.NET Framework Developer's Guide

Diagnosing Errors with Managed Debugging Assistants

Managed debugging assistants (MDAs) are debugging aids that work in conjunction with the common language runtime (CLR) to provide information on runtime state. The assistants generate informational messages about runtime events that you cannot otherwise trap. You can use MDAs to isolate hard-to-find application bugs that occur when transitioning between managed and unmanaged code. You can enable or disable all MDAs by adding a key to the Windows registry or by setting an environment variable. You can enable specific MDAs by using application configuration settings. You can set additional configuration settings for some individual MDAs in the application's configuration file. Because these configuration files are parsed when the runtime is loaded, you must enable the MDA before the managed application starts. You cannot enable it for applications that have already started.

The following table lists the MDAs that ship with the .NET Framework.

AsynchronousThreadAbort [http://msdn2.microsoft.com/en-us/library/ms172218.aspx]	BindingFailure [http://msdn2.microsoft.com/en-us/library/ms172215.aspx]
CallbackOnCollectedDelegate [http://msdn2.microsoft.com/en-us/library/43yky316.aspx]	ContextSwitchDeadlock [http://msdn2.microsoft.com/en-us/library/ms172233.aspx]
DangerousThreadingAPI [http://msdn2.microsoft.com/en-us/library/ms228985.aspx]	DateTimeInvalidLocalFormat [http://msdn2.microsoft.com/en-us/library/ms172213.aspx]
DirtyCastAndCallOnInterface [http://msdn2.microsoft.com/en-us/library/4k0c1c02.aspx]	DisconnectedContext [http://msdn2.microsoft.com/en-us/library/2c1czate.aspx]
DllMainReturnsFalse [http://msdn2.microsoft.com/en-us/library/e23362sb.aspx]	ExceptionSwallowedOnCallFromCom [http://msdn2.microsoft.com/en-us/library/ms172231.aspx]
FailedQI [http://msdn2.microsoft.com/en-us/library/k63k8b85.aspx]	FatalExecutionError [http://msdn2.microsoft.com/en-us/library/ms228990.aspx]
GcManagedToUnmanaged [http://msdn2.microsoft.com/en-us/library/ya4zeek2.aspx]	GcUnmanagedToManaged [http://msdn2.microsoft.com/en-us/library/w5b3680b.aspx]
IllegalPrepareConstrainedRegion [http://msdn2.microsoft.com/en-us/library/13x9c35f.aspx]	InvalidApartmentStateChange [http://msdn2.microsoft.com/en-us/library/tx1w5eks.aspx]
InvalidCERCall [http://msdn2.microsoft.com/en-us/library/bax7szd1.aspx]	InvalidFunctionPointerInDelegate [http://msdn2.microsoft.com/en-us/library/3bc7y8w2.aspx]
InvalidGCHandleCookie [http://msdn2.microsoft.com/en-us/library/ms228987.aspx]	InvalidIUnknownPointer [http://msdn2.microsoft.com/en-us/library/5z2ek4za.aspx]
InvalidMemberDeclaration [http://msdn2.microsoft.com/en-us/library/ms172230.aspx]	InvalidOverlappedToPInvoke [http://msdn2.microsoft.com/en-us/library/ms228991.aspx]
InvalidVariant [http://msdn2.microsoft.com/en-us/library/7zw2e2w7.aspx]	JitCompilationStart [http://msdn2.microsoft.com/en-us/library/fw872k46.aspx]

LoaderLock [http://msdn2.microsoft.com/en-us/library/ms172219.aspx]	LoadFromContext [http://msdn2.microsoft.com/en-us/library/ms172214.aspx]
MarshalCleanupError [http://msdn2.microsoft.com/en-us/library/cck0763d.aspx]	Marshaling MDA [http://msdn2.microsoft.com/en-us/library/a8fdekae.aspx]
MemberInfoCacheCreation [http://msdn2.microsoft.com/en-us/library/ms172216.aspx]	ModuloObjectHashcode [http://msdn2.microsoft.com/en-us/library/d2b773z9.aspx]
NonComVisibleBaseClass [http://msdn2.microsoft.com/en-us/library/ms172235.aspx]	NotMarshalable [http://msdn2.microsoft.com/en-us/library/y13tfsx2.aspx]
OpenGenericCERCall [http://msdn2.microsoft.com/en-us/library/d37kczxb.aspx]	OverlappedFreeError [http://msdn2.microsoft.com/en-us/library/ms228988.aspx]
PInvokeLog [http://msdn2.microsoft.com/en-us/library/ms172217.aspx]	PInvokeStackImbalance [http://msdn2.microsoft.com/en-us/library/0htdy0k3.aspx]
RaceOnRCWCleanup [http://msdn2.microsoft.com/en-us/library/ms172234.aspx]	Reentrancy [http://msdn2.microsoft.com/en-us/library/ms172237.aspx]
ReleaseHandleFailed [http://msdn2.microsoft.com/en-us/library/85eak4a0.aspx]	ReportAVOnComRelease [http://msdn2.microsoft.com/en-us/library/c9180yaf.aspx]
StreamWriterBufferedDataLost [http://msdn2.microsoft.com/en-us/library/ms172236.aspx]	VirtualCERCall [http://msdn2.microsoft.com/en-us/library/w5kxfdaf.aspx]

Enabling and Disabling MDAs

You can enable and disable MDAs by using a registry key, an environment variable, and application configuration settings. Either the registry key or the environment variable must be enabled to use the application configuration settings.

