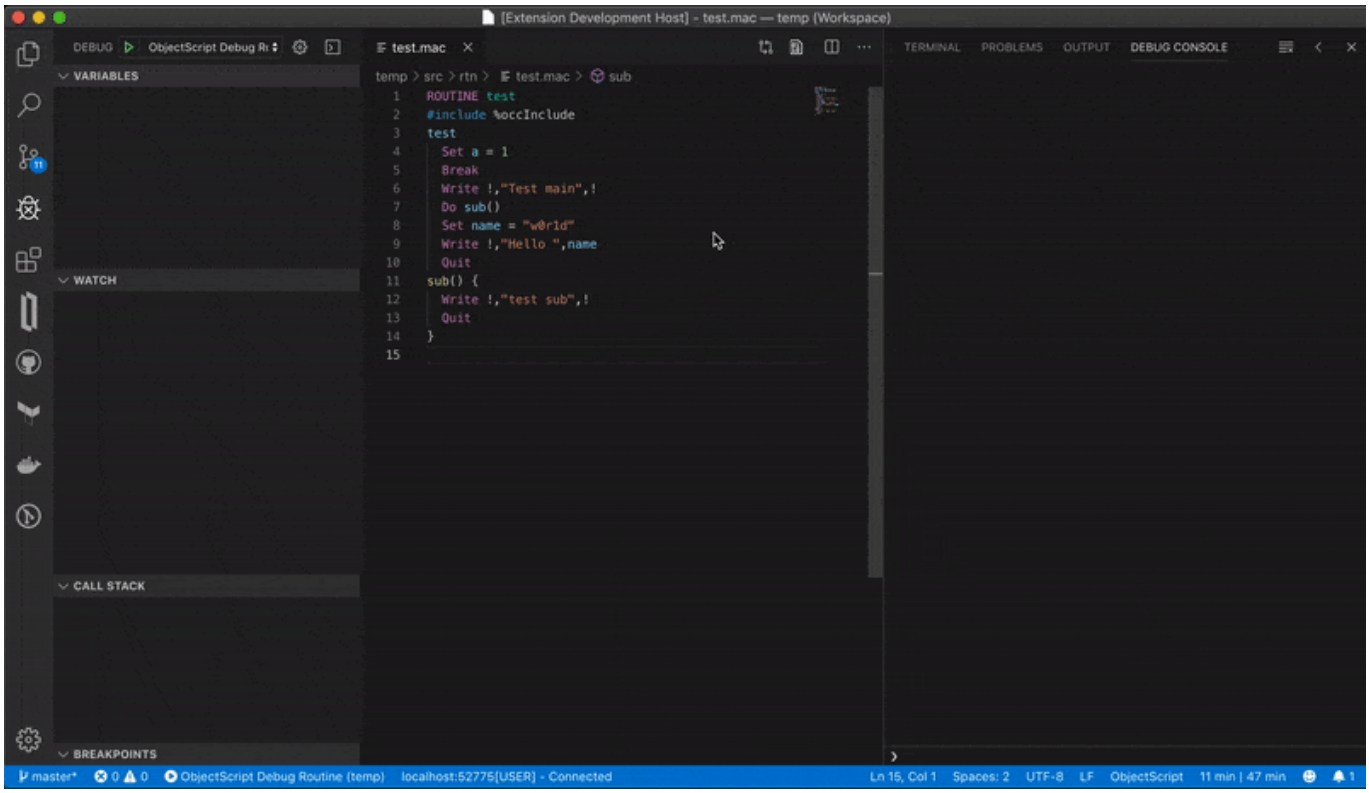
It is now possible to debug running ObjectScript code with VSCode-ObjectScript extension. You just have to fill .vscode/launch.json file with content similar to this.

```
{
  // Use IntelliSense to learn about possible attributes.
  // Hover to view descriptions of existing attributes.
  // For more information, visit: https://go.microsoft.com/fwlink/?linkid=830387
  "version": "0.2.0",
  "configurations": [
    {
      "type": "objectscript",
      "request": "launch",
      "name": "ObjectScript Debug Class",
      "program": "##class(User.Test).Test()",
    },
    {
      "type": "objectscript",
      "request": "launch",
      "name": "ObjectScript Debug Routine",
      "program": "^test",
    },
```

