# Working with arrays

You can edit the code for action parameters that involve arrays of objects.
Many action parameters in Prinergy require lists (or arrays) of objects. For example, the **Refine** action needs a list of input files to refine, and the **Perform Loose Page Output** action needs a list of pages. Many event properties are also lists. For example, the **Page Approval Changed** event provides a list of pages where the approval status has changed.
Working with these arrays in code is considerably more complex than working with simple assignments.

### Simple array assignment

When you want to pass all the members of an event property array to an action parameter array, the code looks the same as a simple assignment:

```
'This takes ALL the input files from the event and gives them to the action
action.InputFiles = triggerEvent.InputFiles
```

### Using ArrayLists

ArrayLists are useful because they are untyped and unsized . They will expand as you add to them and you can put anything in them.

```
'Create an ArrayList and add items to it
Dim oArray As System.Collections.ArrayList = New
System.Collections.ArrayList
oArray.AddRange(System.IO.Directory.GetFiles("c:\", "*.pdf"))
oArray.Add(someOtherObject)
```

### Converting ArrayLists to other arrays

After you have prepared the `ArrayList`, you probably need to convert it into other arrays to be used within the rules—arrays that are typed and sized.
`ArrayList` has a `ToArray` method that will copy the elements of the `ArrayList` to an array of a given type.

```
' Converts the ArrayList into an Array of Automation Pages
Dim oAlist As System.Collections.ArrayList ' . . . fill with contents
action.Pages =
oAList.ToArray(GetType(Creo.PWS.Automation.PrinergyDataModel.Page))
```

The `ToArray` method only works if the contents of the `ArrayList` can be cast to the type you specify. If not, a runtime error appears. You must manually convert each item in the `ArrayList` to the array. For example, you cannot cast a `String` to a `FileSystemItem`, but you can use the `CreateFrom` factory method to create a `FileSystemItem` from a `String`.

```
' Converts the ArrayList into an Array of Automation FileSystemItems.
Dim oFSIArray() As Creo.PWS.Automation.PrinergyDataModel.FileSystemItem = _
New Creo.PWS.Automation.PrinergyDataModel.FileSystemItem(oAList.count
- 1) {}
For i As Integer = 0 To oAList.Count - 1
oFSIArray(i) = _
Creo.PWS.Automation.PrinergyDataModel.FileSystemItem.CreateFrom(oAList(i).T
oString)
Next i
```

**Flattening nested arrays**

Sometimes the data that you want to assign to an action parameter are spread over the members of an array in the event properties. For example, the results of a **Refine** action are available from an **Input Files Refine OK** event. When you look into that event, you see that it has the following properties:

| | |
|---|---|
| Input Files Refined Successfully | The list of input files that refined successfully |
| Process | Refine process |
| Event time | Time that the event was processed |
| Root event | First event in the sequence |
| Previous Event | Previous event in the sequence |

Where are the pages? Each of the refined input files has its own list of pages that were created from it. In other words, each member of the `InputFiles` array contains an array of pages.
To pass a list of all the refined pages to an action such as **Perform Loose Page Output**, we need a way to reach into each input file and build a collection of all the pages.

```
'First we need to create a holder for the new list we will be building
Dim newList As System.Collections.ArrayList = New
System.Collections.ArrayList
'Now we create a counter variable and initialize to the value 0
Dim i As Integer = 0
'Next we create a loop that will go through all the input files (until
'the counter i reaches the end of the list)
Do While (i < triggerEvent.InputFiles.Length)
'We reach into the ith Input File and add its Pages array to our
list
newList.AddRange(triggerEvent.InputFiles(i).Pages)
'Increment the value of i
i = (i + 1)
Loop
'Finally, we convert our list into an Array of Pages and assign it to the
'Pages parameter of the action
action.Pages =
newList.ToArray(GetType(Creo.PWS.Automation.PrinergyDataModel.Page))
```

**Filtering an array**

Another common requirement is the ability to filter parts of a list that you are not interested in. For example, you might want a simple way to create imposition proofs of only the front surfaces of every signature in your job. You want to pass to the **Imposed Proof** action a list of surfaces that has all the back surfaces filtered out.

```
'First we need to create a holder for the new list we will be building
Dim newList As System.Collections.ArrayList = New
System.Collections.ArrayList
'Now we create a counter variable and initialize to the value 0
Dim i As Integer = 0
'Next we create a loop that will go through all the input files (until
'the counter i reaches the end of the list)
Do While (i < triggerEvent.Surfaces.Length)
'We look at the ith Surface and add it to our list if it is “Front”
If triggerEvent.Surfaces(i).Side = "Front" Then
newList.AddRange(triggerEvent.Surfaces(i))
End If
'Increment the value of i
i = (i + 1)
Loop
'Finally, we convert our list into an Array of Surfaces and assign it to
the
'Surfaces parameter of the action
action.Surfaces =
newList.ToArray(GetType(Creo.PWS.Automation.PrinergyDataModel.Surface))
```

**Filtering and flattening arrays**

Combining the two patterns for flattening and filtering gives us the most power of all. In this example, we want a list of all the refined pages that are larger than 8.5 x 11. Since we need to look at every page in order to determine its size, this requires us to introduce a second loop. The outer loop goes through all the input files, while the inner (nested) loop goes through all the refined pages for each input file.

```
'First we need to create a holder for the new list we will be building
Dim newList As System.Collections.ArrayList = New System.Collections.ArrayList

'Now we create a counter variable and initialize to the value 0
Dim i As Integer = 0

'Next we create a loop that will go through all the input files (until
'the counter i reaches the end of the list)
Do While (i &lt; triggerEvent.InputFiles.Length)

'Create a second counter and make a second loop to go through all
' the pages for input file i
Dim j As Integer = 0
Do While (j &lt; triggerEvent.InputFiles(i).Pages.Length)

'Grab the jth Page from the ith Input File
Dim thePage As Page = triggerEvent.InputFiles(i).Pages(j)

'See if the page is larger than 8.5 x 11 (in points) – if so add it to our list
If (thePage.TrimSize.x &gt; 8.5*72 AND thePage.TrimSize.y &gt; 11*72) Then
newList.add(thePage
End If

'Increment the value of j
j = (j + 1)
Loop

'Increment the value of i
i = (i + 1)

Loop

'Finally, we convert our list into an Array of Pages and assign it to the
'Pages parameter of the action
action.Pages = newList.ToArray(GetType(Creo.PWS.Automation.PrinergyDataModel.
Page))
```