**SUMMARY**

A concurrency bug in Eclipse 3.4 Platform Debug.

**DETAILS**

Some details can also be found at: [https://bugs.eclipse.org/bugs/show\_bug.cgi?id=246148](https://bugs.eclipse.org/bugs/show_bug.cgi?id=57467)

This bug is due to a data race.

In org.eclipse.debug.internal.core.WatchExpression#evaluate(), there is a race condition: when an expression is done evaluating, the following sequence of events takes place:

- The "pending" flag is cleared

- A DebugEvent is broadcast indicating a state change (since it is no longer pending)

- The WatchExpression's result is stored in fResult

- Another DebugEvent is broadcast indicating a content change (since the result is now present)

The code is simple: it calls setPending(false), and then setResult(result). setPending() sends out the first DebugEvent, and setResult() sends out the second one.

The result of this sequence of events is that for a brief time, the WatchExpression is marked as "not pending" and yet its result has not yet been set. This contradicts the JavaDoc for isPending(), which seems to indicate that if isPending() returns false, then a result is available:

|  |  |
| --- | --- |
| Thread1 (WatchExpression.java) | Thread2 (WatchExpression.java) |
| public void evaluate() {  IDebugElement context= fCurrentContext;  if (context == null) {  return;  }    IWatchExpressionListener listener= new IWatchExpressionListener() {  /\* (non-Javadoc)  \* @see org.eclipse.debug.core.model.IWatchExpressionListener#watchEvaluationFinished(org.eclipse.debug.core.model.IWatchExpressionResult)  \*/  public void watchEvaluationFinished(IWatchExpressionResult result) {  setPending(false);  setResult(result);  }  };  setPending(true);  IWatchExpressionDelegate delegate= DebugPlugin.getDefault().getExpressionManager().newWatchExpressionDelegate(context.getModelIdentifier());  if (delegate != null) {  delegate.evaluateExpression(getExpressionText(), context, listener);  } else {  // No delegate provided  listener.watchEvaluationFinished(new IWatchExpressionResult() {  public IValue getValue() {  return null;  }  public boolean hasErrors() {  return true;  }  public String[] getErrorMessages() {  return new String[] { DebugCoreMessages.WatchExpression\_0 };  }  public String getExpressionText() {  return WatchExpression.this.getExpressionText();  }  public DebugException getException() {  return null;  }  });  }  } | /\*\*  \* Returns whether the result of this watch expression is pending.  \* An expression is pending if an evaluation has been requested, but  \* the value has not yet been returned.  \*  \* @return whether this expression's result is pending  \*/  public boolean isPending() {  return fPending;  } |
| protected void setPending(boolean pending) {  fPending= pending;  fireEvent(new DebugEvent(this, DebugEvent.CHANGE, DebugEvent.STATE));  } | public void setResult(IWatchExpressionResult result) {  fResult= result;  fireEvent(new DebugEvent(this, DebugEvent.CHANGE, DebugEvent.CONTENT));  } |
| 0-lock | 0-lock |