Fuzzino

Fuzzino provides several interfaces. This document describes the XML based interface and shows how Fuzzino is used in order to retrieve fuzzed values from it.

General Usage

In order to retrieve fuzzed values from Fuzzino, a request in the form of an XML document (herein after referred to as “request document”) has to be created. A request document may contain several type specific requests, namely string requests and number requests. Each of these type specific requests has attributes to specify the data format of **valid values**.

Additionally to this data form specification, generators and operators (in combination with concrete valid values) can be specified that shall be used by Fuzzino for creating fuzzed values. If no generators or operators are specified, Fuzzino will use all generators and, if valid values are contained in a type specific request, all operators that match the data format description. To avoid this behaviour, in the case of generators the tag `noGenerators` can be included in the request while no operators are used if no valid values are contained in a request.

Because most of the fuzzing generators and operators create a huge number of valid values, the number of fuzzed values has to be specified for each type specific request.

The usage of Fuzzino is illustrated in Figure 1 for the case of a request of the type string. The first message contains the above mentioned information.

Every request is answered by Fuzzino with a response containing the fuzzed values, grouped by the generators and by operators in combination with the corresponding valid value that created them. This is depicted by message 2 in Figure 1.

Because of the limited number of fuzzed values returned by Fuzzino, it is often necessary to retrieve further values from Fuzzino. For that purpose, each response has two additional attributes, `moreValues` that indicates if further values can be retrieved, and `id` that has to be used to retrieve further values from Fuzzino that differs from the already retrieved values (see message 2 in Figure 1). To do so, type specific request has to be complemented with this `id` obtained from the response of Fuzzino. The data format description as well as the generators, operators and valid values can be omitted because they are already known from the initial request (see messages 3 and 4 in Figure 1).

If the desired number of fuzzed values was retrieved from Fuzzino, a request should be closed by sending the corresponding tag `closeRequest` with the `id` from Fuzzino. When receiving such a tag, Fuzzino removes temporary files regarding the request to be closed. After closing a request, it is impossible to retrieve further values for this request (see messages 5 and 6 in Figure 1).

