

---

Article

[Tani Frankel](#) · Dec 28, 2016 5m read

## Preventing Globals From Getting Journalled (Continued from How do I Minimize My Journals)

Coming back to the topic of how to minimize journals.

(It took me longer than I hoped since [my last post on this topic...](#))

As I concluded in my last post -

To avoid a global from being journalled there are several options:

- Turning off journaling system wide
- Mapping globals to CACHETEMP
- Mapping globals to non-journalled databases
- Turning off journaling for a process
- Turning off transactions for object filing

In this post I will cover these options:

a. [Stop journaling on a system-wide level](#)

For obvious reasons (detailed in my previous post), this is not recommended and is quite an "aggressive" move to solve a specific problem.

Programmatically you can do this via calling the [%SYS.Journal.System:Stop\(\)](#) method. (or you can use the [^JRNSTOP](#) utility).

For example:

```
%SYS>write ##class(%SYS.Journal.System).Stop()  
1
```

Then if I try the same example shown in the [previous post](#) of saving a Person object, in SAMPLES:

```
SAMPLES>set person = ##Class(Sample.Person).%New()  
SAMPLES>set person.Name="Doe, John"  
SAMPLES>set person.SSN="123-45-6789"
```

This does not get into the Journal file...

Note I am using the same SSN value that already exists in the database, though I know there is a Unique index defined on that field, so this Save should fail, and a rollback should be performed...

```
SAMPLES>set status = person.%Save()
```

But journaling is off so we get a [ROLLFAIL](#) error:

```
SAMPLES>write $system.Status.GetErrorText(status)
ERROR #5808: Key not unique: SSNKey
ERROR #5002: Cache error: <ROLLFAIL>%TrollBack+10^%occTransaction
```

(In theory if one would want to take this approach of totally disabling journaling, you'd also need to take care of system startup, since the system always comes up with journaling on, so you need to make sure journaling is to be stopped for every system startup (you'd put this into the SYSTEM label of the %ZSTART routine you'd define. See [here](#) for more details)

b. Map the relevant globals to CACHETEMP

CACHETEMP is a special database for temporary data, and [data in it is not journalled even in transactions](#).

Assuming you know what globals are the problematic ones (see a [previous post](#) re finding them) you can map the specific ones to CACHETEMP

A few considerations before you go ahead with this:

- You'd need to take special care if you'd consider mapping the same global from different namespaces to CACHETEMP, unintended "clashes" can occur and this should be prevented (for example by using a different identifier subscript).
- Another consideration is security. You might not want some Roles/Users having access to different data coming from various namespaces.
- The third one is quite obvious but still worth pointing out - CACHETEMP is for temporary data. As such it's contents gets deleted upon restart. In some cases you might have opted not to journal the data as in case of disaster you can recover the data by some other means (rebuilding from some external source, or by some logic) but you might not want to have to rebuild this data every system startup.

Here's an example how this plays out - say I [map the global](#) ^myMapped from the ENSEMBLE namespace to CACHETEMP database:

## Global Mappings

New Global MappingSave ChangesCancel Changes

The global mappings for namespace ENSEMBLE are c

Global Mapping
<h3>Global Mapping</h3> <p>Map a new global in namespace ENSEMBLE:</p> <div><div>Global database location: CACHETEMP</div><div>Global name: myMapped</div></div>

[By the way if you prefix your global name by CacheTemp (e.g. ^CacheTempMyGlobal) it will be automagically mapped to CACHETEMP, without explicitly defining the mapping]

I then set it (together with another global for comparison - ^notMapped):

```
ENSEMBLE>set ^notMapped=1
ENSEMBLE>set ^myMapped=1
ENSEMBLE>set ^notMapped=2
```

In the journal we just see the ^notMapped global:

132064	2013-06-16 15:09:11	8852	SET	No	^notMapped	c:\intersystems\ensemble201310\mgr\ensemble\
132112	2013-06-16 15:09:11	8852	SET	No	^notMapped	c:\intersystems\ensemble201310\mgr\ensemble\

And since this global is mapped to CACHETEMP this is true even in a transaction:

```
ENSEMBLE>tstart
TL1:ENSEMBLE>set ^notMapped=1
TL1:ENSEMBLE>set ^myMapped=1
TL1:ENSEMBLE>set ^notMapped=2
TL1:ENSEMBLE>tcommit
ENSEMBLE>
```

Again, the ^myMapped global is not journaled:

133012	2013-06-16 15:09:11	8852	BeginTrans	Yes		
133028	2013-06-16 15:09:11	8852	SET	Yes	^notMapped	c:\intersystems\ensemble201310\mgr\ensemble\
133080	2013-06-16 15:09:11	8852	SET	Yes	^notMapped	c:\intersystems\ensemble201310\mgr\ensemble\
133560	2013-06-16 15:09:11	8852	CommitTrans	Yes		

Note rolling back such a transaction will not generate an error, but the CACHTEMP-mapped global will not get rolled-back.

Say I start off with both (mapped and not mapped) globals like this:

```
ENSEMBLE>set ^myMapped=0
ENSEMBLE>set ^notMapped=0
```

Then in the transaction I change their values, but eventually roll it back:

```
ENSEMBLE>tstart
TL1:ENSEMBLE>set ^myMapped=1
TL1:ENSEMBLE>set ^notMapped=1
TL1:ENSEMBLE>trollback
```

So the not-mapped will roll back its value:

```
ENSEMBLE>write ^notMapped
0
```

But the mapped will not...

```
ENSEMBLE>write ^myMapped
1
```

You can see [here](#) more about mapping globals (as well as the link from the Tutorial provided above). Note you can also use wildcards to define a global prefix.