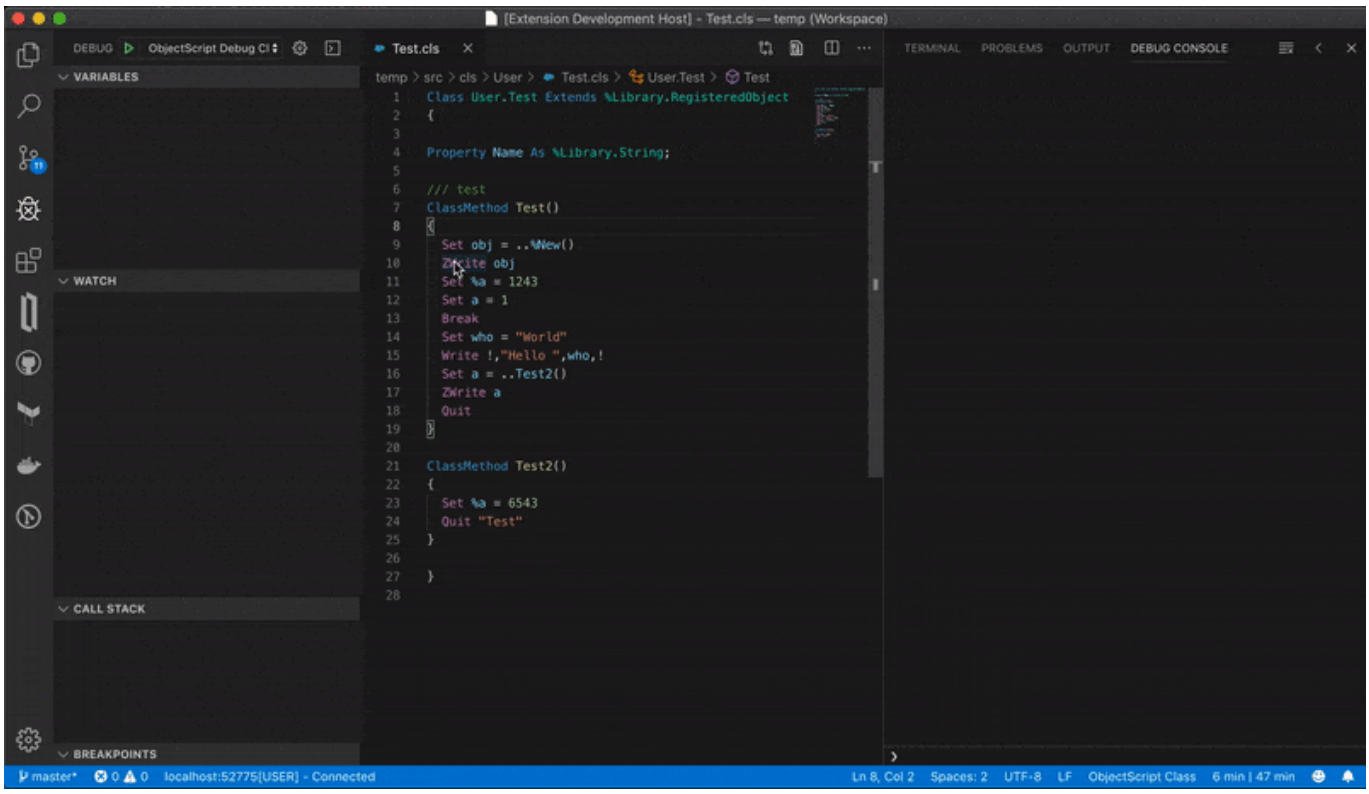
```
    {
      "type": "objectscript",
      "request": "attach",
      "name": "ObjectScript Attach",
      "processId": "${command:PickProcess}",
      "system": true
    }
  ]

}
```

