Data Quality Committee - Test Cases

Introduction

Test cases help software developers implement draft and approved rules published by the Data Quality Committee (DQC). These test cases include pass and fail examples for each rule developed. The test cases are subject to change and will improve as exception cases are identified. The suite of updated test cases is released with each subsequent release of DQC rules. The latest version should be used to test that software implementations of the rules are deployed properly and operating as expected in different software environments.

How do the test cases work?

The test cases may include XBRL files that are created to test a specific DQC rule, as well as URL references to actual SEC filings. The zip file of test cases includes three types of files.

1. An index file listing the location of the test cases.
2. A variation file that details the test case and the expected results.
3. Custom XBRL files that either pass or fail a test case that are referenced by the variation file.

The Index File

The index file lists the location of the variation files documenting the test case and expected results. The index file lists the location(s) of the variation file(s) in the directory structure. An example of an index file is below:

```
<?xml version="1.0" encoding="utf-8"?>
<!-- Copyright 2015 XBRL US Inc. All Rights Reserved. →
<?xml-stylesheet type="text/xsl" href="."?>
<documentation name="Conformance suite tests for DQC Rules" date="2015-11-09">
  <testcases title="DQC Rules" root=".">
    <testcase uri="DQC_0004/variation.xml" />
    <testcase uri="DQC_0005/variation.xml" />
    <testcase uri="DQC_0006/variation.xml" />
    <testcase uri="DQC_0009/variation.xml" />
    <testcase uri="DQC_0015/variation.xml" />
    <testcase uri="DQC_0033/variation.xml" />
    <testcase uri="DQC_0036/variation.xml" />
  </testcases>
</documentation>
```

The Variation File

The variation file describes the rule, the associated test case and the expected results. There is a variation file for each rule, so if there are 6 rules there will be 6 variation files. A single rule can run for thousands of elements, so the variation file may be large, as it details a test case for each impacted element.

Structure of the file

The variation file is broken down into the following sections

1. Rule Header
2. Variation Section

The rule header lists the creator of the test cases, the DQC rule number (ie. DQC_0009), rule identifier/name (ie. Element A must be less than or equal to element B)), a description of the rule and an example of what the rule message will look like with variables showing.
Each variation section begins and ends with a <variation> tag and is a separate test case for the rule. An example of variation structure is below:

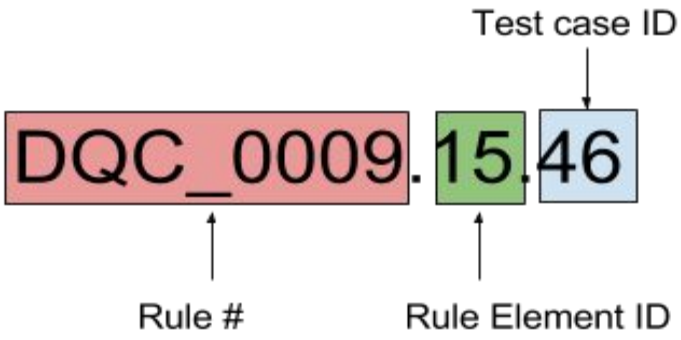
```
<variation id="DQC_0015.200.112">
    <name>Flags a fail case when the element
    us-gaap:AcceleratedShareRepurchasesFinalPricePaidPerShare nonneg items have an
    inappropriate negative value.</name>
    <description>Creates a fail case when the value of
    us-gaap:AcceleratedShareRepurchasesFinalPricePaidPerShare is less than
    zero.</description>
     <data>
       <schema>DQC_0015.200/CASE_112/testco-20141231_112.xsd</schema>
       <instance readMeFirst="true">DQC_0015.200/CASE_112/testco-20141231_112.xml</instance>
    </data>
    <results>
      <error severity="error" count="1">DQC.US.0015.200</error>
       <result>
       <primaryElement>us-gaap:AcceleratedShareRepurchasesFinalPricePaidPerShare</primaryEl
       ement>
        <factValue>-11.683</factValue>
        <startDate>2014-01-01</startDate>
        <endDate>2014-12-31</endDate>
        <dimensions />
        <message>${fact1.label} has a value of -11.683 which is less than zero. This element
        should not have a negative value. The properties of this
        us-gaap:AcceleratedShareRepurchasesFinalPricePaidPerShare fact are:
        Period: 2014-12-31 - 2014-12-31
        Dimensions: none
        Unit: USD/Share
        Rule version: ${ruleVersion}</message>
      </result>
    </results>
</variation>
```

Each variation has its own identifier (shown in red as id)

```
<variation id="DQC_0015.200.112">
```

The variation identifier is comprised of the rule number the rule element id and the test case number. The figure below illustrates the elements of a variation identifier:

