

# Protégez votre code DDL avec sp\_hidetext



par fadace ([Fabien Celaia](#))

Date de publication : 9.9.2007

Dernière mise à jour : 14.9.2007

Comment bien utiliser sp\_hidetext... et comment l'inactiver...

I - Introduction.....	3
II - Stockage des objets compilés.....	4
III - Usage de sp_hidetext.....	6
IV - Inactivation de sp_hidetext.....	7
V - Réactivation de sp_hidetext.....	8

## I - Introduction

Sybase ASE a toujours été considéré comme une base "ouverte", ce qui a fait son succès pendant de nombreuses années. Pour un DBA Sybase expérimenté, il est quasiment toujours possible de s'en sortir puisque l'accès aux couches "profondes" du SGBDR lui sont toujours ouvertes et relativement bien documentées : accès direct aux tables systèmes, nombreuses commandes dbcc, vues et procédures système éditables et modifiables ... à ne pas confondre cependant avec les bases OpenSource, le code C du moteur ASE étant scrupuleusement gardé.

Certains éditeurs de logiciels sur Sybase ASE ont émis le désir de pouvoir cacher le code de leurs objets compilés, ce qui n'était pas possible en versions antérieures à 11.5.

## II - Stockage des objets compilés

Il faut savoir que les objets compilés sous ASE (et plus particulièrement les procédures stockées et les déclencheurs) sont stockés dans deux tables distinctes et spécifiques dans leur base propre.

- sysprocedures : stocke le code compilé
- syscomments : stocke le code en clair

Voici donc un exemple basé sur la création d'une petite procédure stockée baptisée sp\_test

```
create procedure sp_test (@cle int)
as
begin
select @@version

select *
from test
where id=@cle
end
go
```

A noter que syscomments permet de prouver au passage que les \* sont remplacé par la liste des colonnes dans un objet compilé. L'exemple ci-après restitue assez correctement la syntaxe de la procédure stockée (dernier éditeur Sybase avec possibilité de revenir à la ligne à 255 caractères). Il n'en est cependant pas toujours de même, principalement avec de longues lignes, la colonne syscomments.text étant un varchar(255).

```
select text
from syscomments
where id = object_id('sp_test')
order by colid
GO

text
-----
create procedure dbo.sp_test (@cle int)
as
begin
select @@version

/* Adaptive Server has expanded all '*' elements in the following statement */ select test.id,
test.dates, test.jours,
test.mois, test.annees, test.heures, test.minutes, test.semestres, test.semaines, test.jourDansAnnee,
test.jourDansSemaine
from test
where id=@cle
end
```

Quant à sysprocedures, rien n'est visible par le quidam...

```
select *
from sysprocedures
where id = object_id('sp_test') and type&2=2
order by sequence
```

type	id	sequence	status	number	version
2	768002736	0	1152	1	15000
2	768002736	1	1152	1	15000
2	768002736	2	1152	1	15000
2	768002736	3	1152	1	15000
2	768002736	4	1152	1	15000
2	768002736	5	1152	1	15000
2	768002736	6	1152	1	15000
2	768002736	7	1152	1	15000


2	768002736	8	1152	1	15000
2	768002736	9	1152	1	15000
2	768002736	10	1152	1	15000

La première technique de caviardage de son code consistait à purger la table syscomments, mais cela posait des problèmes lors des migrations où la recompilation de l'objet nécessitait la relecture via la table.

```
sp_configure "allow updates",1
go
delete syscomments where id = object_id('sp_test')
go
(2 lignes affectées)

sp_configure "allow updates",0
go
```

Dès à présent, l'extraction du DDL posera problème : c'est l'effet escompté, mais assez peu élégant.

 **CO55: The specified Compiled Object Item not found in the Server: test.dbo.sp\_test at com.sybase.ddlgen.item.CompiledObjectItem.open(CompiledObjectItem.java:258) at com.sybase.ddlgen.container.CompiledObjectContainer.open(CompiledObjectContainer.java:131) at com.sybase.ddlgen.DDLThread.run(DDLThread.java:88)**

### III - Usage de sp\_hidetext

Après avoir recréé notre procédure stockée sp\_test, nous allons la caviarder à l'aide de sp\_hidetext

```
drop procedure sp_test
go

create procedure dbo.sp_test (@cle int)
as
begin
select @@version

select *
from test
where id=@cle
end
go

sp_hidetext sp_test
go
```

Voyons maintenant ce qu'il en advient dans syscomments

```
select text from syscomments where id=object_id('sp_test')
go

TEXT
----
NULL
NULL
```

La tentative d'extraire le DDL retourne maintenant un message explicite, sans plantage

```
--Source text for compiled object test.dbo.sp_test (id = 816002907) is hidden.
```


## IV - Inactivation de sp\_hidetext

Si un applicatif s'installe chez vous en utilisant cette fonction afin de vous cacher ses propres SP (c'est le cas par exemple de Cast Workbench), vous pouvez toujours, avant son installation, rendre inopérant le sp\_hidetext de la manière suivante

```
use sybssystemprocs
go
create procedure sp_hidetext
@objname varchar(255) = NULL,
@tablename varchar(255) = NULL,
@username varchar(255) = NULL
as
return 0
go
```

... et afin d'éviter que l'applicatif lui-même ne tente de la recréer (ceci étant sans effet si l'installation se fait à l'aide d'un compte ayant des droits system admin)...

```
grant execute on sp_hidetext to public
revoke delete on sp_hidetext from public
```

 **La fonction sp\_unhidetext n'existe pas : il est donc impossible de retrouver un texte caché par sp\_hidetext. La procédure qui suit permet d'inactiver sp\_hidetext pour les nouveaux objets, mais ne traite pas les objets déjà impactés.**

## V - Réactivation de sp\_hidetext

Le script d'origine est aisément extractable de instmstr, fichier situé sous \$SYBASE/\$SYBASE\_ASE/scripts (ou %SYBASE%\%SYBASE\_ASE%\scripts sous Windows)

```
if exists (select *
  from sysobjects
  where sysstat & 7 = 4
  and name = 'sp_hidetext')
begin
  drop procedure sp_hidetext
end
go
print "Installing sp_hidetext"
go
create procedure sp_hidetext
@objname varchar(255) = NULL,
@tablename varchar(255) = NULL,
@username varchar(255) = NULL
as
declare @ret int
execute @ret = sp_aux_text @objname, @tablename, @username, 2
return @ret
go
exec sp_procxmode 'sp_hidetext', 'anymode'
go
grant execute on sp_hidetext to public
go
```