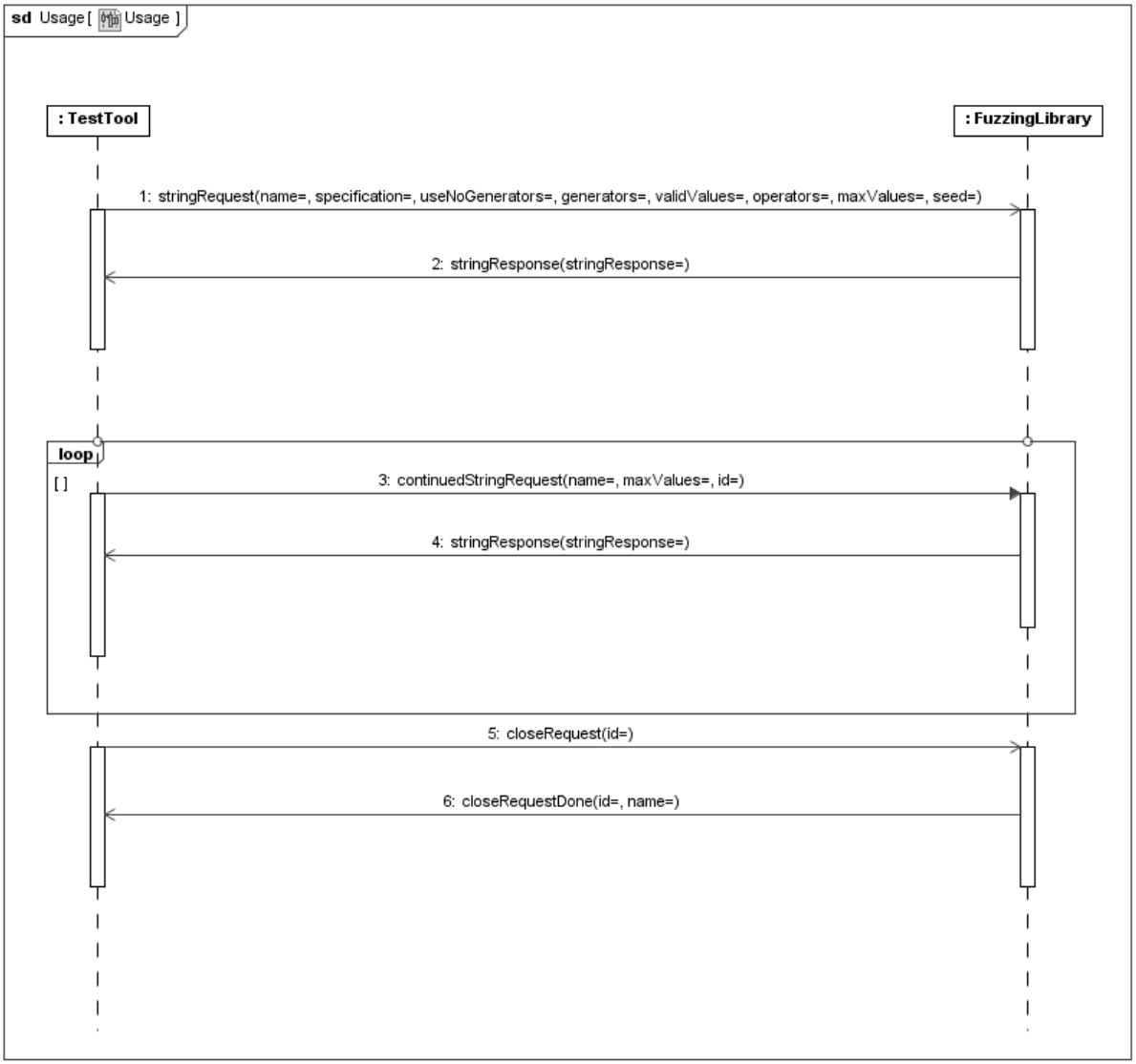

Figure 1: Usage of Fuzzino by Sending Requests and Receiving Responses

Requests to Fuzzino

request:request

This is the root element of the xml file. It contains all requests.

Parent Elements

none

Child Elements

request:number (0..*), request:string (0..*), request:closeRequest (0..*)

Content

none

Attributes

XML standard

Example

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns2:request xmlns:ns2="http://fuzzino.fuzzing.fokus.fraunhofer.de/request">
</ns:request>
```

request:string

This element opens a request for a string to be fuzzed by Fuzzino. This type determines the fuzzing generators and operators that shall be used for generating fuzz testing values.

Parent Elements

request:request

Child Elements

request:noGenerators (0..1), request:specification (0..1), request:generator (0..*), request:validValues (0..1)

Content

none

Attributes

Name	Type	Values	Description
name	xs:ID	*user-defined, unique*	Identifier for the user of Fuzzino, will be mentioned in the response. The name must be unique in the whole request.
maxValues	xs:int		Maximum number of fuzzed values requested.
id	xs:NCName	*a valid id given in a response from Fuzzino*	Required for continued requests. Identifier for Fuzzino when requesting more values. Can only be set if an initial request has been taken place. If set, all child elements of this request are ignored.
seed	xs:long		Optional. A seed for getting the same results if random-based generators or operators are involved.

Example

The following example shows a valid initial request for a string to be fuzzed. The child elements are optional and omitted for readability.

```
<string name="uniqueName" maxValues="50">
    ...
</string>
```

A request for further values can be taken as follows. The value of the attribute *id* must be taken from the response to the initial request.

