Automated Error Analyzer for Reference Clock

# Introduction

This tool takes as input failure data ( error logs) , analyses the data to determine the root cause of failure, and the parts to be replaced.

# Prerequisites

* Set of error logs to be parsed.
* Rules defined by the user, based on which the error log will be parsed.

# Interface

errl\_parser.py [[--help] | [--ruleHelp] | [--keyList]] --elog <errlog\_path> [--ruleXml <xml\_path>]

--help: to print this message

--ruleHelp: to print the help message on how to add custom rule

--errlEducation:

--keyList: to print list of keys available in the oscc error log.

--elog: to specify the path to directory in which error logs are kept

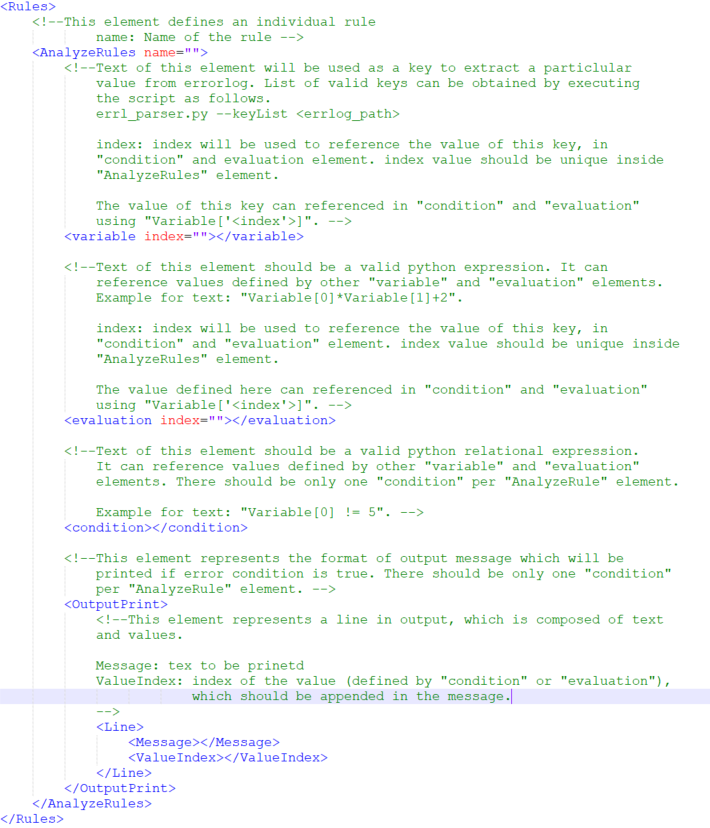
--ruleXml: to specify the path to rule XML. Default is “rule.xml”

# Rules definition

Error log parser tool checks for conditions in error logs that are specified through rule xml. The rule xml can be defined as per the specifications provided below.

# Rule XML

## Format



Rule tag details

Rules: Root node of the xml.

AnalyzeRules: This node represents an individual rule, which is used to detect error in each log. User can give any name to the rule using “name” attribute. , which will be used to print the message when the error defined in this rule is detected.

* “variable”: Variable represents a parameter in the error log that will be evaluated for certain condition, if the condition is found in the error log then appropriate error message will be displayed. User is free to use any number of variables. These parameters can be referred as “Variable[‘<index>’]” in “evaluation” and “condition” elements.

E.g.: - <variable index="0">CP Reg 1020019</variable>

* “evaluation”: Using this element user can generate intermediate values that can be used as a parameter while defining the condition. Data of this elements should be a python compatible expression, where user can use “Variable[‘<index>’]” to refer any parameter or intermediate value already defined.

E.g.: - <evaluation index="1">(Variable['0'] &amp; 0xF0) >> 4 - 1</evaluation>

* “condition”: Using this element user can define the condition which will be used to detect the error. This should be a python compatible relational expression, where user can use “Variable[‘<index>’]” to refer any parameter or intermediate value already defined. If this condition is met then only it will process remaining elements in the rule. Only 1 condition is expected in each rule.

E.g.: - <condition>(Variable['0'] == 0) or (Variable['1'] == 1)</condition>

* “OutputPrint”: This tag defines the format of the out put message. This will print only if the condition is true.
  + “Line”: this element defines a line in the output message. A line is a combination of text and values, which are defines as following elements.
    - “Message”: This elements data should be a string, which will be printed as it is, in the console.
    - “ValueIndex”: This element defines the index corresponding to the value to be printed.

E.g.: - <OutputPrint>

   <!--Print an empty line-->

   <Line/>

   <!--Print string like "Failed Node: <node\_value>"-->

   <Line>

      <Message>Failed Node: </Message>

      <ValueIndex>1</ValueIndex>

   </Line>

</OutputPrint>

Please note that the node value has to be assigned to index 1 using “variable” or “evaluation” element prior to this.

## Example

This is example of a rule to evaluate condition for clock card failover.

<AnalyzeRules name="Failover">

    <variable index="0">Current Card Group</variable>

    <variable index="1">Last Requested Card Group</variable>

    <!--"Current Card Group" not same as "Last Requested Card Group"-->

    <condition>(Variable['0'] != Variable['1'])</condition>

    <!--Since this value is not used in condition it is defined after condition-->

    <variable index="2">Slot id</variable>

    <!--Calculating <node\_value>-->

    <evaluation index="3">(Variable['2'] &amp; 0xF0) >> 4 - 1</evaluation>

    <!--Calculating <slot\_id>-->

    <evaluation index="4">(Variable['2'] &amp; 0xFFFFFFF0) | (Variable['1'] &amp; 0x0F)</evaluation>

    <OutputPrint>

        <!--Empty line-->

        <Line/>

        <!--Failed Node: <node\_value>-->

        <Line>

            <Message>Failed Node: </Message>

            <ValueIndex>3</ValueIndex>

        </Line>

        <!--Failed Card Group: <card\_group>-->

        <Line>

            <Message>Failed Card Group: </Message>

            <ValueIndex>1</ValueIndex>

        </Line>

        <!--Failed Slot id: <slot\_id>-->

        <Line>

            <Message>Failed Slot id: </Message>

            <ValueIndex>4</ValueIndex>

        </Line>

    </OutputPrint>

</AnalyzeRules>

In this example failover error rule is converted. Failover error is described below:

* Name of error: Failover
* Error exist if “Current Card Group” value and “Last Requested Card Group” are not same.
* If error exist following message should print

Failed Node: <node\_value>

Failed Card Group: <card\_group>

Failed Slot id: <slot\_id>

Where,

* + <node\_vale> can be calculated as one subtracted from 4-7 bit of “Slot id”
  + <card\_group> value will be same as “Last Requested Card Group”
  + <slot\_id> can be calculated using expression.

XML will be as follows: