Contents

Validation Types Supported By ValidateThis

ValidateThis comes bundled with a number of validation types, and you can also add your own validation types. This page describes all of the built-in validation types.

Default Failure Messages

Each validation type generates a default message when it fails, which generally includes the name of the property for which the failure occurred, as well as the values of the parameters defined for the rule. The default failure message for each rule type is documented below.

Note that by default each failure message is prepended with the word "The ", followed by the property description, for example "The Email Address is required." You can change the text that is prepended to the default validation failure message using the *defaultFailureMessagePrefix* key of the ValidateThisConfig Struct. For example, if you set defaultFailureMessagePrefix="", the example message above would be generated as "Email Address is required".

Built-In Validation Types

Boolean

The *Boolean* type ensures that the contents of a property is a valid ColdFusion boolean value.

Parameters: None

Default Failure Message: The *propertyDescription* must be a valid boolean.

Metadata Example:

```
<rule type="boolean" />
```

CollectionSize

The *CollectionSize* type ensures that the contents of a property is of a specific size. On the server this can be a list, struct or array. On the client it can be a multi-select form field.

Parameters:

- *min (optional)* - the minimum size of the collection. Defaults to *1*.
- *max (optional)* - the maximum size of the collection. Defaults to *min*.

Default Failure Message: The *propertyDescription* size is not equal or greater than *min*. Or The *propertyDescription* size is not between *min* and *max*.

Metadata Examples:

```
<rule type="collectionSize">
 <param name="min" value="3" />
</rule>
```

This will ensure that the contents of the property is a collection with at least 3 elements.

```
<rule type="collectionSize">
 <param name="min" value="1" />
 <param name="max" value="3" />
```

```
</rule>
```

This will ensure that the contents of the property is a collection with between 1 and 3 elements.

Custom

The *Custom* type allows you to create a validation which uses any arbitrary CFML code. You do so by specifying the name of a method in your object that will perform the validation.

Parameters:

- *methodName* - the name of the method in your object that determines whether the validation passes or not
- *remoteURL (optional)* - a url that can be called via AJAX which will run code to determine whether the validation passes or not, and returns a message to the client

Default Failure Message: A custom validator failed. Note that this should be overridden using either the *failureMessage* attribute of the rule or via the data returned from the method called, although the former will not return a meaningful message for client-side validations.

Metadata Example:

```
<rule type="custom">
 <param name="methodName" value="checkDuplicateNickname" />
 <param name="remoteURL" value="checkDuplicateNickname.cfm" />
</rule>
```

The method defined in your object, which would be *checkDuplicateNickname* in the above example, should return either a boolean or a structure with at least one key: *isSuccess*, which should be set to *true* if the validation passed and to *false* if the validation failed. The structure can also contain a *failureMessage* key, which would contain the message to display upon failure. The process that is reached via the *remoteURL* parameter must return a simple text string, which should which should be *true* if the validation passed and *false* if the validation failed.

More information on using the Custom validation type can be found on the Using the Custom Validation Type page.

Date

The *Date* type ensures that the contents of a property is a valid date value.

Parameters: None

Default Failure Message: The *propertyDescription* must be a valid date.

Metadata Example:

```
<rule type="date" />
```

DateRange

The *DateRange* type ensures that the contents of a property is a valid date that falls between two dates.

Parameters:

- *from* - the start date of the range
- *until* - the end date of the range

Default Failure Message: The *propertyDescription* must be a date between *from* and *until*.

Metadata Example:

```
<rule type="date">
 <param name="from" value="2010-01-01" />
 <param name="until" value="2010-12-31" />
</rule>
```

This will ensure that the property has a date between January 1, 2010 and December 31, 2010.

DoesNotContainOtherProperties

The *DoesNotContainOtherProperties* type ensures that the contents of a property does not contain values found in other properties. This can be used, for example, to ensure that a *password* property doesn't include the user's first or last names.

Parameters:

- *propertyNames* - the names of the other properties
- *delim (optional)* - the delimiter used in the list of other properties specified in the *propertyNames* parameter. Defaults to a comma.

Default Failure Message: The *propertyDescription* must not contain the values of properties named: *propertyNames*.

Metadata Example:

```
<rule type="doesNotContainOtherProperties" >
 <param name="propertyNames" value="firstName,lastName" />
</rule>
```

This will ensure that the property does not contain the values of either the *firstName* or *lastName* properties.

Email

The *Email* type ensures that the contents of a property is a valid email address.

Parameters: None

Default Failure Message: The *propertyDescription* must be a valid email address.

Metadata Example:

```
<rule type="email" />
```

Note that the test on the client-side uses the jQuery Validation plugin's built-in email validator and the test on the server-side uses ColdFusion's isValid(), so the results are not always identical. It is possible to have an email address that passes on the client and not on the server.

