|  |  |
| --- | --- |
| January 19, 2011 |  |
|  | Security Framework |
| Version 1.0 |  |
| Validators user’s guide | |



Table of Contents

1 Introduction 1

1.1 SubsystemProperties 1

1.2 Engine 1

1.2.1 Non-configurable engines 2

1.2.2 Basic Int Validator 2

1.2.3 Basic Longint Validator 3

1.2.4 Basic Double Validator 4

1.2.5 Basic String Validator 5

1.2.6 Printable String Validator 6

1.2.7 Pattern String Validator 7

1.2.8 BASE 64 Validator 8

2 Validator subsystem API 9

2.1 Simplified API 9

2.2 General API 11

2.3 API Examples 12

2.3.1 Simplified API 12

2.3.2 General API 13

# Introduction

Validator subsystem provides Configuration files structure

## SubsystemProperties

Root element. It’s not required but recommended do not change the element’s name. It matches with name of the class that takes the configuration properties when it is loading.

There are the following child elements and attributes under the root.

Attributes:

| Attribute | Description |
| --- | --- |
| IsRuntimeApiEnabled | Required attribute.  Indicates whether the Security Monitor API is enabled. Prevents the API from initializing if set to **false**. |
| IsRuntimeApiPublic | Required attribute.  Indicates whether the Security Monitor API is exposed to public access. |
| IsControlApiEnabled | Required attribute.  Indicates whether the Security Monitor control API is enabled. Prevents the control API from initializing if set to **false**. *For future use*. |
| IsControlApiPublic | Required attribute.  Indicates whether the Security Monitor control API is exposed to public access. *For future use*. |

Elements:

| Element | Description |
| --- | --- |
| EngineProps | Required element.  Defines available engines and their properties. |

The element’s structure is described below.

## Engine

The Engine element’s structure depends on the engine, defined by the element. There are configurable and non-configurable engines.

### Non-configurable engines

The following engines don’t have any especial configuration: Credit card number validator, HEX string validator, date string validator.

Credit Card number validator checks whether an input looks like a valid credit card number. It allows any number of spaces and hyphen characters in the number but checks an input by Luhn algorithm and remove all non-numeric characters from output.

HEX string validator checks whether an input contains hexadecimal digits only.

Date string validator checks whether an input represents a date or date and time in some of the formats of the current culture (in the thread).

All non-configurable engines use the same configuration structure.

<SubsystemProperties>

<Engines>

|  |
| --- |
| <Engine Id="<Engine\_Id>" IsDefault="true|false" RealType="<Engine\_type\_name>" /> |

| Attribute | Description |
| --- | --- |
| Id | Required attribute.  A unique ID of the engine. The engine can be gotten by this value via Validator API. |
| RealType | Required attribute.  Defines a class taking the engine’s configuration.  Value: the following engines are defined in the subsystem   * MetraTech.SecurityFramework.Core.Validator.CcnStringValidatorEngine * MetraTech.SecurityFramework.Core.Validator.HexStringValidatorEngine * MetraTech.SecurityFramework.Core.Validator.DateStringValidatorEngine |
| IsDefault | Optional attribute.  Specifies whether the engine is default for its category.  Value: Boolean (default False). |

### Basic Int Validator

Basic Int validator checks whether input string represents an integer number specified in decimal, hexadecimal or octal formats and converts an input to a number when it is. Its configuration is described below.

<SubsystemProperties>

<Engines>

|  |
| --- |
| <Engine Id="<Engine\_Id>" IsDefault="true|false" RealType="MetraTech.SecurityFramework.Core.Validator.BasicIntValidatorEngine" Min="<Integer\_number>" Max="<Integer\_number>" DoFormatAutoDetect="true|false" ExpectHexFormat="true|false" /> |

| Attribute | Description |
| --- | --- |
| Id | Required attribute.  A unique ID of the engine. The engine can be gotten by this value via Validator API. |
| RealType | Required attribute.  Defines a class taking the engine’s configuration.  Value: must be “MetraTech.SecurityFramework.Core.Validator.BasicIntValidatorEngine”. |
| IsDefault | Optional attribute.  Specifies whether the engine is default for its category.  Value: Boolean (default False). |
| Min | Optional attribute.  Specifies a minimal numeric value to pass the validation.  Value: integer number (default Int32.MinValue) |
| Max | Optional attribute.  Specifies a maximal numeric value to pass the validation.  Value: integer number (default Int32.MaxValue) |
| DoFormatAutoDetect | Optional attribute.  Indicates to the validator try to detect the input format automatically. If set to true the validator looks whether an input value starts with 0x (for hexadecimal) or with \0 (for octal) character.  Value: Boolean (default True) |
| ExpectHexFormat | Optional attribute.  Indicates to the validator always to treat an input as hexadecimal number.  Value: Boolean (default False) |

### Basic Longint Validator

Basic longint validator checks whether input string represents a long (8 bytes) integer number specified in decimal, hexadecimal or octal formats and converts an input to a number when it is. Its configuration is described below.

