Contents

The ability to **intercept a validation** chain while it is executing and possibly change validation behaviors and/or results is a surprisingly common requirement. To accomplish this DevForce provides a pluggable delegate that, if defined, will be called after the execution of each verifier within the validation chain. This is called a *VerifierBatchInterceptor* .

BatchInterceptor

A *BatchInterceptor* property is defined on the *VerifierEngine*.

```
public virtual VerifierBatchInterceptor BatchInterceptor { get; set; }
```

```
Public Overridable Property BatchInterceptor() As VerifierBatchInterceptor
```

The *VerifierBatchInterceptor* delegate is defined as:

```
public delegate VerifierErrorContinuationMode VerifierBatchInterceptor(
  object instance,
   TriggerContext triggerContext,
   VerifierContext verifierContext
)
```

```
Public Delegate Function VerifierBatchInterceptor( _
 ByVal instance As Object, ByVal triggerContext As TriggerContext, _
 ByVal verifierContext As VerifierContext) As  _
  VerifierErrorContinuationMode _
 Implements VerifierEngine.VerifierBatchInterceptor
```

with the following *VerifierErrorContinuationMode* enum values:

Value	Description
Stop	Stop executing, any verifiers further down the chain will be skipped.
Continue	Continue executing
Inherit	(See *VerifierOptions* inheritance discussion)

Interception via the *BatchInterceptor* can be used to decide whether to continue with the current verification batch or to stop and return to the point in the program where the validation was triggered. The interception delegate also allows the currently executing validation's *VerifierResultCollection* to be modified. In other words, additional *VerifierResults* can be added or removed or summarized into a smaller number of *VerifierResults* after each verifier within the batch executes.

One example of its possible uses is to stop the verification process when the verification results already contains more than a maximum number of errors.

The following code sample below shows how you would create and use an interceptor.

```
public static VerifierErrorContinuationMode
  EmployeeBatchInterceptor(Object instance, TriggerContext
  triggerContext, VerifierContext verifierContext) {
 if (verifierContext.VerifierResults.Errors.Count > 10) {
    verifierContext.VerifierResults.Add(
     new VerifierResult(false,
       "Too many errors, stop immediately"));
   return VerifierErrorContinuationMode.Stop;
  } else {
   return VerifierErrorContinuationMode.Continue;
  }
}
public void DoEmployeeBatchInterceptor() {
  var mgr = new NorthwindIBEntityManager();
  mgr.VerifierEngine.BatchInterceptor = EmployeeBatchInterceptor;
  mgr.VerifierEngine.DefaultVerifierOptions.ExecutionModes =
    VerifierExecutionModes.InstanceAndOnAfterSetTriggers;
  mgr.VerifierEngine.DefaultVerifierOptions.ErrorNotificationMode =
    VerifierErrorNotificationMode.Notify;
  var employee = mgr.Employees.FirstOrNullEntity();
  employee.BirthDate = DateTime.Today.AddYears(7);
   //Verification batch will be executed here
 var results = mgr.VerifierEngine.Execute(employee);
```

```
  _localOutput.Append(results.Errors.Last().Message);
}
```

```
Public Shared Function EmployeeBatchInterceptor( _
 ByVal instance As Object, ByVal triggerContext_Renamed As  _
  TriggerContext, ByVal verifierContext_Renamed As VerifierContext) _
   As VerifierErrorContinuationMode
 If verifierContext_Renamed.VerifierResults.Errors.Count > 10 Then
    verifierContext_Renamed.VerifierResults.Add( _
     New VerifierResult(False, "Too many errors, stop immediately"))
   Return VerifierErrorContinuationMode.Stop
 Else
   Return VerifierErrorContinuationMode.Continue
 End If
End Function
Public Sub DoEmployeeBatchInterceptor()
 Dim mgr = New NorthwindIBEntityManager()
  mgr.VerifierEngine.BatchInterceptor = _
   AddressOf EmployeeBatchInterceptor
  mgr.VerifierEngine.DefaultVerifierOptions.ExecutionModes = _
    VerifierExecutionModes.InstanceAndOnAfterSetTriggers
  mgr.VerifierEngine.DefaultVerifierOptions.ErrorNotificationMode _
    = VerifierErrorNotificationMode.Notify
 Dim employee = mgr.Employees.FirstOrNullEntity()
  employee.BirthDate = Date.Today.AddYears(7)
 'Verification batch will be executed here
 Dim results = mgr.VerifierEngine.Execute(employee)
  _localOutput.Append(results.Errors.Last().Message)
End Sub
```

VerifierContext.EndOfBatch

In addition to being fired after each verifier, the *BatchInterceptor* is also fired at the very end of the current verification batch. This is to provide useful behavior such as analyzing the results of the verification batch. Below is an example of the *EmployeeBatchInterceptor* delegate above modified by adding a simple logic using the *VerifierContext.EndOfBatch* property. The logic adds a new *VerifierResult* to the *VerifierResultsCollection* with a *VerifierResultCode.Error* and a string message to indicate how many total errors are accumulated at the end of batch.

```
public static VerifierErrorContinuationMode EmployeeBatchInterceptor(
  Object instance, TriggerContext triggerContext,
  VerifierContext verifierContext) {
 if (verifierContext.EndOfBatch) {
    verifierContext.VerifierResults.Add(
     new VerifierResult(VerifierResultCode.Error,
      String.Format("Accumulated a total of {0} error(s)",
      verifierContext.VerifierResults.Errors.Count)));
   return VerifierErrorContinuationMode.Stop;
  } else {
   if (verifierContext.VerifierResults.Errors.Count > 3) {
      verifierContext.VerifierResults.Add(new VerifierResult(
        VerifierResultCode.Error,
       "Too many errors, stop verification immediately"));
     return VerifierErrorContinuationMode.Stop;
    } else {
     return VerifierErrorContinuationMode.Continue;
    }
  }
}
```

```
Public Shared Function EmployeeBatchInterceptor(ByVal instance _
 As Object, ByVal triggerContext_Renamed As TriggerContext, _
 ByVal verifierContext_Renamed As VerifierContext) As  _
  VerifierErrorContinuationMode
 If verifierContext_Renamed.EndOfBatch Then
    verifierContext_Renamed.VerifierResults.Add( _
     New VerifierResult(VerifierResultCode.Error, _
     String.Format("Accumulated a total of {0} error(s)", _
      verifierContext_Renamed.VerifierResults.Errors.Count)))
   Return VerifierErrorContinuationMode.Stop
 Else
   If verifierContext_Renamed.VerifierResults.Errors.Count > 3 Then
```

```
      verifierContext_Renamed.VerifierResults.Add( _
       New VerifierResult(VerifierResultCode.Error, _
       "Too many errors, stop verification immediately"))
      Return VerifierErrorContinuationMode.Stop
    Else
      Return VerifierErrorContinuationMode.Continue
    End If
  End If
End Function
```