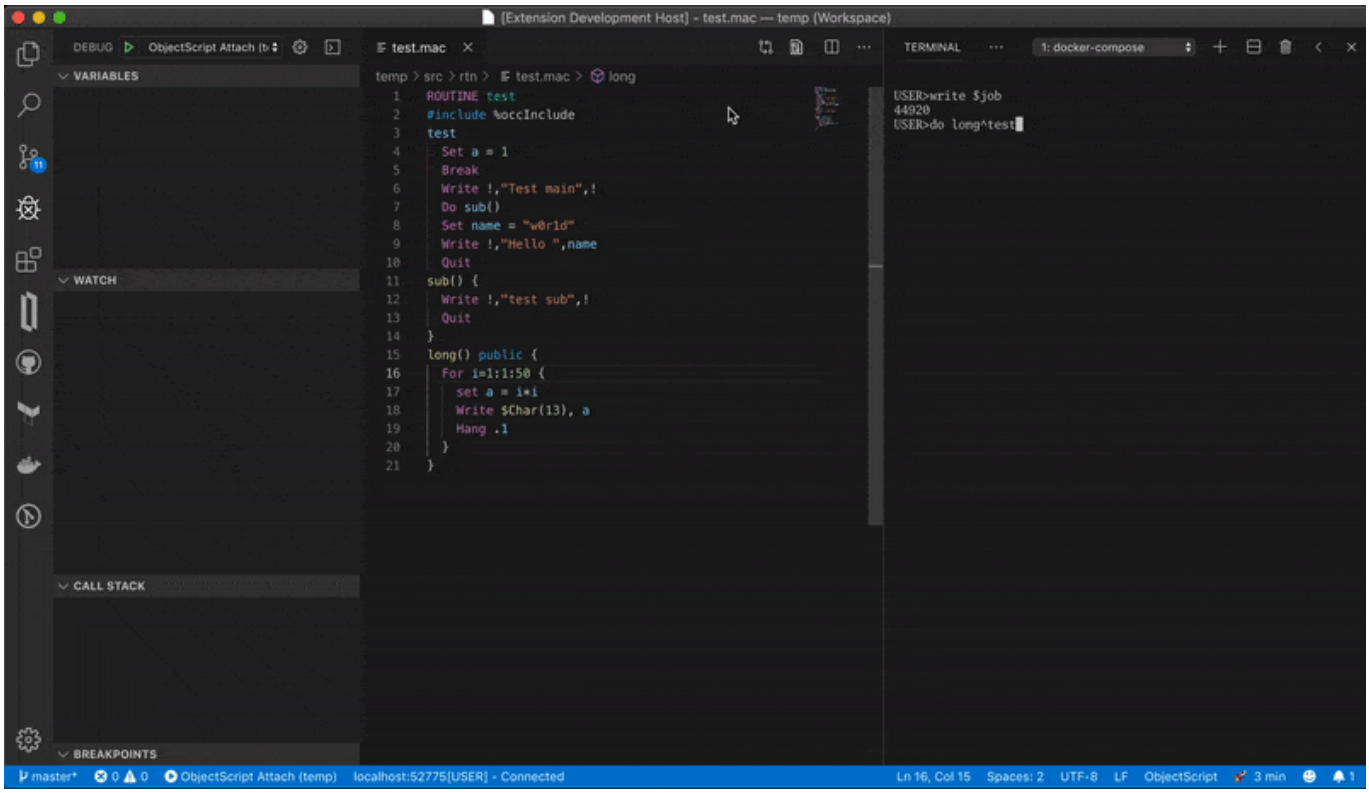
Let's debug some routine. You may notice here, that debug console shows all output happen during the run. Local and global variables visible there, as well as stack.



Next, go with debugging some class. As you may notice here, it is also possible to evaluate any ObjectScript expressions (variables, functions and so on) in the debug console, or in watch panel. Break command also stops running, of course.



You can also connect to any running process, just after connect it stops, and after continuing it returns control back to the process.



VSCode-ObjectScript debugger utilize Atelier debugger API, so, you don't need to install anything on your server, just connect to any Caché or IRIS with version (2016.1+) which supports Atelier.