```
<string name="uniqueName" id="IdFromResponse" maxValues="50" />
```

request:number

This element opens a request for a number to be fuzzed by Fuzzino. This type determines the fuzzing generators and operators that shall be used for generating fuzz testing values.

Parent Elements

request:request

Child Elements

request:noGenerators (0..1), request:specification (0..1), request:generator (0..*), request:validValues (0..1)

Content

none

Attributes

Name	Type	Values	Description
name	xs:ID	*user-defined, unique*	Identifier for the user of Fuzzino will be mentioned in the response.
maxValues	xs:int		Maximum number of fuzzed values requested.
id	xs:NCName	*a valid id given in a response from Fuzzino*	Required for continued requests. Identifier for Fuzzino when requesting more values. Can only be set if an initial request has been taken place. If set, all child elements of this request are ignored.
seed	xs:long		Optional. A seed for getting the same results if random-based generators or operators are involved.

Example

The following example shows a valid initial request for a number to be fuzzed. The child elements are omitted for readability.

```
<number name="uniqueName" maxValues="50">
    ...
</number>
```

A request for further values can be taken as follows. The value of the attribute *id* must be taken from the response to the initial request.

```
<number name="uniqueName" id="IdFromResponse" maxValues="50" />
```

request:specification

This element is an additional specification for a request. It gives hints to Fuzzino that reduces the set of fuzz testing values in a reasonable way by omitting values that are not appropriate to this specification. Fuzzino uses default values for the request, if you do not provide a specification.

Parent Elements

request:number, request:string

Child Elements

none

Content

none

Attributes

For the parent element *request:string* the following attributes are possible:

Name	Type	Values	Description
Type	xs:NCName	String *(default)* SQL Path Filename Hostname Delimiter RegExValid RegExInvalid Number Command Date Time IPAddress PIN4Digit	Optional. By setting this attribute the format of the string can be constrained. If this attribute is not set, it has the default value *String*. *SQL* specifies that the string is a parameter for a SQL query. *Path* and *Filename* refer to descriptions of objects of a file system. *Hostname* refers to a hostname that are part of a URL. *Delimiter* refers to string that can act as a delimiter in different situations, e.g. "." for an IP address. *RegExValid* is set to describe valid values with a regular expression in the content of this *request:specification* element (see attribute regEx for more information). *RegExInvalid* is set to describe invalid values with a regular expression in the content of this *request:specification* element (see attribute regEx for more information). Fuzzino responds to this request by creating all values from the regular expression. *Number* specifies a string that contains only numbers. *Command* specifies a string that is conveyed to the command line. *Date* and *Time* refers to strings for date and time specification. *IPAddress* refers to addresses according

Name	Type	Values	Description
			to IPv4. *PIN4Digit* refers to a string consisting of 4 digits.
minLength maxLength	xs:int		Optional. Describes the minimal respectively the maximal length of a valid string.
nullTerminated	xs:boolean	false *(default)* true	Optional. Specifies whether a string is terminated by a null character, for instance in C strings. If this attribute is not set, it has the default value *false*.
encoding	xs:NCName	ASCII *(default)* UTF8 UTF16 UTF32	Optional. Specifies the encoding that is used by the SUT and hence determines the available character set. It does not describe the encoding that is used for valid values within this *request:string* or that shall be used by Fuzzino for the response. If this attribute is not set, it has the default value *ASCII*.
ignoreLengths	xs:boolean	false (default), true	Optional. The attributes minLength and maxLength will be ignored if set to true.
regEx	xs:string		Optional. If the parent element is *request:string* and the value of the attribute *type* is either *RegExValid* or *RegExInvalid* this attribute is a regular expression (currently under the constraint that the regular expression may not allow a variable length).

For the parent element *request:number* the following attributes are possible:

Name	Type	Values	Description
type	xs:NCName	integer	Optional. Specifies the number as an integer (will be extended soon with float).
minValue maxValue	xs:int		Optional. Specifies a smallest or biggest valid number.
bits	xs:int	8 16 32 *(default)* 64 128	Optional. Specifies the number of bits that is used for the representation of the number. If this attribute is not set, it has the default value *32*.
signed	xs:boolean	false true *(default)*	Optional. Specifies whether the number is signed.

	If this attribute is not set, is has the default value *true*.

Example

The following example shows a valid specification within *request:string* where the attributes describe a string that acts as parameter for a SQL query with a minimal length of 1 character and a maximal length of 5 characters. The string is encoding using UTF-8. Because the attribute *nullTerminated* is not set, it is *false* indicating that the string does not end with a null character.

