**[Active Context及XP主题支持](http://www.baidu.com/s?tn=xue5&wd=.Net%CA%B5%C0%FD" \o "点击查看更多.Net实例" \t "_blank)**

Windows下的动态链接库可能存在多个版本，比如系统提供的基本控件所在的Comctl32.dll 就包括版本6.0和之前的版本。在新版本之前编写的程序能否保证绝对和新版本的动态链接库兼容呢？只能说，一般情况下可以，但是不能绝对保证。 Windows提供了解决办法，那就是设定程序执行的Active Context。对于客户自己编写的动态链接库，同样也存在这样的问题。这里说的是动态链接库，如果考虑动态链接库的其他应用，就涉及到更广阔的应用范围 了，可能是普通的动态链接库，可能是COM服务器等，这些都是可以通过配置程序执行环境来改变具体使用哪个动态链接库。  
        配置程序执行上下文，实际就是配置程序执行的组件配置，这里的组件只要是指动态链接库。对组件进行配置，简单来说，就是配置程序使用什么动态链接库。这种 配置具体是怎么实现的呢？其实就是在Windows程序真正启动之前增加了一层处理。这层处理就是把程序执行所需要的环境先建立好，直接点说就是把需要的 动态链接库准备好。  
  
        上面说到，配置组件就是在程序真正执行之前准备好需要的动态链接库，这项工作可以由Windows系统的Module Loader或者应用程序自己完成。具体的实现方式有三种：  
        1.文件级的配置：在应用程序所在的目录下创建一个.exe.manifest文件，这个文件中说明组件配置的具体信息，具体的信息格式可以查询MSDN。在VS2003及以上的版本，新建工程时，会自动生成一个manifest文件在res目录下。  
        2.资源级的配置：创建资源，类型：RT\_MANIFEST（值为24），ID：CREATEPROCESS\_  
MANIFEST\_RESOURCE\_ID（值 为1），内容为上面的manifest文件的内容。Module Loader在加载EXE之后，执行程序之前，会检测到程序的资源表中是否有类型为24，ID为1的资源，如果有这项配置信息，就会按照配置信息中的配 置，加载对应的动态链接库了。  
        3.资源和代码混合的配置：首先也需要添加资源，资源类型RT\_MANIFEST（值为24），ID为 ISOLATIONAWARE\_MANIFEST\_RESOURCE\_ID（值为2），内容也为上述符合格式的manifest文件；然后再 在#include之前增加一句  
#define ISOLATION\_AWARE\_ENABLED 1。这样编译器在编译时，会把一些API重定向到另外一个API，这个API先进行当成程序Active Context的检测，如果程序执行Context没有配置成资源文件所描述，会调用相应的API配置好Context，然后再从目的API所在的动态链 接库中获得函数地址（GetProcAddress）后，通过函数指针调用目标API，最后执行完毕后恢复执行环境。这可以保证在系统Module Loader没有程序Active Context支持时，配置好程序执行环境。但是采用这种方法时需要注意，这是一种代码级的支持，所以对MFC/ATL并不支持，因为MFC/ATL库编 译时，并不是在ISOLATION\_AWARE\_ENABLED已经定义的情况下编译的。  
        4.代码级的配置：除了以上几种方法之外，你还可以通过调用相应的API，自己通过代码来控制当前的执行环境，还可以做到一段代码调用这个版本动态链接库 中的代码，另外一段代码调用另外一个版本的动态链接库代码。程序执行Context是线程有效的。这可能在什么情况下有应用呢？比如说，你要开发某个应用 程序的插件，你的插件依赖与系统提供的某个版本的动态链接库，但是应用程序在不断的升级，不能保证应用程序的执行环境始终与最初发布版本一致。这时你就可 以自己通过代码来限定执行环境了。需要注意的是，如果程序已经采用了第三种方式进行执行环境的管理了，你就不能使用代码级的环境管理了。  
前面说的都是程序执行Context，实际准确的说应该是模块执行Context。对于动态链接库，同样可以进行类似的配置。这些配置都有相应的API及manifest文件格式，详见MSDN。