

ISIS - Bug #4206

Intermittent failure cnetedit app test

2016-08-05 04:30 PM - Jean Backer

Status: Resolved	
Priority: Normal	
Assignee: Ian Humphrey	
Category: Infrastructure	
Target version: FY17 Backlog	
Impact:	Software Version:
Description	
<i>Spike (investigation) Closed (1 story point)</i>	
The cnetedit checkValid app test [log2.txt] fails intermittently.	
IPCE should be built using debug and the following command should be run:	
<pre>valgrind cnetedit \ CNET=\$(INPUT)/cnet.net \ ONET=\$(OUTPUT)/cnet2.net \ LOG=\$(OUTPUT)/log2.txt \ CHECKVALID=yes \ IGNOREALL=true \ FROMLIST=\$(OUTPUT)/list.lis \ DEFFILE=\$(INPUT)/deffile.def</pre>	

History

#1 - 2016-08-05 04:43 PM - Jean Backer

Sample truth and output folders can be found for comparison in [/work/projects/isis/latest/data/m04206]

#2 - 2016-08-12 07:22 AM - Kenneth Edmundson

- Target version set to Sprint 6 - The Desperate Hours

#3 - 2016-08-12 11:05 AM - Jean Backer

- Story points set to 4

- Status changed from New to Acknowledged

#4 - 2016-08-16 03:01 PM - Ian Humphrey

- Assignee set to Ian Humphrey

#5 - 2016-08-16 03:01 PM - Ian Humphrey

- Status changed from Acknowledged to Assigned

#6 - 2016-08-16 03:16 PM - Ian Humphrey

- Status changed from Assigned to In Progress

#7 - 2016-08-18 12:33 PM - Ian Humphrey

- Status changed from In Progress to Resolved

This is most likely NOT and IPCE issue.

I ran the checkValid test 300 times in trunk and could not get a failure.

I also ran the checkValid test in an up-to-date checkout of v006LibrariesV2, and I could get intermittent failures about every 5 times.

Since it fails in v006LibrariesV2, this issue should probably be reconsidered as a maintenance issue.

#8 - 2016-08-18 04:02 PM - Ian Humphrey

- Project changed from Jigsaw to ISIS
- Category set to Infrastructure
- Status changed from Resolved to In Progress
- Target version deleted (Sprint 6 - The Desperate Hours)

#9 - 2016-10-03 10:44 AM - Stuart Sides

- Target version set to FY17 Backlog

#12 - 2016-10-04 04:00 PM - Ian Humphrey

Overview

There are essentially two tests performed in cnetedit [checkValid]:

First, cnetedit is run with `CHECKVALID=yes` with `LOG=log.txt`. This test works fine and does not have intermittent failures.

```
cnetedit CNET=input/cnet.net ONET=output/cnet.net LOG=output/log.txt CHECKVALID=yes \  
FROMLIST=output/list.lis DEFFILE=input/deffile.def RETAIN_REFERENCE=true
```

Second, cnetedit is run with `CHECKVALID=yes`, `LOG=log2.txt`, and `IGNOREALL=true`. This test has intermittent failures.

```
cnetedit CNET=cnet.net ONET=output/cnet2.net LOG=output/log2.txt CHECKVALID=yes \  
IGNOREALL=true FROMLIST=output/list.lis DEFFILE=input/deffile.def
```

From this second command line and the xml, we can extrapolate more parameters being set implicitly:
`IGNORE=true`, `DELETE=true`, `PRESERVE=false`, `RETAIN_REFERENCE=false`

So, the second command line essentially means:

I want to edit cnet.net and I will call the edited control network 'cnet2.net.' I will set the output log information to be written to 'log2.txt.' I want to perform a validity check, so I'll provide the control network images in list.lis. I am going to determine whether a control measure is valid by examining if the tolerance criteria defined in deffile.def are satisfied. If a control measure does not meet these tolerance criteria, I will ignore that control measure. If the control measure being ignored is a reference measure, the entire associated control point AND all the point's measures are ignored. I want to delete any ignored points and ignored measures so they are not in my edited control network, cnet2.net.

IGNOREALL and Associated Parameters in cnetedit.cpp

The following parameters are grabbed from the UI and stored in *global* booleans:

- deleteIgnored = DELETE (**true**)
- preservePoints = PRESERVE (**false**)
- retainRef = RETAIN_REFERENCE (**false**)
- ignoreAll = IGNOREALL (**true**)

Difference between output/log2.txt and truth/log2.txt

When cnetedit [checkValid] fails, the output and truth for log2.txt have the following difference (*indicated by "<<<<"*)

```
output/log2.txt:
```

```
Object = Measures
```

```

Group = pointregTest0001
  MRO/HIRISE/856864216:41044/RED5/2 = "Validity Check failed:  Sample      <<<
                                         Shift is greater than tolerance 3      <<<
                                         Line Shift is greater than tolerance  <<<
                                         5"                                     <<<

  MRO/HIRISE/856864216:41044/RED4/2 = "Validity Check failed:  Sample
                                         Residual is greater than tolerance
                                         5"

End_Group
...

