
Article

[Eduard Lebedyuk](#) · Oct 6, 2016 3m read

Tips & Tricks - Automatic removal of system methods from codebase

Recently I [wrote](#) about automatic removal of system methods from the codebase.

Today, I'm ready to present you a working [solution](#) (codename: SMR) for this task.

What it does?

For each class you wish to process SMR does the following

1. Replaces .\$ with .<something of your choice> (.% by default)
2. (Optionally) Capitalizes the letter after the \$
3. Replaces references from %Object to %DynamicObject and so on

User guide

[Import classes](#) and call one of the entry points, depending on what classes you want to process. There are entry points for:

- all user classes
- subclasses of some class
- classes with name matching LIKE SQL condition

Entry points are the following methods:

```
set st = ##class(SMR.Main).RemoveFromAllClasses(Replace, Capitalize)
set st = ##class(SMR.Main).RemoveFromSubclassesOf(Class, Replace, Capitalize)
set st = ##class(SMR.Main).RemoveFromMatchingClasses(Mask, Replace, Capitalize)
```

Arguments:

- Replace - what to replace \$ with (% by default but, for example, you can specify \$\$\$ if you have macros)
- Capitalize - capitalize the letter after \$ (boolean, yse by default)
- Class - class which subclasses the utility would try to convert (including the class)
- Mask - passed into the SQL query SELECT ID FROM %Dictionary.ClassDefinition Where ID LIKE ?

More throughout documentation is available in the code.

Example

Here's SMR.A class

```
Class SMR.A
{
ClassMethod A()
```

```
{
  set obj = ##class(%Library.Object).%New()
  set obj = {}. $fromJSON( '{"a":1}' )
  write obj.$toJSON()
}
}
```

If you don't care about 2016.1 compatibility, you can just call SMR with the defaults, like this:

```
write ##class(SMR.Main).RemoveFromMatchingClasses( "SMR.A" )
```

And SMR.A class would look like this:

```
Class SMR.A
{
ClassMethod A()
{
  set obj = ##class(%Library.DynamicObject).%New()
  set obj = {}. %FromJSON( '{"a":1}' )
  w obj.%ToJSON()
}
}
```

If, however you do care about 2016.1 compatibility, you can use [macro](#) approach as described [here](#). Lets say we want to replace \$<system method name> with \$\$\$json<System method name>. To do that, call SMR like this:

```
write ##class(SMR.Main).RemoveFromMatchingClasses( "SMR.A", " $$$json" )
```

It yields:

```
Class SMR.A
{
ClassMethod A()
{
  set obj = ##class(%Library.DynamicObject).%New()
  set obj = {}. $$$jsonFromJSON( '{"a":1}' )
  write obj.$$$jsonToJSON()
}
}
```

If you don't want to capitalize the first letter of system method name, you can provide Capitalize argument:

```
write ##class(SMR.Main).RemoveFromMatchingClasses( "SMR.A", " $$$json", 0 )
```

It yields:

```
Class SMR.A
{
ClassMethod A()
{
  set obj = ##class(%Library.DynamicObject).%New()
```

```
set obj = {}.$$$jsonfromJSON("{\"a\":1\"")
write obj.$$$jsontoJSON()
}
}
```

Notes

- Call TSTART before calling SMR, then view results in Studio and call TCOMMIT or TROLLBACK to finalize or remove the changes
- Carefully check resulting commit. This tool is NOT a syntax analyzer of any kind, it just uses \$find and regexp
- \$compose/%Compose method is unavailable in 2016.2. Don't forget to remove it
- SMR can be easily modified to do some general purpose find-replacing for arbitrary group of classes
- Works only in a current namespace
- Works only in 2016.2 Field Test or later
- SMR does not compile processed classes

Links

- [GitHub](#)

[#Caché](#) [#Change Management](#) [#Code Snippet](#) [#ObjectScript](#) [#Tips & Tricks](#)

Source URL: <https://community.intersystems.com/post/tips-tricks-automatic-removal-system-methods-codebase>