Enabling and Disabling by Using a Registry Key
You can enable MDAs by adding the HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\.NETFramework\MDA subkey in the Windows registry. Copy the following example into a text file named "MDAEnable.reg":

Copy Code

```
Windows Registry Editor Version 5.00
[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\.NETFramework]
"MDA"="1"
```

From a command prompt, execute the MDAEnable.reg file to enable MDAs on that computer. To disable MDAs, copy the following example into a text file named "MDADisable.reg":

Copy Code

```
Windows Registry Editor Version 5.00
[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\.NETFramework]
"MDA"="0"
```

Then, run the MDADisable.reg file from a command prompt.

By default, some MDAs are enabled when running the application attached to a debugger, even without adding the registry key. Examples of such assistants are PInvokeStackImbalance [http://msdn2.microsoft.com/en-us/library/0htdy0k3.aspx] and InvalidApartmentStateChange [http://msdn2.microsoft.com/en-us/library/tx1w5eks.aspx] . You can disable these assistants by running the MDADisable.reg file as described above.

Enabling and Disabling by Using an Environment Variable
MDA activation can also controlled by the environment variable COMPLUS_MDA. The environment variable overrides the

registry key. The string is a case-insensitive, semicolon delimited list of MDA names or other special control strings. Starting under a managed or unmanaged debugger enables a set of MDAs by default. This is done by implicitly prepending the semicolon delimited list of MDAs enabled by default under debuggers to the value of the environment variable or registry key. The special control strings are the following:

- 0 - Deactivates all MDAs.
- 1 - Reads MDA settings from *ApplicationName*.mda.config.
- `managedDebugger` - Explicitly activates all MDAs that are implicitly activated when a managed executable is started under a debugger.
- `unmanagedDebugger` - Explicitly activates all MDAs that are implicitly activated when an unmanaged executable is started under a debugger.

If there are conflicting settings, the most recent settings override previous settings:

- `COMPLUS_MDA=0` disables all MDAs including those implicitly enabled under a debugger.
- `COMPLUS_MDA=gcUnmanagedToManaged` enables **gcUnmanagedToManaged** in addition to any implicitly enabled under a debugger.
- `COMPLUS_MDA =0;gcUnmanagedToManaged` enables **gcUnmanagedToManaged** but disables those MDAs that would otherwise be implicitly enabled under a debugger.

Enabling and Disabling by Using Application-Specific Configuration Settings
You can enable, disable, and configure some assistants individually in the MDA configuration file for the application. To enable the use of an application configuration file for configuring MDAs, either the MDA registry key or the COMPLUS_MDA environment variable must be set. The application configuration file is typically located in the same directory as the application's executable file (.exe). The file name takes the form *ApplicationName*.mda.config; for example, notepad.exe.mda.config. Assistants enabled in the application configuration file may have attributes or elements specifically designed to control that assistant's behavior. The following example shows how to enable and configure the Marshaling MDA [http://msdn2.microsoft.com/en-us/library/a8fdekae.aspx] .

Copy Code

```
<mdaConfig>
  <assistants>
    <marshaling>
      <methodFilter>
        <match name="*"/>
      </methodFilter>
      <fieldFilter>
        <match name="*"/>
      </fieldFilter>
    </marshaling>
  </assistants>
</mdaConfig>
```

These settings enable and configure the **Marshaling** MDA, which emits information describing the managed type that is being marshaled to an unmanaged type for each managed-to-unmanaged transition in the application. The **Marshaling** MDA has further flexibility to filter the name of the method and structure fields supplied in the **<methodFilter>** and **<fieldFilter>** child elements, respectively.

For more information on the settings specific to each individual MDA, see the documentation for that MDA.

MDA Output
MDA output is similar to the following example, which shows the output from the **pInvokeStackImbalance** MDA.

```
A call to PInvoke function 'MDATest!MDATest.Program::StdCall' has unbalanced the stack. This is likely
because the managed PInvoke signature does not match the unmanaged target signature. Check that the
calling convention and parameters of the PInvoke signature match the target unmanaged signature.
```

See Also

Other Resources
Debugging and Profiling Applications [http://msdn2.microsoft.com/en-us/library/7fe0dd2y.aspx]

Community Content