It is very like Basic Int Validator described above with the exception it processes 8 bytes integer numbers instead of 4 bytes. Also Min and Max attributes take long integers too. Also it has a different **RealType** – “MetraTech.SecurityFramework.Core.Validator.BasicLongValidatorEngine”.

<SubsystemProperties>

<Engines>

|  |
| --- |
| <Engine Id="<Engine\_Id>" IsDefault="true|false" RealType="MetraTech.SecurityFramework.Core.Validator.BasicLongValidatorEngine" Min="<Long\_Integer\_number>" Max="<Long\_Integer\_number>" DoFormatAutoDetect="true|false" ExpectHexFormat="true|false" /> |

### Basic Double Validator

Basic double validator checks whether an input represents a double precision number and converts an input to a number when it is. Its configuration is described below.

<SubsystemProperties>

<Engines>

|  |
| --- |
| <Engine Id="BasicDouble.Limited" IsDefault="false" RealType="MetraTech.SecurityFramework.Core.Validator.BasicDoubleValidatorEngine" Min="-2000" Max="2000" /> |

| Attribute | Description |
| --- | --- |
| Id | Required attribute.  A unique ID of the engine. The engine can be gotten by this value via Validator API. |
| RealType | Required attribute.  Defines a class taking the engine’s configuration.  Value: must be “MetraTech.SecurityFramework.Core.Validator.BasicDoubleValidatorEngine”. |
| IsDefault | Optional attribute.  Specifies whether the engine is default for its category.  Value: Boolean (default False). |
| Min | Optional attribute.  Specifies a minimal numeric value to pass the validation.  Value: Double (default Double.MinValue) |
| Max | Optional attribute.  Specifies a maximal numeric value to pass the validation.  Value: Double (default Double.MaxValue) |

### Basic String Validator

Basic string validator provides simple validation for textual values. It allows checking text length to fit within the specified range, checking an input against black and while lists of characters. Its configuration is described below.

<SubsystemProperties>

<Engines>

|  |
| --- |
| <Engine Id="<Engine\_Id>" IsDefault="true|false" RealType="MetraTech.SecurityFramework.Core.Validator.BasicStringValidatorEngine" MinLength="<Non-negative\_integer\_number>" MaxLength="<Non-negative\_integer\_number>" CharSet="<Characters\_without\_delimiter>" BlackList="true|false" WhiteList="true|false" TrimInput="true|false" /> |

| Attribute | Description |
| --- | --- |
| Id | Required attribute.  A unique ID of the engine. The engine can be gotten by this value via Validator API. |
| RealType | Required attribute.  Defines a class taking the engine’s configuration.  Value: must be “MetraTech.SecurityFramework.Core.Validator.BasicStringValidatorEngine”. |
| IsDefault | Optional attribute.  Specifies whether the engine is default for its category.  Value: Boolean (default False). |
| MinLength | Optional attribute.  Specifies minimal input length to pass the validation.  Value: Unsigned Integer number (default 0) |
| MaxLength | Optional attribute.  Specifies maximal input length to pass the validation.  Value: Unsigned Integer number (default UInt32.MaxValue) |
| CharSet | Optional attribute.  Specifies characters to be used for white or black list.  Value: string from characters without any delimiter (default empty) |
| BlackList | Optional attribute.  Indicates whether the characters specified by the CharSet are treated as black list.  Value: Boolean (default False) |
| WhiteList | Optional attribute.  Indicates whether the characters specified by the CharSet are treated as white list. It is mutually exclusive to BlackList attribure. BlackList has a higher priority.  Value: Boolean (default True) |
| TrimInput | Optional attribute.  Specifies if an input value is trimmed before processing. When this attribute is set to true an input value is trimmed before returning from the validator.  Value: Boolean (default false) |

### Printable String Validator

Printable string validator checks whether an input string consists of printable ASCII characters only. Its configuration is described below.

