

# Cross-AppDomain calls and different assembly versions

---

*Source:*

<http://www.tech-archive.net/Archive/DotNet/microsoft.public.dotnet.framework.clr/2006-07/msg00026.html>

---

- *From:* [kai.iske@xxxxxx](mailto:kai.iske@xxxxxx)
  - *Date:* 10 Jul 2006 23:53:46 -0700
- 

Hi,

I have seen a couple of threads dealing with similar questions, but I haven't found one to answer my question, so here I go:

I have a service application that loads a couple of tasks (Plug-Ins you might want to call them). When a task gets executed it will be executed in its own AppDomain to allow

- a) Easy unloading of DLLs
- b) Version / Assembly independence to the controlling service

The service exposes a support object that implements a specific interface. This object supports (for example) sending emails and relies on a third party component for performing the SMTP tasks.

The interface reference to this object is passed to the tasks when being executed (so we have one cross-appdomain call here). The tasks perform their actions and might call back on the supplied interface (cross-appdomain again).

However some tasks might require more improved email functionality. So they link against that SMTP component themselves which might be different to the version used by the service.

Problem is when one of these tasks actually calls back into the supplied support object, the CLR comes back to me with a version / assembly manifest mismatch exception. I would have thought that when I return to the service AppDomain from my Task AppDomain I would be in "safe land" as the object would be executed with any of its local assemblies (no assembly is installed in the GAC), However to me it seems that when the task calls back to the service, CLR attempts to bind to the task's version of the SMTP component rather than the service's component.

Is there any easy way I could get around this?

## Cross-AppDomain calls and different assembly versions

Thanks

Kai

.