Regarding defining this programmatically you can use the [Config.MapGlobals](#) related class, or use the related %Installer manifest tag [<GlobalMapping>](#). For example from the Sample.Installer class in the SAMPLES namespace:

```
<!-- MyApp mappings back to the MYAPP DB.
-->
<GlobalMapping Global="MyAppData.*"           From="MYAPP"/>
<GlobalMapping Global="cspRule"               From="MYAPP"/>
```

c. Map the relevant globals to a non-journaled database

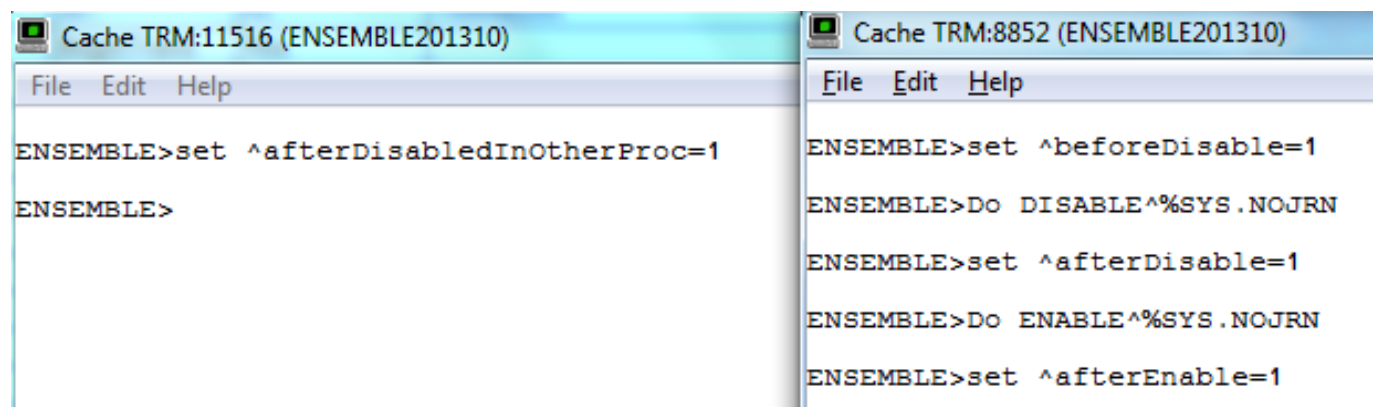
This is similar to the CACHETEMP mapping approach - only one needs to keep in mind, as previously mentioned, that if the global gets set within a transaction the global will still be journaled even if the database it's in is marked as not journaled. Therefore this approach is appropriate only if you are sure the data is being manipulated not within transactions. Note that by default, any object saving, is done within transactions (see below a note about that)

d. Turn off journaling for the specific process

Turning off journaling all-together for a specific process (and for a specific time-frame) would prevent journaling for any globals within the process (even in transactions).

You can do this by calling the [DISABLE^%NOJRN](#) routine (at some stage also referred to as %SYS.NOJRN).

For example I did the following (in two different Terminals to illustrate that while one process has journals disabled, the other still journals):



```
Cache TRM:11516 (ENSEMBLE201310)
File Edit Help
ENSEMBLE>set ^afterDisabledInOtherProc=1
ENSEMBLE>

Cache TRM:8852 (ENSEMBLE201310)
File Edit Help
ENSEMBLE>set ^beforeDisable=1
ENSEMBLE>Do DISABLE^%SYS.NOJRN
ENSEMBLE>set ^afterDisable=1
ENSEMBLE>Do ENABLE^%SYS.NOJRN
ENSEMBLE>set ^afterEnable=1
```

In the right-hand side (Process #8852) you can see I set a global before disabling, one during, and again one after, and on the left side (process #11516) I set one global while the journaling was disabled on the right side.

This is what we can see in the Journal:

145452	2013-06-16 15:09:11	8852	SET	No	^beforeDisable	c:\intersystems\ensemble201310\mgr\ensemble\
145748	2013-06-16 15:09:11	8852	SET	No	^CacheAuditID("2013-06-16 12:37:16.884","BookT:ENSEMBLE201310",4699)	c:\intersystems\ensemble201310\mgr\cacheaudit\
146280	2013-06-16 15:09:11	11516	SET	No	^afterDisabledInOtherProc	c:\intersystems\ensemble201310\mgr\ensemble\
146344	2013-06-16 15:09:11	8852	SET	No	^afterEnable	c:\intersystems\ensemble201310\mgr\ensemble\

You can see the ^beforeDisable from 8852 (you can ignore the Audit entry as it audits the journal state change), but not the SET while journal was disabled for process 8852, while you still see the SET that was performed in 11516 while it was disabled in 8852, and then finally once it was enabled again you can see the 8852 got the SET journaled again.

(The reverse by the way, as seen in the example above, is ENABLE^%NOJRN)

#### e. [Turning off transactions for object filing](#)

Per above even if a database is marked to be not journaled, within a transaction, and an object Save is by default in a transaction, globals are still journaled. So you can turn off transactions for object saving by calling the [%SYSTEM.OBJ.SetTransactionMode\(\)](#) method.

For example, the same from above, while saving a Person object in SAMPLES which is not journaled, though in %Save(), this will not journal due to turning off the transaction mode:

```
SAMPLES>write $system.OBJ.SetTransactionMode(0)
1
SAMPLES>set person=##class(Sample.Person).%New()

SAMPLES>set person.Name="Last,First"

SAMPLES>set person.SSN="123-45-6788"

SAMPLES>write person.%Save()
1
```

Again this would help only for a database that is anyway not journalled.

[#Journaling](#) [#Mapping](#) [#Object Data Model](#) [#System Administration](#) [#Caché](#)

---

Source

URL: <https://community.intersystems.com/post/preventing-globals-getting-journalled-continued-how-do-i-minimize-my-journals>