类CConnectionPoint定义了与OLE对象通信的一类特殊接口，它称为“连接点”（Connection Point）。一般的OLE接口实现并向外提供OLE控件的功能，而连接点则实现了外向的接口，从而能初始化其它对象的动作，如激活事件并改变通知。  
一个连接由两部分组成：调用接口的对象，称为源端；实现接口的对象，称为接收端（槽）。源端提供连接点使接收端和它建立连接。通过连接点机制，源对象可以取得接收端对一些成员函数的实现的指针。例如，要激活一个由接收端实现的事件，源端可以调用接收端的相应实现。  
缺省时，从类COleControl派生出的类实现了两个连接点：一个用于事件，一个用于属性变化通知。它们分别用于激活事件和通知接收端（如控件的包容器）属性发生了变化。MFC也为实现更多的连接点提供了支持。在用户派生的控制类中的每个附加的连接点，都必须声明一个“连接部分”（connection part）来实现连接点。实现附加连接点，也需要在控制类中声明一个“连接映射”（connection map）。  
下面的例子实现了Sample OLE控件上一个简单的连接映射和连接点，它包括两部分代码：第一部分声明了连接映射和连接点；第二部分实现了该映射和连接点。第一部分要插入到控件类protected部分下的声明中：  
// Connection point for ISample interface  
BEGIN\_CONNECTION\_PART( CSampleCtrl, SampleConnPt )  
    CONNECTION\_IID( IID\_ISampleSink )  
END\_CONNECTION\_PART( SampleConnPt )  
  
DECLARE\_CONNECTION\_MAP( )  
宏BEGIN\_CONNECTION\_PART和END\_CONNECTION\_PART声明了一个嵌入类XSampleConnPt（从CConnectionPoint继承来），该类实现了这个连接点。如果想覆盖CConnectionPoint的成员函数或者增加新的成员函数，要在这两个宏之间声明。例如，上例中的宏CONNECTION\_IID覆盖了CConnectionPoint::GetIID。  
代码的第二部分要插入到控制类的实现文件（.CPP）中。它实现了连接映射，包括附加的连接点SampleConnPt：  
BEGIN\_CONNECTION\_MAP( CSampleCtrl, COleControl )  
    CONNECTION\_PART( CSampleCtrl, IID\_SampleSink, SampleConnPt )  
END\_CONNECTION\_MAP( )  
插入这些代码之后，Sample OLE控件就为接口ISampleSink提供了一个连接点。  
典型地，连接点支持多播，即向连接到同一个接口的多个接收端广播。下面的例子通过向每个接收端发送实现多播。  
void CSampleCrtl::CallSinkFunc( )  
{  
  const CPtrArray\* pConnections = m\_xSampleConnPt.GetConnections( );  
  ASSERT( pConnections !=NULL );  
  int cConnections = pConnections->GetSize( );  
  ISampleSink\* pSampleSink;  
  for( int i=0 ; i<cConnections; i++)  
  {  
    pSampleSink = (ISampleSink\*)(pConnections->GetAt(i));  
    ASSERT(pSamplesink != NULL);  
    pSampleSink->SinkFunc( );  
  }  
}