<SubsystemProperties>

<Engines>

|  |
| --- |
| <Engine Id="<Engine\_Id>" IsDefault="true|false" RealType="MetraTech.SecurityFramework.Core.Validator.PrintableStringValidatorEngine" TrimInput="true|false" /> |

| Attribute | Description |
| --- | --- |
| Id | Required attribute.  A unique ID of the engine. The engine can be gotten by this value via Validator API. |
| RealType | Required attribute.  Defines a class taking the engine’s configuration.  Value: must be “MetraTech.SecurityFramework.Core.Validator.PrintableStringValidatorEngine”. |
| IsDefault | Optional attribute.  Specifies whether the engine is default for its category.  Value: Boolean (default False). |
| TrimInput | Optional attribute.  Specifies if an input value is trimmed before processing. When this attribute is set to true an input value is trimmed before returning from the validator.  Value: Boolean (default false) |

### Pattern String Validator

Pattern string validator checks an input value against specified regular expressions. Regular expressions can represent as black as white lists. If the expression is for the black list, input value does not pass the validation if it matches the expression. On the other hand, if the expression is for a white list, an input string must match with at least one of expressions from the list to pass the validation.

Validator’s configuration is described below.

<SubsystemProperties>

<Engines>

|  |
| --- |
| <Engine Id="<Engine\_Id>" IsDefault="true|false" RealType="MetraTech.SecurityFramework.Core.Validator.PatternStringValidatorEngine" BlackList="true|false" WhiteList="true|false">  <PatternParams>  <item Exclude="true|false" Pattern="<Regular\_expression>"/>  <item Exclude="true|false" Pattern="<Regular\_expression>"/>  …  </PatternParams>  </Engine> |

Attributes:

| Attribute | Description |
| --- | --- |
| Id | Required attribute.  A unique ID of the engine. The engine can be gotten by this value via Validator API. |
| RealType | Required attribute.  Defines a class taking the engine’s configuration.  Value: must be “MetraTech.SecurityFramework.Core.Validator.PrintableStringValidatorEngine”. |
| IsDefault | Optional attribute.  Specifies whether the engine is default for its category.  Value: Boolean (default False). |
| BlackList | Optional attribute.  Indicates whether the expressions specified within PatternParams element are treated as black list.  Value: Boolean (default False) |
| WhiteList | Optional attribute.  Indicates whether the characters specified within PatternParams element are treated as white list. It is mutually exclusive to BlackList attribure. BlackList has a higher priority.  Value: Boolean (default True) |

Elements:

| Element | Description |
| --- | --- |
| PatternParams | Required element.  Specifies a list of regular expressions to validate an input against. Contains a list of “item” child elements. Each “item” element has two attributes:   * Pattern – specifies a regular expression itself. Value must a valid regular expression * Exclude – specifies a value indicating the pattern’s behavior is inverted. I.e., when the list represents black list and Exclude set to True, the pattern represents “white” list expression. And vice versa, setting the Exclude attribute to true together with WhiteList = True indicates that the pattern is for black list. |

### BASE 64 Validator

BASE 64 validator checks whether an input contains only characters allowed in the BASE 64 encoding (.NET default). It can optionally decode the input data.

Validator’s configuration is described below.

<SubsystemProperties>

<Engines>

|  |
| --- |
| <Engine Id="<Engine\_Id>" IsDefault="true|false" RealType="MetraTech.SecurityFramework.Core.Validator.Base64StringValidatorEngine" DoDecode="true|false" /> |

| Attribute | Description |
| --- | --- |
| Id | Required attribute.  A unique ID of the engine. The engine can be gotten by this value via Validator API. |
| RealType | Required attribute.  Defines a class taking the engine’s configuration.  Value: must be “MetraTech.SecurityFramework.Core.Validator.Base64StringValidatorEngine”. |
| IsDefault | Optional attribute.  Specifies whether the engine is default for its category.  Value: Boolean (default False). |
| DoDecode | Optional attribute.  Specifies a value indicating a valid input will be decoded from BASE 64 format.  Value: Boolean (default false) |

# Validator subsystem API

## Simplified API

Simplified Validator subsystem API is provided by MetraTech.SecurityFramework.ValidatorExtensions class and consists of extension methods for the System.String class.

To use the simplified API the reference to SecurityFramework.dll must be added and MetraTech.SecurityFramework must be declared in using directive.