```
<specification type="SQL"
               minLength="1"
               maxLength="5"
               encoding="UTF8" />
```

A valid specification for *request:number* could be the following:

```
<specification type="integer"
               minValue="5"
               maxValue="15"
               bits="32"
               signed="true" />
```

This specification element determines an integer with a minimal value of 5, a maximal value of 15 that is represented by 32 bits and signed.

request:generator

This element requests a specific fuzzing generator to be used by Fuzzino. It is optional but can appear as often as required within its parent element. If at least one generator element is present, only the denoted generators will be used by Fuzzino. Otherwise all generators that match the specification of the request will be used to generate fuzz testing values.

Parent Elements

request:number, request:string

Child Elements

None

Content

none

Attributes

Name	Type	Values	Description
value	xs:string	LongStrings, BadStrings, …	The value must be a valid generator name that must match the type of the request it is enclosed in. It is not case sensitive. For a description of all generators see List of Generators.
parameter	xs:NCName	*depends on type of generator*	Optional. Some generators may have a parameter that can be used in order to specify how the generator should work. If no parameter is set for a generator that may have one, its default value for the parameter is used.

Example

The following example requests two generators, *LongStrings* and *BadStrings* that shall be used by Fuzzino.

```
<generator value="LongStrings"/>
<generator value="BadStrings"/>
```

request:noGenerators

This element specifies that no generators shall be used. It can be used to specify that only operators shall be used avoiding that all generators will be used by Fuzzino if no generator was specified within the request.

Parent Elements

request:number, request:string

Child Elements

none

Content

The content is a Boolean, so either “false” or “true”. The default value is “false”.

Attributes

none

Example

```
<noGenerators>true<noGenerators/>
```

request:validValues

This element indicates that valid values are available for Fuzzino. If this element is present, at least one valid value must be given as a child. Additionally, the operators to be used can be denoted by an arbitrary number of operator elements.

Parent Elements

request:number, request:string

Child Elements

request:value (1..*), request:operator (0..*)

Content

none

Attributes

none

Example

The following example shows a *validValues* element with the minimal required child elements:

```
<validValues>
    <value>ABC</value>
</validValues>
```

request:value

This element contains a valid value of a request.

Parent Elements

request:validValues

Child Elements

none

Content

The content is the value.

Attributes

none

Example

The following example shows a *validValues* element with the minimal required child elements:

```
<validValues>
    <value>ABC</value>
</validValues>
```

request:operator

This element requests that a specific fuzzing operator shall be applied to the valid values given in the request. It is optional but can appear as often a required within its parent element. If at least one operator element is present, only the denoted operators will be used by Fuzzino. Otherwise all operators that match the specification of the request will be used to generate fuzz testing values.

Parent Elements

request:validValues

Child Elements

none

Content

none

Attributes

Name	Type	Values	Description
name	xs:string	StringCase, NumericalVariance, …	The name must be a valid operator that must match the type of the request it is enclosed in. It is not case sensitive. For a description of all operators see List of Operators.
parameter	xs:NCName	*depends on type of generator*	Optional. Some operators may have a parameter that can be used in order to specify how the operator should work. If no parameter is set for an operator that may have one, its default value for the parameter is used.

Example

The following example requests two operators, *StringCase* and *StringRepetition* to be used by Fuzzino.

```
<operator name="StringCase"/>
<operator name="NumericalVariance" parameter="5"/>
```

request:closeRequest

This element tells Fuzzino that no further values will be requested. After closing a request, no further values can be requested for the closed request.

Parent Elements

request:request

Child Elements

none

Content

none

Attributes

Name	Type	Values	Description
id	xs:ID	*a valid id given in a response from Fuzzino*	Identifier of a request for Fuzzino that shall be closed.

Example