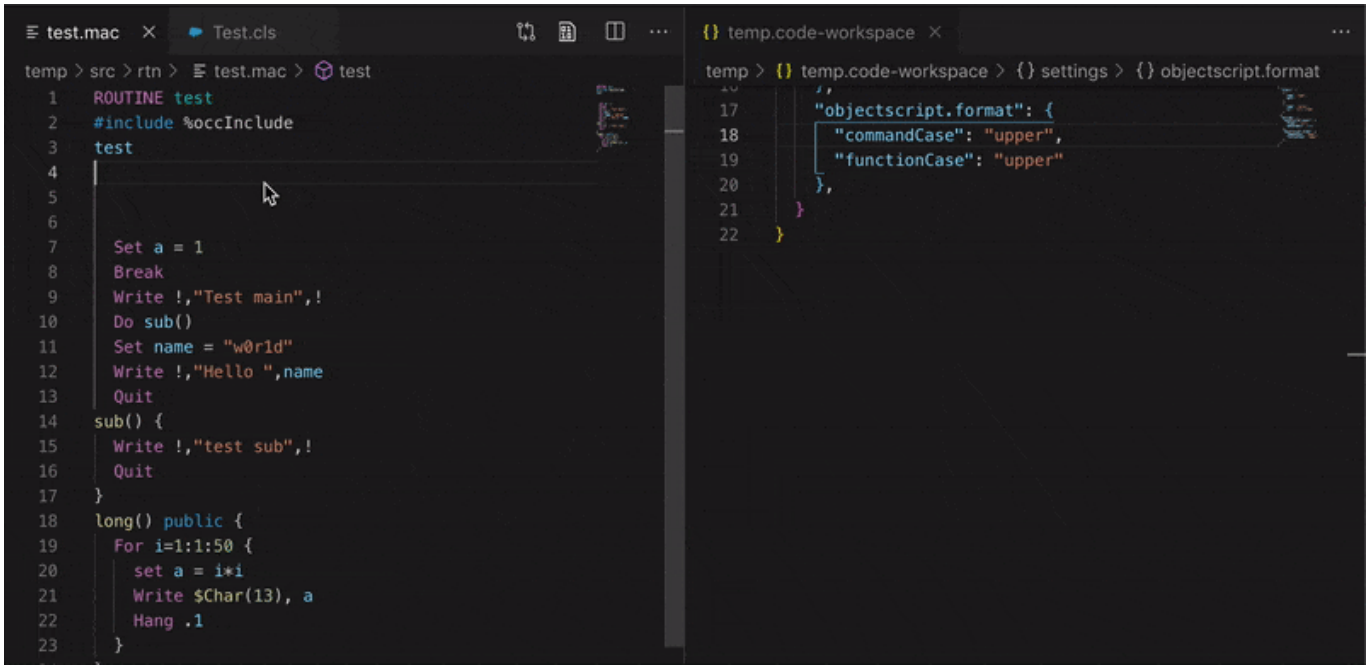
Formatting

Previously when you vscode suggested commands and functions it was instead than in UPPERCASE only. Now

