**General notes on fields**

Each rule starts with <Rule> and ends with </Rule> tag.

**<Name>** tag defines a human–readable name for the rule, such as “Company name check”

**<Type>** tag defines the rule (what to do for checking). It will be described below

**<Severity>** tag is optional. It defines if the rule should report “Warning” or “Pass” instead of an “Error”. If the tag is omitted, “Error” is assumed.

**<Constant>** tag defines a mnemonic used in LAS files for constants and parameters

**<Channel>** tag defines a mnemonic used in LAS files for channel names

**<Value>** tag specifies possible value and can be grouped within <Values> </Values> pair

**<MinValueError>** tag specifies possible value and can be grouped within <Values> </Values> pair

**<MinValueWarning>** tag specifies possible value and can be grouped within <Values> </Values> pair

**<MaxValueWarning>** tag specifies possible value and can be grouped within <Values> </Values> pair

**<MaxValueError>** tag specifies possible value and can be grouped within <Values> </Values> pair

**<Comparison>** one of **LESS**, **LESS\_OR\_EQUAL**, **EQUAL**, **MORE\_OR\_EQUAL**, **MORE**, or **NOT\_EQUAL**

**<Hole>** optional tag specifies if the channel check is performed in **OPEN** hole, **CASED** hole or **BOTH**. If the tag is omitted, the entire log is checked (same as both).

**Rules description**

1. **NamingConvention** – checks if the file name follows the convention

<Rule>

<Name> </Name>

<Type>NamingConvention</Type>

<Values>

<Value> </Value>

<Value> </Value>

</Values>

<Capitals>YES</Capitals>

</Rule>

1. **ConstantSet** – checks if the constant is set to any non-empty value

<Rule>

<Name> </Name>

<Type>ConstantSet</Type>

<Constant> </Constant>

</Rule>

1. **ParameterSet** – checks if the parameter is set to any non-empty value

<Rule>

<Name> </Name>

<Type>ParameterSet</Type>

<Constant> </Constant>

</Rule>

1. **ConstantEquals** – checks if the constant is exactly equal to a particular value

<Rule>

<Name> </Name>

<Type> Constant Equals</Type>

<Constant> </Constant>

<Value> </Value>

</Rule>

1. **ParameterEquals** – checks if the parameter is exactly equal to a particular value

<Rule>

<Name> </Name>

<Type>ParameterEquals</Type>

<Constant> </Constant>

<Value> </Value>

</Rule>

1. **ConstantOneOf** – checks if the constant is one of the specified values

<Rule>

<Name> </Name>

<Type>ConstantOneOf</Type>

<Constant> </Constant>

<Values>

<Value> </Value>

<Value> </Value>

</Values>

1. **ParameterOneOf** – checks if the parameter is one of the specified values

<Rule>

<Name> </Name>

<Type>ParameterOneOf</Type>

<Constant></Constant>

<Values>

<Value> </Value>

<Value> </Value>

</Values>

1. **ConstantStartsWithOneOf** – checks if the constant starts with one of the specified substrings

<Rule>

<Name> </Name>

<Type>ConstantStartsWithOneOf</Type>

<Constant> </Constant>

<Values>

<Value> </Value>

<Value> </Value>

</Values>

1. **ParameterStartsWithOneOf** – checks if the parameter starts with one of the specified substrings

<Rule>

<Name> </Name>

<Type>ParameterStartsWithOneOf</Type>

<Constant> </Constant>

<Values>

<Value> </Value>

<Value> </Value>

</Values>

1. **ConstantContainsOneOf** – checks if the constant contains one of the specified substrings

<Rule>

<Name> </Name>

<Type>ConstantContainsOneOf</Type>

<Constant> </Constant>

<Values>

<Value> </Value>

<Value> </Value>

</Values>

1. **ParameterContainsOneOf** – checks if the parameter contains one of the specified substrings

<Rule>

<Name> </Name>

<Type>ParameterContainsOneOf</Type>

<Constant> </Constant>

<Values>

<Value> </Value>

<Value> </Value>

</Values>

1. **ConstantUnit** – checks if the constant has the correct unit (if present)

<Rule>

<Name> </Name>

<Type>ConstantUnit</Type>

<Constant> </Constant>

<Value></Value>

</Rule>

1. **ParameterUnit** – checks if the parameter has the correct unit (if present)

<Rule>

<Name> </Name>

<Type>ParameterUnit</Type>

<Constant> </Constant>

<Value></Value>

</Rule>

1. **ConstantNumericWithin** – checks if the constant is within specified values numerically

