**Auto Application Inialization (Preload) on IIS**

3 minute read

**Problem:**

Sometimes we need to perform application initialization to avoid “warm up” time on first request or “cold start” issue on huge and complex web application which may need to perform lengthy startup processing, load quite a few libraries, prime in-memory caches, generate or pre-process some content etc. prior to serving the first HTTP request.

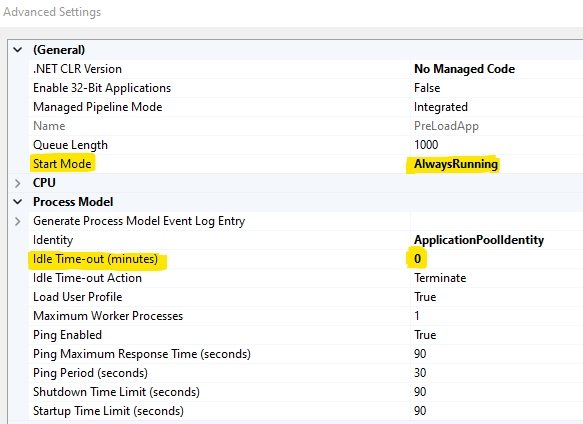
Sometimes we also face time-out issues due to longer “warm-up” time. Also in some weird case, when first request to Website fails and you want to hide that from end users.

**Solution:**

The Application Initialization feature is configured through a combination of global and application-specific rules that tell IIS how and when to initialize web applications. The Application Initialization feature also supports integration with the IIS Url Rewrite Module to support more complex handling of placeholder content while an application is still initializing.

While an application is being initialized, IIS can also be configured to return static content as a placeholder or “splash page” until an application has completed its initialization tasks.

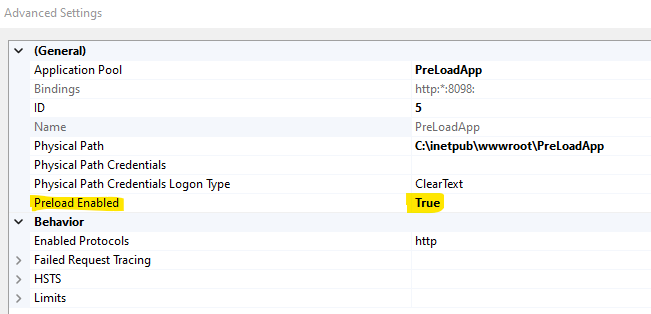
Following steps need to perform on IIS server to proactively perform Application Initialization task.

* Install Application Initialization feature - [*IIS 8.0 Application Initialization*](https://docs.microsoft.com/en-us/iis/get-started/whats-new-in-iis-8/iis-80-application-initialization)
* Make sure the warmup.dll (which should load from C:\Windows\SysWOW64\inetsrv\warmup.dll or from C:\Windows\system32\inetsrv\warmup.dll depending on the bitness of your process) present at the path.
* Configure the app pool to be always running (from the application pool advanced properties).  
    
  In the **applicationHost.config** (%WINDIR%\system32\inetsrv\config\applicationHost.config) file the application pool setting looks like this…

<add name="PreLoadApp" autoStart="true" managedRuntimeVersion="" startMode="AlwaysRunning">

<processModel idleTimeout="00:00:00" />

</add>

* Enable Preload by navigating to the website/application and then right click and go to ‘Manage Website’ and set ‘Preload Enabled’ to true.  
    
  You may also directly open the **applicationHost.config** (%WINDIR%\system32\inetsrv\config\applicationHost.config) file and do the changes on site as below…

<site name="PreLoadApp" id="5">

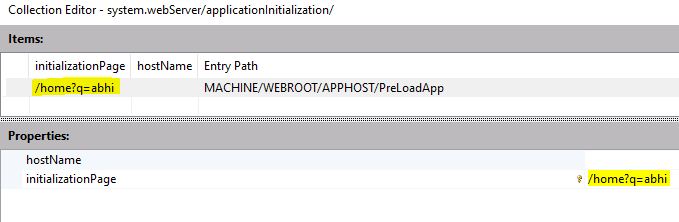
<application path="/" applicationPool="PreLoadApp" preloadEnabled="true">

<virtualDirectory path="/" physicalPath="C:\inetpub\wwwroot\PreLoadApp" />

</application>

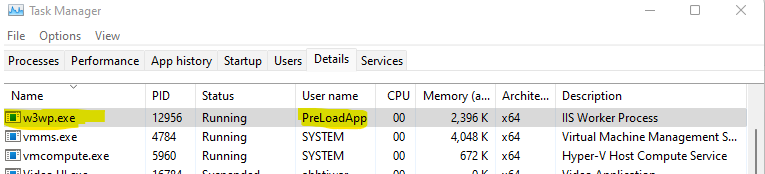
...

Setting preloadEnabled to “true” tells IIS that it sends a “fake” request to the application when the associated application pool starts up. That is why in the previous step we set the application pool’s startMode to “AlwaysRunning”.

* Then select the site from the IIS Manager tree view on the left-hand side and go to the configuration editor.
* this time underneath the <system.WebServer/applicationInitialization> tag, and look at the list of requests
* set the request to only target a page called (you may or may not set any host header). I provided a query string `q=abhi` //to verify in IIS log to ensure that request is coming from preload only.
* set the doAppInitAfterRestart parameter to true and apply the settings 

And you are done. Now after IIS restart/recycle, application should start automatically.   
To verify the configuration, restart the IIS by running below command from elevated command prompt.

net stop w3svc & net start w3svc

Open the task manager and you will see that application would be initialized automatically. Also, you can verify the IIS log to see if there is any request with the query-string q=abhi. 

**Additional:**

If your application is taking longer time to warm-up and in the meantime if you want to show some loading page or some informative page for your users, you can configure splash page by adding configurations as below in your application web.config inside the <system.webServer> configuration section.

<applicationInitialization

remapManagedRequestsTo="Startup.htm"

skipManagedModules="true" >

<add initializationPage="/default.aspx" />

</applicationInitialization>

Note: The applicationInitialization element tells IIS that it should issue a request to the application’s root Url (“/default.aspx” in this example) in order to initialize the application. While IIS waits for the request to “/default.aspx to complete, it will serve “Startup.htm” to any active browser clients. “Startup.htm” is the “splash page” for the application.

**Authentication:**

* Ideally the Preload endpoint/ Application Initialization should be configured with anonymous authentication. The InitializationPage gets hit by the IIS Account NT AUTHORITY\IUSR and not App Pool Account, but it fails when you disable anonymous auth and just enable windows.
* It is better to have one Preload page anonymously accessible in your application and put your desired auth on rest of the application pages. You can also limit the caller to that preload page to localhost only.
* If you want to use Windows auth and can’t have separate page, maybe you can send a request to the server, receive a response, in this case it should be 401 and that should serve the purpose of preloading the app. But make sure that request reaches application layer, and not only the Windows Authentication module (then the application’s code will never be called, and you will kind of miss the whole point).

Reference links -

 [Application Initialization | Microsoft Docs](https://docs.microsoft.com/en-us/iis/configuration/system.webServer/applicationInitialization/)

 [IIS 8.0 Application Initialization | Microsoft Docs](https://docs.microsoft.com/en-us/iis/get-started/whats-new-in-iis-8/iis-80-application-initialization)