Figure 1 Elements of a Variation Identifier



The first field in the variation section is the name of the test case.

```
<variation id="DQC_0015.200.112">
    <name>Flags a fail case when the element
    us-gaap:AcceleratedShareRepurchasesFinalPricePaidPerShare nonneg items have an
    inappropriate negative value.</name>
```

This is a human readable name that explains the fact situation in the instance file such as *"CommonStockSharesOutstanding is greater than CommonStockSharesAuthorized"*. The second field is the description which defines if the test case defines a fail or pass case depending on the condition. This is meant to be human readable.

```
<variation id="DQC_0015.200.112">
    <description>Creates a fail case when the value of
    us-gaap:AcceleratedShareRepurchasesFinalPricePaidPerShare is less than zero.</description>
```

Data

The next section of the variation section is the data section. This defines where the data for the test case case can be found -the schema file and instance file.

```
<variation id="DQC_0015.200.112">
    <data>
      <schema>DQC_0015.200/CASE_112/testco-20141231_112.xsd</schema>
      <instance readMeFirst="true">DQC_0015.200/CASE_112/testco-20141231_112.xml</instance>
    </data>
```

This data will either be a relative reference to a local resource or URL to the location of a xml file or zip file. An attribute called *"readMeFirst"* (in red) with a value of true is set on the data field that tells an XBRL processor which file should be used to access the test case. If the attribute does not appear then it is the same as false.

In some variations, SEC files are used for the test case. In variations where SEC files are used, the URL to a zip file is provided for both the instance and schema fields. When opening these the processor needs to open the xbrl instance file if it has the attribute of *"readMeFirst"*. Any processor using the file will have to be able to open the zip files to get the XBRL contents.

Results

The results section lists the error results expected from running the test case. If there is no data in the results section this indicates a pass case and no errors are expected for the filing.

```
<variation id="DQC_0015.200.112">
    <results blockedMessageCodes = "DQC.US.0006.01 |DQC.US.0015.46 ">
```

The results header may have an attribute called *"blockedMessageCodes".* This indicates that the testcase will create errors for other rules and lists all the other rules for which it will create errors. If a processor is running the rules for a given filing, a processor can choose not to run these rules or block the results as the test case was not intended to specifically test this particular case. To avoid confusion however, the file includes a listing of these rules. These are in the format DQC.US.00XX.XX (see the syntax explanation above). Each rule result code is separated by a "|" (pipe) symbol. There is no limit to the number of results this attribute may have.

The results section has a subsection called error that is a summary of the errors for the test case.

```
<variation id="DQC_0015.200.112">
    <results>
      <error severity="error" count="1">DQC.US.0015.200</error>
```

This lists the number of errors expected using the count attribute (Shown in red). Each error is then listed using a section called result. The field for result will repeat for the number of errors that the test case generates for a specific rule.

```
<variation id="DQC_0015.200.112">
      <error severity="error" count="1">DQC.US.0015.200</error>
      <result>
        <primaryElement>us-gaap:AcceleratedShareRepurchasesFinalPricePaidPerShare</primaryElem
        ent>
        <factValue>-11.683</factValue>
        <startDate>2014-01-01</startDate>
        <endDate>2014-12-31</endDate>
        <dimensions />
        <message>${fact1.label} has a value of -11.683 which is less than zero. This element
        should not have a negative value. The properties of this
        us-gaap:AcceleratedShareRepurchasesFinalPricePaidPerShare fact are:
        Period: 2014-12-31 - 2014-12-31
        Dimensions: none
        Unit: USD/Share
        Rule version: ${ruleVersion}</message>
      </result>
```

The result section includes the details of the rule results, so it is possible to check that errors returned by a processor match expected results. An error is returned for each fact value where an error is encountered. It is common to have more than one error - often an error will occur multiple times across different periods. Errors can include the element name, the period information, the dimensions impacted and the unit. The error section also details what the error message should look like. The error message might not complete all the error description variables; it provides the error message for reference.