The following methods are defined in the API:

| Method | Description |
| --- | --- |
| ValidateWithEngine(engineId: string): object | Invokes a validator engine with the specified ID passing the string instance as an input.  Returns a validation result (as an object).  Throws the SubsystemInputParamException when an engine with the specified ID not found.  Throws the ValidatorInputParamException or NullInputdataException when the validation failed. |
| ValidateAsBasicInt(): int | Validates the string instance using a default basic int validator.  Returns a converted value.  Throws the ValidatorInputParamException when the string does not reprerent an integer value.  Throws the NullInputdataException when the string is empty. |
| ValidateAsBasicLong(): long | Validates the string instance using a default basic longint validator.  Returns a converted value.  Throws the ValidatorInputParamException when the string does not reprerent a long integer value.  Throws the NullInputdataException when the string is empty. |
| ValidateAsBasicDouble(): double | Validates the string instance using a default basic double validator.  Returns a converted value.  Throws the ValidatorInputParamException when the string does not reprerent a double precision value.  Throws the NullInputdataException when the string is empty. |
| ValidateAsBasicString(): string | Validates the string instance using a default basic string validator.  Returns an input value itself.  Throws the ValidatorInputParamException when the string does not pass the validation. |
| ValidateAsCreditCardNumber(): string | Validates the string instance using a default credit card number validator.  Returns a sanitized credit card number value.  Throws the ValidatorInputParamException when the string does not reprerent a double precision value.  Throws the NullInputdataException when the string is empty. |
| ValidateAsHexString(): string | Validates the string instance using a default HEX string validator.  Returns an input value itself.  Throws the ValidatorInputParamException when the string does not reprerent a double precision value.  Throws the NullInputdataException when the string is empty. |
| ValidateAsPrintableString(): string | Validates the string instance using a default printable string validator.  Returns an input value itself.  Throws the ValidatorInputParamException when the string does not reprerent a double precision value.  Throws the NullInputdataException when the string is empty. |
| ValidateAsDateString(): DateTime | Validates the string instance using a default date validator.  Returns a converted value.  Throws the ValidatorInputParamException when the string does not reprerent a double precision value.  Throws the NullInputdataException when the string is empty. |
| ValidateAsBase64String(): string | Validates the string instance using a default BASE 64 validator.  Returns a converted from BASE 64 value assuming an initial data represents a UTF-8 encoded string.  Throws the ValidatorInputParamException when the string does not reprerent a double precision value.  Throws the NullInputdataException when the string is empty. |

## General API

In some cases a simplified API cannot be invoked due to some limitations or is inconvenient for some reason. There is a general API to use the Validator subsystem in such cases.

Validator subsystem is accessible through the Validator property of the MetraTech.SecurityFramework.SecurityKernel class.

The following methods are defined in the API:

| Method | Description |
| --- | --- |
| Execute(engineId: string, input: ApiInput): ApiOutput | Invokes a decoder engine with the specified ID.  Returns a validation result.  Throws the SubsystemInputParamException when an engine with the specified ID not found. |
| ExecuteDefaultByCategory(categoryName: string, input: ApiInput): ApiOutput | Invokes a default decoder of the specified category.  Returns a decoding result.  Throws the SubsystemInputParamException when a category with the specified name not found or a default engine was not specified for the category. |
| GetEngine(engineId: string): IEngine | Retrieves an engine the specified ID.  Throws the SubsystemInputParamException when an engine with the specified ID not found. |
| GetDefaultEngine(categoryName: string): IEngine | Retrieves a default engine for a category with the specified name.  Returns null when no default engine specified for the category.  Throws the SubsystemInputParamException when a category with the specified name not found. |
| GetEnginesForCategory(categoryName: string): IEngine[] | Retrieves all engines registered for the category with the specified name. |

## API Examples

### Simplified API

using MetraTech.SecurityFramework;

…

string str = "201";

int actual = ValidatorExtensions.ValidateAsBasicInt(str); // returns 201

…

string input = "201";

long actual = input.ValidateAsBasicLong(); // returns 201

…

string input = "2.0001e-1";

double actual = input.ValidateAsBasicDouble(); // returns 0.20001

…

string input = "abcdef";

string actual = ValidatorExtensions.ValidateAsBasicString(input); // returns "abcdef"

…

string input = "4111-1111-1111-1111";

string actual = input.ValidateAsCreditCardNumber(); // returns "4111111111111111"

…

string input = "01234567890abcdefFEDCBA";

string actual = input.ValidateAsHexString(); // returns "01234567890abcdefFEDCBA"

…

string input = "abcdefgh";

string actual = input.ValidateAsPrintableString(); // returns "abcdefgh"

…

string input = "January 01, 2010 13:30:25";

DateTime actual = input.ValidateAsDateString(); // returns January 01, 2010 13:30:25

…

string input = "all+";

string actual = input.ValidateAsBase64String(); // returns "jY~"

### General API

using MetraTech.SecurityFramework;

…

string input = "0123456789";

string actual = SecurityKernel.Validator.Api.Execute("PatternString.Test", input); // returns "0123456789"