```
<closeRequest id="aValidIdFromAResponse" />
```

List of Generators

for request:string

AllBadStrings
This is just a short cut for the generators *BadLongStrings*, *BadStrings* and *LongStrings*. It is applicable to strings according the following specification:

Attribute	Applicable To
type	*any*
minLength	*any*
maxLength	*any*
nullTerminated	*any*
encoding	*any*

AllXSSGenerator
This is just a short cut for the generators *XSSBasicInput*, *XSSMultipleLinesInput* and *XSSOpenHTMLTagVariance*. It is applicable to strings according the following specification:

Attribute	Applicable To
type	*XSS*
minLength	*any*
maxLength	*any*
nullTerminated	*any*
encoding	*any*

BadDate
This generator creates strings of dates in various formats. The dates have either an invalid form or are part of the edge cases.

Attribute	Applicable To
type	Date
minLength	*any*
maxLength	*any*
nullTerminated	*any*
encoding	*any*

BadFilenames
This generator creates filenames of different lengths and formats, e.g. a long string followed by a "." or a string followed by many repetitions of the file extension ".doc". They are taken from Peach. It is applicable to strings according the following specification:

Attribute	Applicable To
type	Filename Path
minLength	*any*
maxLength	*any*
nullTerminated	*any*
encoding	*any*

BadIpAddresses
This generator creates IPv4 addresses consisting of bad numbers for each part of an IPv4 address and with illegal combinations of "." and numbers . They are taken from Peach. It is applicable to strings according the following specification:

Attribute	Applicable To
type	IPAddress
minLength	*any*
maxLength	*any*
nullTerminated	*any*
encoding	*any*

BadHostnames
This generator creates long hostnames, host names with sequences of "." and similar hostnames including top level domains. They are taken from Peach. It is applicable to strings according the following specification:

Attribute	Applicable To
type	Hostname
minLength	*any*
maxLength	*any*
nullTerminated	*any*
encoding	*any*

BadLongStrings
Bad long strings are strings that are long, up to 20,000 characters, and contain special characters, e.g. the null byte. They are taken from Peach. It is applicable to strings according the following specification:

Attribute	Applicable To
type	*any*
minLength	*any*
maxLength	*any*
nullTerminated	true
encoding	*any*

BadLongUnicodeStrings
This generator creates a list of long Unicode strings that are taken from the fuzzer Peach. It is applicable to strings according the following specification:

Attribute	Applicable To
type	*any*
minLength	*any*
maxLength	*any*
nullTerminated	*any*
encoding	UTF

BadNumbersAsString
This generator provides the values of the generator BadNumbers as string values. It is applicable to strings according the following specification:

Attribute	Applicable To
type	Number
minLength	*any*
maxLength	*any*
nullTerminated	*any*
encoding	*any*

BadPaths
Bad paths are created by combine different special purpose strings for filesystems, e.g. "." and ".." and combines them with directory separators from different operating systems, for instance "/" (from Linux), "\" (from Windows) and ":" (from MacOS). They are taken from Peach. It is applicable to strings according the following specification:

Attribute	Applicable To
type	Filename Path
minLength	*any*
maxLength	*any*
nullTerminated	*any*
encoding	*any*

BadStrings
Bad strings are for instance strings with a special meaning for specific operating systems, e.g. "COM1:", and many other special purpose strings. They are taken from Peach. It is applicable to strings according the following specification:

Attribute	Applicable To
type	*any*
minLength	*any*
maxLength	*any*
nullTerminated	*any*
encoding	*any*

BadTime
Provides a list of bad HTTP time strings. They are taken from Peach. It is applicable to strings according the following specification:

Attribute	Applicable To
type	Time
minLength	*any*
maxLength	*any*
nullTerminated	*any*
encoding	*any*

BadUnicodeUtf8Strings
It is applicable to strings according the following specification:

Attribute	Applicable To
type	String
minLength	*any*
maxLength	*any*
nullTerminated	*any*
encoding	UTF8

CommandInjections
This generator creates some strings having the capability to reveal command injection weaknesses. It is taken from Sulley. It is applicable to strings according the following specification:

Attribute	Applicable To
type	Command
minLength	*any*
maxLength	*any*
nullTerminated	*any*
encoding	*any*

Delimiters
This generator creates usual delimiter values. It is taken from Sulley. It is applicable to strings according the following specification:

Attribute	Applicable To
type	Delimiter
minLength	*any*
maxLength	*any*
nullTerminated	*any*
encoding	*any*

FormatStrings
This generator creates some strings having the capability to reveal format string weaknesses. It is taken from Sulley. It is applicable to strings according the following specification:

Attribute	Applicable To
type	*any*
minLength	*any*
maxLength	*any*
nullTerminated	*any*
encoding	*any*

HTMLFieldInput
This is just a short cut for the generators *AllXSS*, *SQLInjections* and *SQLTimeBasedInjections*. It is applicable to strings according the following specification:

Attribute	Applicable To
type	XSS SQL
minLength	*any*
maxLength	*any*
nullTerminated	*any*
encoding	*any*