EqualTo

The *EqualTo* type ensures that the contents of one property is the same as the contents of another property.

Parameters:

- *comparePropertyName* - the name of another property
- *comparePropertyDesc (optional)* - the description of the other property

Default Failure Message: The *propertyDescription* must be the same as the *comparePropertyDesc*.

Metadata Example:

```
<rule type="equalTo">
 <param name="comparePropertyName" value="anotherProperty" />
</rule>
```

This will ensure that the contents of the property is the same as the contents of the *anotherProperty* property.

False

The *False* type ensures that the contents of a property is a value that can be interpreted as *false*. This includes the values *false*,*no* and *0*.

Parameters: None

Default Failure Message: The *propertyDescription* must be a false boolean.

Metadata Example:

```
<rule type="false" />
```

FutureDate

The *FutureDate* type ensures that the contents of a property is a valid date that falls after a particular date.

Parameters:

- *after (optional)* - defaults to the current date

Default Failure Message: The *propertyDescription* must be a date in the future. Note that if an *after* parameter is specified the message will be appended with: The date entered must come after *after*.

Metadata Examples:

```
<rule type="futureDate" />
```

This will ensure that the property has a date in the future.

```
<rule type="futureDate">
 <param name="after" value="2010-01-01" />
</rule>
```

This will ensure that the property has a date after January 1, 2010.

InList

The *InList* type ensures that the contents of a property is one of the values specified in a list.

Parameters:

- *list* - the list of values to be searched
- *delim (optional)* - the delimiter used in the list of values to be searched. Defaults to a comma.

Default Failure Message: The *propertyDescription* was not found in the list: *list*.

Metadata Example:

```
<rule type="inList">
 <param name="list" value="John,Paul,George" />
</rule>
```

This will ensure that the property has a value of either *John*, *Paul* or *George*.

Integer

The *Integer* type ensures that the contents of a property is a valid integer value.

Parameters: None

Default Failure Message: The *propertyDescription* must be an integer.

Metadata Example:

```
<rule type="integer" />
```

IsValidObject

The *IsValidObject* type is used to validate properties which contain other objects.

Parameters:

- *objectType (optional)* - the type of object contained in the property. This corresponds to what you'd normally pass into the *objectType* argument of the *validate* method when manually validating an object.
- *context (optional)* - the context to be used when validating the object contained in the property. Defaults to *, which is all or no contexts.

Default Failure Message: The *propertyDescription* is invalid: *messages generated while validating the object*.

Metadata Example:

```
<rule type="isValidObject">
 <param name="objectType" value="user" />
 <param name="context" value="register" />
</rule>
```

This will cause the object that is stored inside the property to be validated by ValidateThis as a *user* object and using the *register* context.

Max

The *Max* type ensures that the contents of a property contains a maximum value.

Parameters:

- *max* - a number

Default Failure Message: The *propertyDescription* must be no more than *max*.

Metadata Example:

```
<rule type="max">
 <param name="max" value="5" />
</rule>
```

This will ensure that the property has a value of *5* or less.

MaxLength

The *MaxLength* type ensures that the length of the contents of a property is no more than a certain number of characters.

Parameters:

- *maxLength* - a number

Default Failure Message: The *propertyDescription* must be no more than *maxLength* characters long.

Metadata Example:

```
<rule type="maxLength">
 <param name="maxLength" value="5" />
</rule>
```

This will ensure that the contents of the property is no more than *5* characters long.

Min

The *Min* type ensures that the contents of a property contains a minimum value.

Parameters:

- *min* - a number

Default Failure Message: The *propertyDescription* must be at least *min*.

Metadata Example:

```
<rule type="min">
 <param name="min" value="5" />
</rule>
```

This will ensure that the property has a value of *5* or more.

MinLength

The *MinLength* type ensures that the length of the contents of a property is at least a certain number of characters.

Parameters:

- *minLength* - a number

Default Failure Message: The *propertyDescription* must be at least *minLength* characters long.

Metadata Example:

```
<rule type="minLength">
 <param name="minLength" value="5" />
</rule>
```

This will ensure that the contents of the property is at least *5* characters long.

MinPatternsMatch

The *MinPatternsMatch* type ensures that the contents of a property matches a minimum number of patterns. This can be used, for example, to ensure that a password property conforms to certain standards.

Parameters:

- *minMatches* - the minimum number of patterns that the value must match
- *pattern_x* - where *x* is the name of a pattern and the value of the parameter is a regex to be evaluated

Default Failure Message: *numberOfPatternsMatched* patterns were matched but *minMatches* were required.