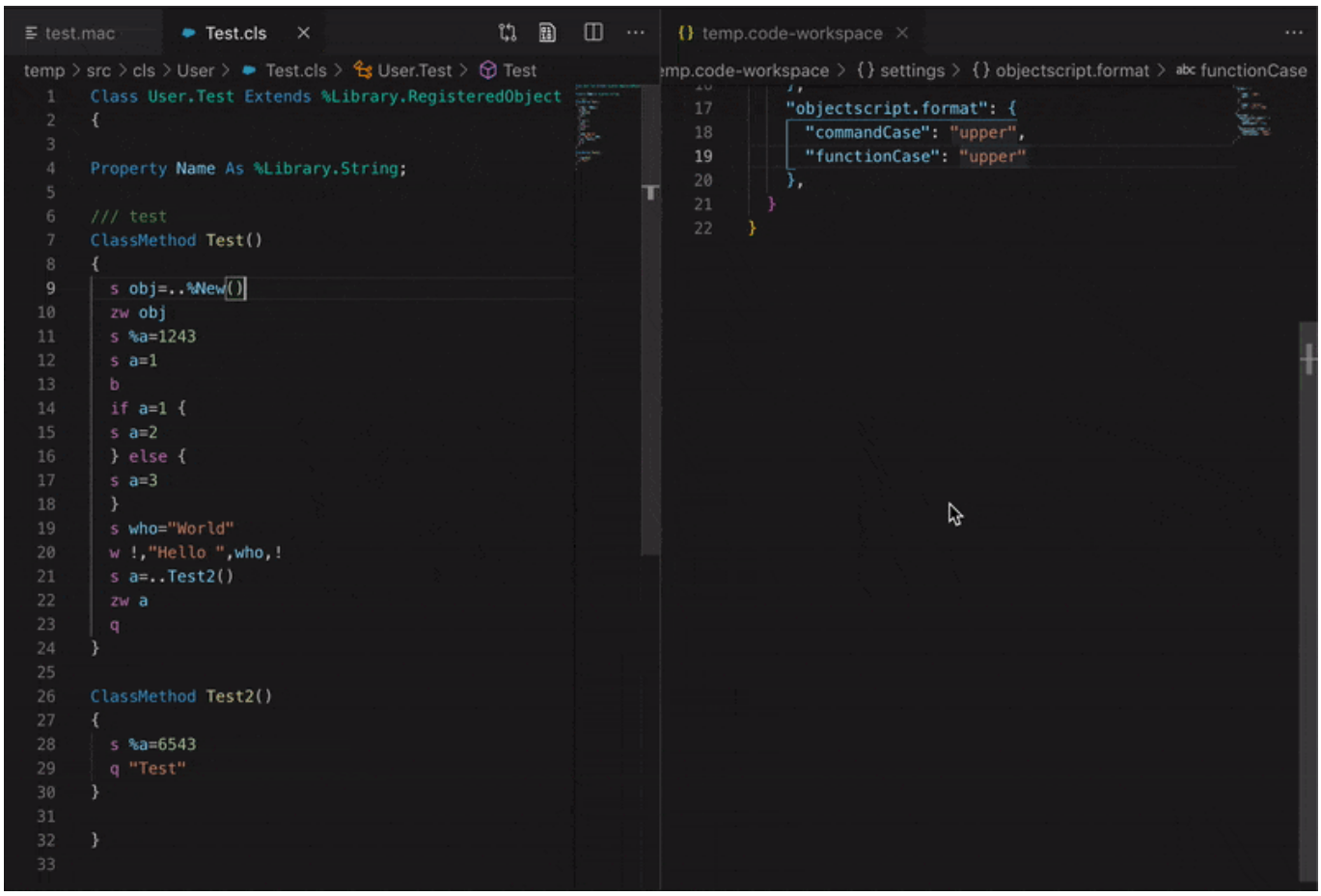
you can select, the way you prefer more. There are three options, upper, lower and word

```
"objectscript.format": {
  "commandCase": "word",
  "functionCase": "word"
},
```

You can see how it works below.



VSCode itself has formatting support by Meta/Alt+Shift+F. And it can format code by some configured rules. Currently, it can replace short commands/functions to complete name, and set case as above. And it can fix indention in case if was used spaces.