LongStrings
This generator creates by itself very long strings, e.g. 10,240 “A”s, and uses the LongSulleySringsGenerator. It is applicable to strings according the following specification:

Attribute	Applicable To
type	*any*
minLength	*any*
maxLength	*any*
nullTerminated	*any*
encoding	*any*

LongSulleyStringsGenerator
This generator creates very long strings based on Sulley’s class string. It is applicable to strings according the following specification:

Attribute	Applicable To
type	*any*
minLength	*any*
maxLength	*any*
nullTerminated	*any*
encoding	*any*

Poplar4DigitPinsGenerator
This generator creates the most popular four digit long pins. It is applicable to strings according the following specification:

Attribute	Applicable To
type	Pin4Digit
minLength	*any*
maxLength	*any*
nullTerminated	*any*
encoding	*any*

SpecialBadIpAddresses
This generator creates special formatted IP addresses. It is applicable to strings according the following specification:

Attribute	Applicable To
type	IPAddress
minLength	*any*
maxLength	*any*
nullTerminated	*any*
encoding	*any*

SpecialUnicodeBomStrings
This generator creates strings with special byte order marks. It is applicable to strings according the following specification:

Attribute	Applicable To
type	any
minLength	*any*
maxLength	*any*
nullTerminated	*any*
encoding	UTF

SQLInjections
This generator creates some strings having the capability to reveal SQL injection weaknesses. It is taken from Sulley. It is applicable to strings according the following specification:

Attribute	Applicable To
type	SQL
minLength	*any*
maxLength	*any*
nullTerminated	*any*
encoding	*any*

SQLTimeBasedInjections
This generator creates strings having the capability to reveal time-based SQL injection weaknesses. It is applicable to strings according the following specification:

Attribute	Applicable To
type	SQL
minLength	*any*
maxLength	*any*
nullTerminated	*any*
encoding	*any*

UnicodeBomStrings
This generator creates different combination of Unicode byte-order-markers of different lengths. It is applicable to strings according the following specification:

Attribute	Applicable To
type	String
minLength	*any*
maxLength	*any*
nullTerminated	*any*
encoding	UTF

XMLInjectionsGenerator
This generator creates strings that can detect XML injection vulnerabilities. It is applicable to strings according the following specification:

Attribute	Applicable To
type	XML
minLength	*any*
maxLength	*any*
nullTerminated	*any*
encoding	*any*

XSSBasicInputGenerator
This generator creates strings that can detect XML injection vulnerabilities. It is applicable to strings according the following specification:

Attribute	Applicable To
type	XSS
minLength	*any*
maxLength	*any*
nullTerminated	*any*
encoding	*any*

XSSMultipleLinesInputGenerator
This generator creates strings that can detect XML injection vulnerabilities. It is applicable to strings according the following specification:

Attribute	Applicable To
type	XSS
minLength	*any*
maxLength	*any*
nullTerminated	*any*
encoding	*any*

XSSOpenHTMLTagVariance
This generator creates strings that can detect XML injection vulnerabilities. It is applicable to strings according the following specification:

Attribute	Applicable To
type	XSS
minLength	*any*
maxLength	*any*
nullTerminated	*any*
encoding	*any*

for request:number

BadFloatGenerator
This generator creates float values of the edge cases.

Attribute	Applicable To
type	integer
minValue	*any*
maxValue	*any*
bits	*any*
signed	*any*

BoundaryNumbers

This generator creates values that are typically minimal and maximal values for the number of bits given by the specification and dividers of it. It is taken from Sulley.

Attribute	Applicable To
type	integer
minValue	*any*
maxValue	*any*
bits	*any*
signed	*any*

List of Operators

for request:string

Delimiter
This operator repeats a delimiter. It is taken from Sulley. It is applicable to strings according the following specification:

Attribute	Applicable To
type	Delimiter
minLength	*any*
maxLength	*any*
nullTerminated	*any*
encoding	*any*

RepeatString
This operator creates different number of repetitions of string in combination with a null character, depending of the encoding. It is applicable to strings according the following specification:

Attribute	Applicable To
type	String
minLength	*any*
maxLength	*any*
nullTerminated	*any*
encoding	*any*

StringCase
This operator changes the capitalization of a string. It works randomly hence using a seed is useful if repeatability is of importance. It is taken from Peach.
It is applicable to strings according the following specification:

Attribute	Applicable To
type	String SQL
minLength	*any*
maxLength	*any*
nullTerminated	*any*
encoding	*any*

for request:number

NumericalVariance

This operator creates values lying around the given valid value. It has a parameter defining the range, default is 10. This operator is taken from Sulley.

Attribute	Applicable To
type	integer
minValue	*any*
maxValue	*any*
bits	*any*
signed	*any*

Responses from Fuzzino

response:response

This is the root element of the xml file. It contains all requests.

Parent Elements

none

Child Elements

response:numberrequest:number (0..*), response:string (0..*), response:errorResponse (0..*), response:closeRequestDone (0..*)

Content

none

Attributes

None

Example