Custom XBRL Files

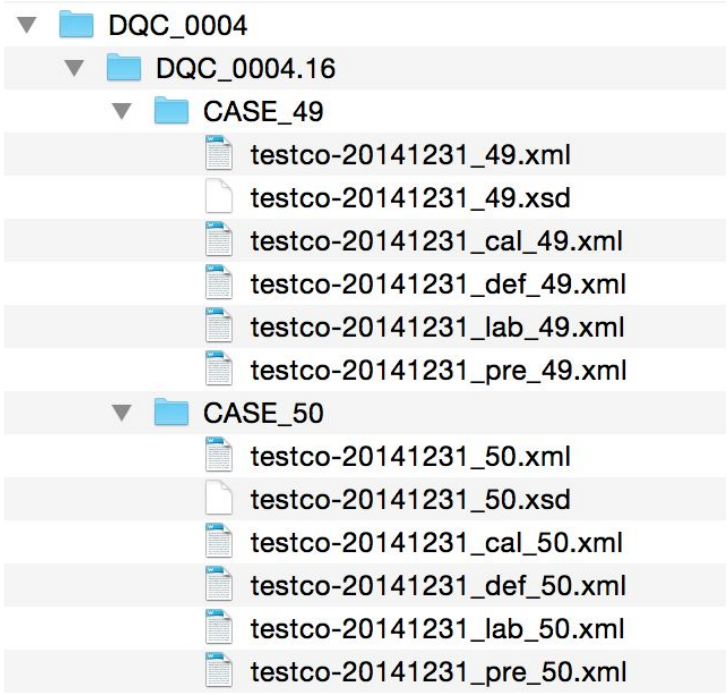
The test suite includes a large number of small custom-created XBRL files that will load and either pass or fail a specific test case quickly. Each of the test case files is referenced in the variation file. These files can either be an instance file and schema file, or a set of files that includes all the associated XBRL linkbases. These custom test files are not necessarily EFM valid but are valid XBRL documents. For this reason EFM validation rules should be disabled in software when testing DQC rules, as this will create a number of superfluous messages.

Location

The files are organized in a single zip file by rule number, rule element id and finally by test case number. This hierarchy is shown in figure 2 below. The variation file references these files using a relative location. The variation file is included in the rule number folder (ie. in Figure 2 below, the variation file is located in the directory called DQC_0004).

The test directories do not include zip files for each set of XBRL files related to a test case. Where test cases are actual SEC filings, URLs referencing the original filing on the SEC website are indicated in the variation file. These URLs could either be a reference to a zip file or an xml file. Software vendors using the test case need to be able to process a dts of files provided either the zip file or an xml file.

Figure 2. Test case Hierarchy



How are the files published?

The test cases are cumulative and all test cases for each rule are released in a single submission at the start of each public exposure period. In addition, specific test cases will be released for each rule release so that changes can be isolated. An index of all test cases will also be posted to the DQC Rules repository on GitHub.

Test cases will be published with the following naming format.

DQC_Testcases_{Release|Review}_{All|Version}_V{Version Number}.zip

So the test case file for the first major release of DQC rules will be named as follows:

- **DQC_Testcases_Release_All_V1.zip**

The file for the second major set of rules for public review will be named as follows:

- **DQC_Testcases_Review_Version_V2.zip**

Run the Test Cases using Arelle

To run the test cases using Arelle (current version):

- Copy the zip file onto a machine that has arelle installed.
- Unzip the files into the directory {PATH TO TESTCASE}/DQC_Testcases
- Ln -s the plugin (src subdirectory) to arelle's plugin/validate subdirectory/DQC
- Here is a sample bash file "runDQCTests.sh" for Mac/Linux.

```
#!/bin/bash
TESTCASESROOT={PATH TO TESTCASE}/DQC_Testcases"
OUTPUTLOGFILE={PATH TO WHERE OUTPUT SHOULD GO}/DQC-log.txt
OUTPUTERRFILE={PATH TO WHERE OUTPUT SHOULD GO}/DQC-err.txt
OUTPUTCSVFILE={PATH TO WHERE OUTPUT SHOULD GO}/DQC-report.csv
TESTCASESINDEXFILE="$TESTCASESROOT/index.xml"
ARELLEDIR={PATH TO WHERE ARELLE IS}
PYTHONPATH=$ARELLEDIR

rm $OUTPUTLOGFILE $OUTPUTERRFILE

python3.4 -m arelle.CntlrCmdLine --file "$TESTCASESINDEXFILE" --validate --plugins '{PATH TO DQC PLUGIN}|logging/dqcParameters.py' --disclosureSystem efm-pragmatic --logCodeFilter '(?!EFM)' --csvTestReport "$OUTPUTCSVFILE" --logFile "$OUTPUTLOGFILE" 2> "$OUTPUTERRFILE"
```

- Run the bash file nohup scripts/runDQCTests.sh > log/nohup.out &

- On Windows, if the binary distribution is installed, to run as above, modifying for your installation locations (test-suite-index-file for where you installed the test suite, dqc-plugin-dir for where you installed the src dir of DQC distribution, and output file locations:

"c:\program files\arelle\arelleCmdLine" --file "test-suite-index-file" --validate --plugins "dqc-plugin-dir|logging\dqcParameters.py" --disclosureSystem efm-pragmatic --logCodeFilter "(?!EFM)"

--csvTestReport "output-csv-file" --logFile "output-log-file" 2> "output-err-file"

Note that this could take several hours to run. The OUTPUTLOGFILE can be inspected to show progress (such as with the linux "tail" command). (You can shorten the run to a few minutes for testing purposes for all rules other than 0015, by temporarily commenting out rule 0015 in the test suite index file.)

Review the DQC-report.csv file to see the results of the test cases against the expected result. This file is written at completion of all of the test suite's processing.