Edit on server

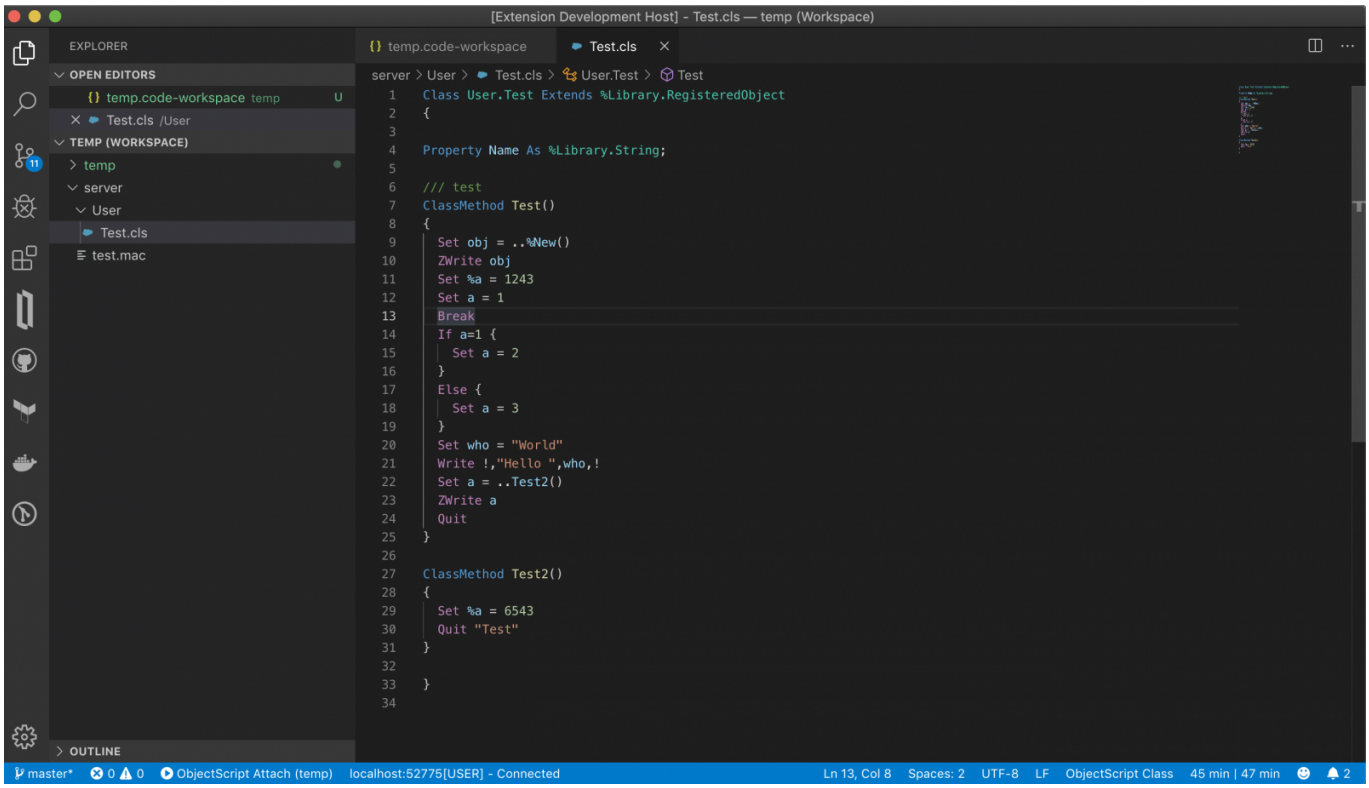
Some of the developers who just in the process of moving to the new way of storing sources and using VSCode, may find the possibility to edit files directly on the server very useful. And if you have Studio Control class, which used to export sources, it will work in VSCode-ObjectScript as well.

In VSCode you can configure projectname.code-workspace file, with content like this

```
{
    "folders": [
        {
            "name": "temp",
            "path": "."
        },
        {
            "name": "server",
            "uri": "isfs://temp"
        }
    ],
    "settings": {
    "objectscript.export": {
      "folder": "src",
      "addCategory": true,
    },
        "objectscript.format": {
            "commandCase": "word",
            "functionCase": "word"
        },
    }
```

```
}
```

When you open such workspace file in VSCode, you will get multi-root environment. With two root folders temp and server, where temp will follow to the root of the folder where you created code-workspace file, and server which will show sources directly from server configured in this project. You will be able to edit code directly on the server, but your changes will not be stored locally unless you have configured Studio Source Control class, exactly the same way as in Studio.



You can also set setting "objectscript.serverSideEditing": true to available to edit classes/rountines right from explorer view.

Export

If in your company you have some convention how to store sources in repository, you may find this feature useful. Setting "objectscript.export.addCategory" got some improvement with the latest version. Previously if you would set it to true, it was exported any classes to CLS folder, mac, inc and int to RTN folder. With this release you may setup a configuration like following below, if looking at the left side, for a regular expression, or just a file extension, and right side for a folder name

```
{
  "objectscript.export": {
    "addCategory": {
      "%.*\.cls": "_cls",
      "cls": "cls",
      "mac": "mac",
      "int": "rtn",
      "inc": null
    }
  }
}
```

Enterprise support

Recently we have started our own company CaretDev, which offers enterprise commercial support for VSCode-ObjectScript extension for companies which would like to introduce VSCode ObjectScript IDE in their software development process with InterSystems data platforms. We have also plans for individual developers, just contact us. But the extension itself is still free to use.

#CaretDev #Development Environment #ObjectScript #VSCode #Other
Check the related application on InterSystems Open Exchange