Any given property may have more than one interceptor action applied to it. For example:

```
#region Chained Interceptors
 [AfterGet(EntityPropertyNames.LastName)]
 public void UppercaseLastName(
   PropertyInterceptorArgs<Employee, String> args) {
  /// … do something interesting
 }
  [AfterGet(EntityPropertyNames.LastName)]
  // same mode (afterGet) and property name as above
 public void InsureNonEmptyLastName(
   PropertyInterceptorArgs<Employee, String> args) {
  // … do something else interesting
 }
  [AfterGet] // same mode as above and
         //applying to all properties on employee.
 public void AfterAnyEmployeeGet(
   PropertyInterceptorArgs<Employee, Object> args) {
  // … global employee action here
 }
#endregion Chained Interceptors
```

```
<AfterGet(EntityPropertyNames.LastName)> _
Public Sub UppercaseLastName(ByVal args _
 As PropertyInterceptorArgs(Of Employee, String))
 ''' … do something interesting
End Sub
<AfterGet(EntityPropertyNames.LastName)> _
Public Sub InsureNonEmptyLastName(ByVal args _
 As PropertyInterceptorArgs(Of Employee, String))
 ' … do something else interesting
End Sub
<AfterGet()> _
Public Sub AfterAnyEmployeeGet(ByVal args _
 As PropertyInterceptorArgs(Of Employee, Object))
 ' … global employee action here
End Sub
```

In this case, three different interceptor actions are all ‘registered’ to occur whenever the Employee.LastName property is called.

To execute these actions, the DevForce engine forms a chain where each of the ‘registered’ interceptor actions is called with the same arguments that were passed to the previous action. Any interceptor can thus change the interceptor arguments in order to change the input to the next interceptor action in the chain. The ‘default’ order in which interceptor actions are called is defined according to the following rules.

1. Base class interceptor actions before subclass interceptor actions.
2. Named interceptor actions before unnamed interceptor actions.
3. Attribute interceptor actions before dynamic interceptor actions.
4. For attribute interceptor actions, in order of their occurrence in the code.
5. For dynamic interceptor actions, in the order that they were added to the *PropertyInterceptorManager*.

Because of the rigidity of these rules, there is also a provision to override the default order that any interceptor action is called by explicitly setting its ‘Order’ property. For attribute interceptors this is accomplished as follows:

```
[AfterGet(EntityPropertyNames.ContactName, Order=1)]
public void ContactNameAfterGet03(
  IbCore.PropertyInterceptorArgs<Customer, String> args) {
  ...
}
```

```
<BeforeSet(EntityPropertyNames.LastName, Order:=-1.0)> _
Public Sub UppercaseLastName(ByVal args As  _
  PropertyInterceptorArgs(Of Employee, String))
 '…
End Sub
```

The ‘Order’ property is defined as being of type ‘double’ and is automatically defaulted to a value of ‘0.0’. Any interceptor action with a property of less that ‘0.0’ will thus occur earlier than any interceptors without a specified order and any value greater that ‘0.0’ will correspondingly be called later, and in order of increasing values of the Order parameter. Exact ordering of interceptor actions can thus be accomplished.