Metadata Example:

```
<rule type="MinPatternsMatch">
 <param name="minMatches" value="3" />
 <param name="pattern_lowerCaseLetter" value="[a-z]" />
 <param name="pattern_upperCaseLetter" value="[A-Z]" />
 <param name="pattern_digit" value="[\d]" />
 <param name="pattern_punct" value="[[:punct:]]" />
</rule>
```

This will ensure that the contents of the property conforms to at least *3* of the 4 regex patterns listed.

NoHTML

The *NoHTML* type ensures that the contents of a property does not contain HTML.

Parameters: None

Default Failure Message: The *propertyDescription* cannot contain HTML tags.

Metadata Example:

```
<rule type="noHTML" />
```

NotInList

The *NotInList* type ensures that the contents of a property is not one of the values specified in a list.

Parameters:

- *list* - the list of values to be searched
- *delim (optional)* - the delimiter used in the list of values to be searched. Defaults to a comma.

Default Failure Message: The *propertyDescription* was found in the list: *list*.

Metadata Example:

```
<rule type="notInList">
 <param name="list" value="John,Paul,George" />
</rule>
```

This will ensure that the property does not have a value of either *John*, *Paul* or *George*.

Numeric

The *Numeric* type ensures that the contents of a property is a valid numeric value.

Parameters: None

Default Failure Message: The *propertyDescription* must be a number.

Metadata Example:

```
<rule type="numeric" />
```

PastDate

The *PastDate* type ensures that the contents of a property is a valid date that falls before a particular date.

Parameters:

- *before (optional)* - defaults to the current date

Default Failure Message: The *propertyDescription* must be a date in the past. Note that if an *before* parameter is specified the message will be appended with: The date entered must come before *before*.

Metadata Examples:

```
<rule type="pastDate" />
```

This will ensure that the property has a date in the past.

```
<rule type="pastDate">
 <param name="before" value="2010-12-31" />
</rule>
```

This will ensure that the property has a date before December 31, 2010.

Range

The *Range* type ensures that the contents of a property contains a value between two numbers.

Parameters:

- *min* - a number
- *max* - a number

Default Failure Message: The *propertyDescription* must be between *min* and *max*.

Metadata Example:

```
<rule type="range">
 <param name="min" value="5" />
 <param name="max" value="10" />
</rule>
```

This will ensure that the property has a value between *5* and *10*.

RangeLength

The *RangeLength* type ensures that the length of the contents of a property is between two numbers.

Parameters:

- *minLength* - a number
- *maxLength* - a number

Default Failure Message: The *propertyDescription* must be between *minLength* and *maxLength* characters long.

Metadata Example:

```
<rule type="rangeLength">
 <param name="minLength" value="5" />
 <param name="maxLength" value="10" />
</rule>
```

This will ensure that the contents of the property is between *5* and *10* characters long.

Regex

The *Regex* type ensures that the contents of a property conforms to a regular expression.

Parameters:

- *regex* - a regular expression
- *serverRegex (optional)* - a regular expression

Default Failure Message: The *propertyDescription* must match the specified pattern.

Metadata Example:

```
<rule type="regex">
 <param name="regex" value="^(Dr|Prof|Mr|Mrs|Ms|Miss)(\.)?$"  />
</rule>
```

This will ensure that the contents of the property matches the expression. Note that you can provide a CFML-only regex using the *serverRegex* parameter, otherwise the same regex will be used on the client and the server.

Required

The *Required* type ensures that the contents of a property is not empty.

Parameters: None

Default Failure Message: The *propertyDescription* is required.

Metadata Example:

```
<rule type="required" />
```

Time

The *Time* type ensures that the contents of a property is a valid time value in the format 00:00. This includes values between 00:00 and 23:59.

Parameters: None

Default Failure Message: The *propertyDescription* must be a must be a valid time, between 00:00 and 23:59.

Metadata Example:

```
<rule type="time" />
```

True

The *True* type ensures that the contents of a property is a value that can be interpreted as *true*. This includes the values *true*,*yes* and any number other than *0*.

Parameters: None

Default Failure Message: The *propertyDescription* must be a true boolean.

Metadata Example:

```
<rule type="true" />
```

URL

The *URL* type ensures that the contents of a property is a valid (properly formed) URL.

Parameters: None

Default Failure Message: The *propertyDescription* must be a valid URL.

Metadata Example:

```
<rule type="url" />
```

Server Only Validation Types

Expression

The *Expression* type ensures that a cfml expression evaluates to true.

Parameters:

- *expression* - the cfml expression to evaluate on the object.

Default Failure Message: [NONE]

Metadata Example:

```
<rule type="expression" failureMessage="Foo is not equal to Bar" >
 <param name="expression" value="getFoo() eq getBar()" />
```

```
</rule>
```