<Rule>

<Name> </Name>

<Type>ConstantNumericWithin</Type>

<Constant> </Constant>

<Values>

<MinValueError> </MinValueError>

<MinValueWarning> </MinValueWarning>

<MaxValueWarning> </MaxValueWarning>

<MaxValueError> </MaxValueError>

</Values>

</Rule>

1. **ParameterNumericWithin** – checks if the parameter is within specified values numerically

<Rule>

<Name> </Name>

<Type>ParameterNumericWithin</Type>

<Constant> </Constant>

<Values>

<MinValueError> </MinValueError>

<MinValueWarning> </MinValueWarning>

<MaxValueWarning> </MaxValueWarning>

<MaxValueError> </MaxValueError>

</Values>

</Rule>

1. **ConstantNumericWithin** – checks if the constant is within specified values numerically

<Rule>

<Name> </Name>

<Type>ParameterCompare</Type>

<Constant> </Constant>

<CompareWith> </ CompareWith >

<Comparison></Comparison>

</Rule>

1. **ParameterCompare** – checks if the parameter is in relation to any other parameter

<Rule>

<Name> </Name>

<Type>ParameterCompare</Type>

<Constant> </Constant>

<CompareWith> </ CompareWith >

<Comparison></Comparison>

</Rule>

1. **ConstantLatitudeWithin** – checks if the constant is within specified values as Latitude

<Rule>

<Name> </Name>

<Type>ConstantLatitudeWithin</Type>

<Constant> </Constant>

<Values>

<MinValueError> </MinValueError>

<MinValueWarning> </MinValueWarning>

<MaxValueWarning> </MaxValueWarning>

<MaxValueError> </MaxValueError>

</Values>

</Rule>

1. **ConstantLongitudeWithin** – checks if the parameter is within specified values as Longitude

<Rule>

<Name> </Name>

<Type>ConstantLongitudeWithin</Type>

<Constant> </Constant>

<Values>

<MinValueError> </MinValueError>

<MinValueWarning> </MinValueWarning>

<MaxValueWarning> </MaxValueWarning>

<MaxValueError> </MaxValueError>

</Values>

</Rule>

1. **ChannelPresent** – checks if the channel is present and has description

<Rule>

<Name> </Name>

<Type>ChannelPresent</Type>

<Channel> </Channel>

</Rule>

1. **ChannelUnit** – checks if the channel correct units (if present)

<Rule>

<Name> </Name>

<Type>ChannelUnitSet</Type>

<Channel> </Channel>

<Value> </Value>

</Rule>

1. **ChannelUnitOneOf** – checks if the channel correct units (if present)

<Rule>

<Name> </Name>

<Type>ChannelUnitOneOf</Type>

<Channel> </Channel>

<Values>

<Value> </Value>

<Value> </Value>

</Values>

</Rule>

1. **ChannelWithin** – checks if the channel is within specified values (if present)

<Rule>

<Name> </Name>

<Type>ChannelWithin</Type>

<Channel> </Channel>

<Hole> </Hole>

<Values>

<MinValueError> </MinValueError>

<MinValueWarning> </MinValueWarning>

<MaxValueWarning> </MaxValueWarning>

<MaxValueError> </MaxValueError>

</Values>

</Rule>

1. **ChannelWithinConstant** – checks if the channel is within specified values from a constant

<Rule>

<Name> </Name>

<Type>ChannelWithin</Type>

<Channel> </Channel>

<Constant> </Constant>

<Hole> </Hole>

<Values>

<MinValueError> </MinValueError>

<MinValueWarning> </MinValueWarning>

<MaxValueWarning> </MaxValueWarning>

<MaxValueError> </MaxValueError>

</Values>

</Rule>

1. **ChannelAverageWithin** – checks if the channel’s average is within specified values (if present)

<Rule>

<Name> </Name>

<Type>ChannelAverageWithin</Type>

<Channel> </Channel>

<Hole> </Hole>

<Values>

<MinValueError> </MinValueError>

<MinValueWarning> </MinValueWarning>

<MaxValueWarning> </MaxValueWarning>

<MaxValueError> </MaxValueError>

</Values>

</Rule>

1. **ChannelCoverage** – checks if the channel (if present) is not a missing value as of % of the hole length

<Rule>

<Name> </Name>

<Type>ChannelNotEmpty</Type>

<Channel> </Channel>

<Hole> </Hole>

<Values>

<MinValueError> </MinValueError>

<MinValueWarning> </MinValueWarning>

<MaxValueWarning> </MaxValueWarning>

<MaxValueError> </MaxValueError>

</Values>

</Rule>