Action Listeners

This chapter describes the J-Link methods that enable you to use action listeners.

Topic

J-Link Action Listeners

An `ActionListener` in Java is a class that is assigned to respond to certain events. In J-Link, you can assign action listeners to respond to events involving the following tasks:

1. Changing windows
2. Changing working directories
3. Model operations
4. Regenerating
5. Creating, deleting, and redefining features
6. Checking for regeneration failures

All action listeners in J.Link are defined by these classes:

1. Interface--Named <Object>ActionListener. This interface defines the methods that can respond to various events.
2. Default class--Named Default<Object>ActionListener. This class has every available method overridden by an empty implementation. You create your own action listeners by extending the default class and overriding the methods for events that interest you.

Creating an ActionListener Implementation

You can create a proper `ActionListener` class using either of the following methods:

1. Define a separate class within the java file.

```
For example:
    public class MyApp {
        session.AddActionListener (new SolidAL1());
    }

    class SolidAL1 extends DefaultSolidActionListener {
        // Include overridden methods here
    }
```

1. To use your action listener in different Java applications, define it in a separate file.

   For example:

```
MyApp.java:
import solidAL1;

public class MyApp {
    session.AddActionListener (new SolidAL1());
}

SolidAL1.java:
public class SolidAL1 extends DefaultSolidActionListener {
    // Include overridden methods here.
}
```

Action Sources

Methods introduced:

- **pfcBase.ActionSource.AddActionListener**
- **pfcBase.ActionSource.RemoveActionListener**

Many J-Link classes inherit the `ActionSource` interface, but only the following classes currently make calls to the methods of registered `ActionListeners`:

1. pfcSession.Session

   - Session Action Listener
   - Model Action Listener
   - Solid Action Listener
   - Model Event Action Listener
   - Feature Action Listener
2. pfcCommand.UICommand

   - UI Action Listener
3. pfcModel.Model (and it's subclasses)

   - Model Action Listener
   - Parameter Action Listenerr
4. pfcSolid.Solid (and it's subclasses)

   - Solid Action Listener
   - Feature Action Listener
5. pfcFeature.Feature (and it's subclasses)

   - Feature Action Listener

Note:
Assigning an action listener to a source not related to it will not cause an error but the listener method will never be called.

Types of Action Listeners

The following sections describe the different kinds of action listeners: session, UI command, solid, and feature.

Session Level Action Listeners

Methods introduced:

- **pfcSession.SessionActionListener.OnAfterDirectoryChange**
- **pfcSession.SessionActionListener.OnAfterWindowChange**
- **pfcSession.SessionActionListener.OnAfterModelDisplay**
- **pfcSession.SessionActionListener.OnBeforeModelErase**
- **pfcSession.SessionActionListener.OnBeforeModelDelete**

- pfcSession.SessionActionListener.OnBeforeModelRename
- pfcSession.SessionActionListener.OnBeforeModelSave
- pfcSession.SessionActionListener.OnBeforeModelPurge
- pfcSession.SessionActionListener.OnBeforeModelCopy
- pfcSession.SessionActionListener.OnAfterModelPurge

The **pfcSession.SessionActionListener.OnAfterDirectoryChange** method activates after the user changes the working directory. This method takes the new directory path as an argument.

The **pfcSession.SessionActionListener.OnAfterWindowChange** method activates when the user activates a window other than the current one. Pass the new window to the method as an argument.

The **pfcSession.SessionActionListener.OnAfterModelDisplay** method activates every time a model is displayed in a window.

Note:
Model display events happen when windows are moved, opened and closed, repainted, or the model is regenerated. The event can occur more than once in succession.

The methods **pfcSession.SessionActionListener.OnBeforeModelErase**, **pfcSession.SessionActionListener.OnBeforeModelRename**, **pfcSession.SessionActionListener.OnBeforeModelSave**, and **pfcSession.SessionActionListener.OnBeforeModelCopy** take special arguments. They are designed to allow you to fill in the arguments and pass this data back to Pro/ENGINEER. The model names placed in the descriptors will be used by Pro/ENGINEER as the default names in the UI.

UI Command Action Listeners

Methods introduced:

- pfcSession.Session.UICreateCommand
- pfcCommand.UICommandActionListener.OnCommand

The **pfcSession.Session.UICreateCommand** method takes a `UICommandActionListener` argument and returns a `UICommand` action source with that action listener already registered. This `UICommand` object is subsequently passed as an argument to the **Session.AddUIButton** method that adds a command button to a Pro/ENGINEER menu. The **pfcCommand.UICommandActionListener.OnCommand** method of the registered `UICommandActionListener` is called whenever the command button is clicked.

Model Level Action listeners

Methods introduced:

- pfcModel.ModelActionListener.OnAfterModelSave
- pfcModel.ModelEventActionListener.OnAfterModelCopy
- pfcModel.ModelEventActionListener.OnAfterModelRename
- pfcModel.ModelEventActionListener.OnAfterModelErase
- pfcModel.ModelEventActionListener.OnAfterModelDelete
- pfcModel.ModelActionListener.OnAfterModelRetrieve
- pfcModel.ModelActionListener.OnBeforeModelDisplay
- pfcModel.ModelActionListener.OnAfterModelCreate

- **pfcModel.ModelActionListener.OnAfterModelSaveAll**
- **pfcModel.ModelEventActionListener.OnAfterModelCopyAll**
- **pfcModel.ModelActionListener.OnAfterModelEraseAll**
- **pfcModel.ModelActionListener.OnAfterModelDeleteAll**
- **pfcModel.ModelActionListener.OnAfterModelRetrieveAll**

Methods ending in All are called after any event of the specified type. The call is made even if the user did not explicitly request that the action take place. Methods that **do not** end in All are only called when the user specifically requests that the event occurs.

The method pfcModel.ModelActionListener.OnAfterModelSave is called after successfully saving a model.

The method pfcModel.ModelEventActionListener.OnAfterModelCopy is called after successfully copying a model.

The method pfcModel.ModelEventActionListener.OnAfterModelRename is called after successfully renaming a model.

The method pfcModel.ModelEventActionListener.OnAfterModelErase is called after successfully erasing a model.

The method pfcModel.ModelEventActionListener.OnAfterModelDelete is called after successfully deleting a model.

The method pfcModel.ModelActionListener.OnAfterModelRetrieve is called after successfully retrieving a model.

The method pfcModel.ModelActionListener.OnBeforeModelDisplay is called before displaying a model.

The method **pfcModel.ModelActionListener.OnAfterModelCreate** is called after the successful creation of a model.

Example Code

This example demonstrates the use of two J-Link listener methods (OnAfterModelCopy and OnAfterModelRename in ModelEventActionListener). It uses these listeners to create or update parameters in the copied or renamed models which tell the history of the last event that happened to the model.

```
package com.ptc.jlinkexamples;

import com.ptc.cipjava.*;
import com.ptc.pfc.pfcGlobal.*;
import com.ptc.pfc.pfcSession.*;
import com.ptc.pfc.pfcModel.*;
import com.ptc.pfc.pfcModelItem.*;

public class pfcModelEventExamples
{

  public static void addEventListener (Session s) throws jxthrowable
    {
     s.AddActionListener (new ModelEventListenerExample ());
    }
}

class ModelEventListenerExample
     extends com.ptc.pfc.pfcModel.DefaultModelEventActionListener
```

```
{
 public void OnAfterModelCopy (ModelDescriptor From,
                        ModelDescriptor To) throws jxthrowable
   {
     Session s = pfcGlobal.GetProESession ();
     Model m = s.GetModelFromDescr (To);

     if (m == null)
       {
        /* Couldn't find the new model handle - might happen, for example, when a copy is saved to disk but no
        return;
       }

     ParamValue pv =
       pfcModelItem.CreateStringParamValue (From.GetFullName().toUpperCase());
     Parameter p = m.GetParam ("COPIED_FROM");
     if (p == null)
     m.CreateParam ("COPIED_FROM", pv);
     else
     p.SetValue (pv);
   }

 public void OnAfterModelRename (ModelDescriptor From,
 ModelDescriptor To) throws jxthrowable
   {
     Session s = pfcGlobal.GetProESession ();
     Model m = s.GetModelFromDescr (To);

     if (m == null)
       {
        /* Couldn't find the new model handle in memory */
        return;
       }

     ParamValue pv =
     pfcModelItem.CreateStringParamValue
     (From.GetFullName().toUpperCase());
     Parameter p = m.GetParam ("RENAMED_FROM");
     if (p == null)
     m.CreateParam ("RENAMED_FROM", pv);
     else
       p.SetValue (pv);
   }
}
```