```

```

=====
truth/log2.txt:

```

```

Object = Measures
  Group = pointregTest0001
    MRO/HIRISE/856864216:41044/RED4/2 = "Validity Check failed:  Sample
                                         Residual is greater than tolerance
                                         5"

    MRO/HIRISE/856864216:41044/RED5/2 = "Reference ignored"      <<<
  End_Group
...

```

Why is this happening?

In `cnedit.cpp`'s `main()`, we have a condition that checks if `ignore == true`. We enter this block for this failing test.

Within this block, there are three checks that are NOT relevant to us, since we do not specify `POINTLIST`, `CUBELIST`, or `MEASURLIST` in this test.

Still within this `ignore` block, we check if `CHECKVALID` was entered, and it was, so we enter this block.

Now, we set up a `ControlNetValidMeasure` (a measure validator) with the `DEFFILE` that was passed.

Next, we call `checkAllMeasureValidity(cnet, FROMLIST)`.

This function, `void checkAllMeasureValidity(ControlNet &cnet, QString cubeList)`, is responsible for validating the measures in the control network against the `DEFFILE` tolerance that were established.

In this function, we create a `SerialNumberList` from the passed `cubeList` (`FROMLIST`).

We then get the `ControlCubeGraphNode`'s from the control network as follows:

THIS LINE BELOW IS RESPONSIBLE

```
QList graphNodes = cnet.GetCubeGraphNodes();
```

Internally, `ControlNet::GetCubeGraphNodes()` is implemented:
`return cubeGraphNodes->values();`

cubeGraphNodes is a QHash, so the order of the returned graph nodes is NOT guaranteed!

```
(QHash< QString, ControlCubeGraphNode * >)
```

Order Matters

If we look at the logic of the `checkAllMeasureValidity()` function, we can see the following structure:

```

for each serial number:
  // set up the graph node according to the serial number
  // set up other cube/camera related memory for validation, if required

  get a list of control measures from the graph node

  for each measure:

    // Check the measure's validity if it is not ignored to determine if we want it in the output control net
    if measure NOT ignored:

```

```

    results = validate the measure

    if the results NOT valid:
        get the failure message from the results

        if measure is reference and retainRef == true:
            //... for our test, we won't reach this since RETAIN_REFERENCE=false

        else: // always this for our test case
            ignore the measure with the failure message

            if measure is reference measure AND IGNOREALL == true:
                // ignorePoint() makes a call, point->SetIgnored(true), which will set all point's measures to ignored.
                ignore the point with "Reference measure ignored" message

```

Since we are relying on a QHash that arbitrarily determines the order of the cube graph nodes, the order of our measures being checked will be different each run.

If we check the validity of ...RED4/2 first, we match the truth/log2.txt data. This happens because we first ignore the measure with "Sample Residual > 5..." logged as its validation failure message. Then, **since this is a reference measure**, we ignore the **entire** point with the reason "Reference measure ignored" (this reason appears above the difference we encounter). Then, once we examine ...RED5/2, **we don't need to validate it since it has already been ignored since the parent point was just ignored earlier**.

If we check the validity of ...RED5/2 first, we do not match truth/log2.txt data. This happens because we ignore this measure and log the failure message/reason "Sample Shift > ..." Then, we check the validity of ...RED4/2, and we then ignore this measure and log the failure message "Sample Residual > 5..."

Important: If a reference measure is not valid, we ignore it, the parent control point, and ALL of the parent control point's measures. Once a measure is ignored, we do not need to check its validity anymore (i.e. we will not enter the "if measure NOT ignored" block).

Solution(s)?

- Check the validity of the reference measures first?
- Use a QMap instead of QHash (possibly undesirable solution, since this would require modifying ControlNet!)
- The log2.txt file is structured as a PVL (except for the first two lines), so possibly set up a diff file that ignores the "MRO/HIRISE/.../RED5/2" keyword in the pointregTest0001 PvlGroup.

#13 - 2016-10-05 11:11 AM - Ian Humphrey

- *Description updated*

#14 - 2016-10-14 12:21 PM - Ian Humphrey

- *Status changed from In Progress to Resolved*