

Rhino Mocks Quick Reference	
<b>Basic Flow</b>	<pre>' Create a MockRepository and objects to mock Dim mockery As New MockRepository Dim myView As IView = mockery.CreateMock(Of IView)() Dim myService As IService = mockery.CreateMock(Of IService)()  ' Create/Record expectations for these calls Expect.Call(myView.SelectedPersonUid).Return(Guid.Empty) Expect.Call(myService.GetPerson(Guid.Empty)).Return(New Person) Expect.Call(myView.LastName).PropertyBehavior()  mockery.ReplayAll() ' Switch to Replay mode  Dim myPresenter As New Presenter(myView, myService) myPresenter.Initialize() ' Call the method under test  ' Run assertions on Fakes and PropertyBehavior Mocks Assert.AreEqual("Enter Last Name", myView.LastName)  ' Verify all expectations mockery.VerifyAll()</pre>
Function	
Expect Call	Expect.Call(myService.GetPerson(Guid.Empty)).Return(New Person)
Expect Call Twice	Expect.Call(myService.GetPerson(Guid.Empty)).Return(New Person).Repeat.Twice()
Expect No Call	Expect.Call(myService.GetPerson(Guid.Empty)).Return(New Person).Repeat.Never()
Setup Result for arguments	SetupResult.For(myService.GetPerson(Guid.Empty)).Return(New Person)
Setup Result for all Arguments	SetupResult.For(myService.GetPerson(Guid.Empty)).IgnoreArguments.Return(New Person)
Sub	
Expect Call	myView.NavigateTo("EditPerson")
Expect Call but ignore arguments	myView.NavigateTo("") LastCall.IgnoreArguments()
Expect No Call	myView.NavigateTo("EditPerson") LastCall.Repeat.Never()
Setup Result	N/A
Property - Get	
Expect Call	Expect.Call(myView.LastName).Return("Smith")
Expect Call with constraints	Expect.Call(myView.LastName).Return("Smith").Repeat.Twice()
Expect No Call	Expect.Call(myView.LastName).Return("Smith").Repeat.Never()
Setup Result	SetupResult.For(myView.LastName).Return("Smith")
Property - Set	
Expect Call	myView.LastName = "Smith"
Expect Call Twice	myView.LastName = "Smith" LastCall.Repeat.Twice()
Expect No Call	myView.LastName = "Smith" LastCall.Repeat.Never()
Setup Result	N/A
<b>Property Behavior</b>	Expect.Call(myView.LastName).PropertyBehavior()