Solid Level Action Listeners

Methods introduced:

- **pfcSolid.SolidActionListener.OnBeforeRegen**
- **pfcSolid.SolidActionListener.OnAfterRegen**
- **pfcSolid.SolidActionListener.OnBeforeUnitConvert**
- **pfcSolid.SolidActionListener.OnAfterUnitConvert**
- **pfcSolid.SolidActionListener.OnBeforeFeatureCreate**
- **pfcSolid.SolidActionListener.OnAfterFeatureCreate**

- **pfcSolid.SolidActionListener.OnAfterFeatureDelete**

The **pfcSolid.SolidActionListener.OnBeforeRegen** and **pfcSolid.SolidActionListener.OnAfterRegen** methods occur when the user regenerates a solid object within the `ActionSource` to which the listener is assigned. These methods take the first feature to be regenerated and a handle to the `Solid` object as arguments. In addition, the method **pfcSolid.SolidActionListener.OnAfterRegenerate** includes a Boolean argument that indicates whether regeneration was successful.

Note:
It is not recommended to modify geometry or dimensions using the pfcSolid.SolidActionListener.OnBeforeRegenerate method call.

Note:
A regeneration that did not take place because nothing was modified is identified as a regeneration failure.

The **pfcSolid.SolidActionListener.OnBeforeUnitConvert** and **pfcSolid.SolidActionListener.OnAfterUnitConvert** methods activate when a user modifies the unit scheme (by selecting the Pro/ENGINEER command **Set Up**, **Units**). The methods receive the `Solid` object to be converted and a Boolean flag that identifies whether the conversion changed the dimension values to keep the object the same size.

Note:
SolidActionListeners can be registered with the session object so that its methods are called when these events occur for any solid model that is in session.

The **pfcSolid.SolidActionListener.OnBeforeFeatureCreate** method activates when the user starts to create a feature that requires the Feature Creation dialog box. Because this event occurs only after the dialog box is displayed, it will not occur at all for datums and other features that do not use this dialog box. This method takes two arguments: the solid model that will contain the feature and the `ModelItem` identifier.

The **pfcSolid.SolidActionListener.OnAfterFeatureCreate** method activates after any feature, including datums, has been created. This method takes the new `Feature` object as an argument.

The **pfcSolid.SolidActionListener.OnAfterFeatureDelete** method activates after any feature has been deleted. The method receives the solid that contained the feature and the (now defunct) `ModelItem` identifier.

Feature Level Action Listeners

Methods introduced:

- **pfcFeature.FeatureActionListener.OnBeforeDelete**
- **pfcFeature.FeatureActionListener.OnBeforeSuppress**
- **pfcFeature.FeatureActionListener.OnAfterSuppress**
- **pfcFeature.FeatureActionListener.OnBeforeRegen**
- **pfcFeature.FeatureActionListener.OnAfterRegen**
- **pfcFeature.FeatureActionListener.OnRegenFailure**
- **pfcFeature.FeatureActionListener.OnBeforeRedefine**
- **pfcFeature.FeatureActionListener.OnAfterCopy**
- **pfcFeature.FeatureActionListener.OnBeforeParameterDelete**

Each method in `FeatureActionListener` takes as an argument the feature that triggered the event.

`FeatureActionListeners` can be registered with the Session object so that the action listener's methods are called whenever these events occur for any feature that is in session or with a solid model to react to changes only in that model.

The method **pfcFeature.FeatureActionListener.OnBeforeDelete** is called before a feature is deleted.

The method **pfcFeature.FeatureActionListener.OnBeforeSuppres**s is called before a feature is suppressed.

The method **pfcFeature.FeatureActionListener.OnAfterSuppress** is called after a successful feature suppression.

The method **pfcFeature.FeatureActionListener.OnBeforeRegen** is called before a feature is regenerated.

The method **pfcFeature.FeatureActionListener.OnAfterRegen** is called after a successful feature regeneration.

The method **pfcFeature.FeatureActionListener.OnRegenFailure** is called when a feature fails regeneration.

The method **pfcFeature.FeatureActionListener.OnBeforeRedefine** is called before a feature is redefined.

The method **pfcFeature.FeatureActionListener.OnAfterCopy** is called after a feature has been successfully copied.

The method **pfcFeature.FeatureActionListener.OnBeforeParameterDelete** is called before a feature parameter is deleted.