```
<?xml version="1.0" encoding="UTF-8 standalone="yes">
<ns2:response xmlns:ns2="http://fuzzino.fuzzing.fokus.fraunhofer.de/response">
```

response:string

This element contains the response to a string request.

Parent Elements

response:response

Child Elements

response:generatorBased (0..*), response:operatorBased (0..*), response:value

This element contains one fuzzed value.

Parent Elements

response:fuzzedValueresponse:generator

Child Elements

none

Content

This element contains exactly one fuzzed value. The type depends on the original request (see request:string and request:number).
Non-printable and Unicode characters has to be represented in the following format:
\x*nn* for a 8-bit character, e.g. \x00
\u*nnnn* for a 16-bit Unicode character, e.g. \00ef
\U*nnnnnnnn* for a 32-bit Unicode character
A backslash followed by a lowercase *x* or *u* or an uppercase *U* has to be printed two times, e.g. \\x.

Attributes

none

Example

The following example shows a value from a response for a string request.

```
<value xmlns:xsi="http://www.w3.rg/2001/XMLSchema-instance"
      xmlns:xs="http://www.w3.org/2001/XMLSchema" xsi:type="xs:string">
1
</value>
```

response:basedOnValue

This element contains the valid value before it was mutated by an operator.

Parent Elements

response:fuzzedValueresponse:generator

Child Elements

none

Content

This element contains exactly one valid value. The type depends on the original request (see request:string and request:number).
Non-printable and Unicode characters has to be represented in the following format:
\x*nn* for a 8-bit character, e.g. \x00
\u*nnnn* for a 16-bit Unicode character, e.g. \00ef
\U*nnnnnnnn* for a 32-bit Unicode character
A backslash followed by a lowercase *x* or *u* or an uppercase *U* has to be printed two times, e.g. \\x.

Attributes

none

Example

The following example shows the original valid value from a response for a number request.

```
<basedOnValue xmlns:xsi="http://www.w3.rg/2001/XMLSchema-instance"
      xmlns:xs="http://www.w3.org/2001/XMLSchema" xsi:type="xs:long">
1
</basedOnValue>
```

response:kind

This element contains the type of the fuzzedValue.

Parent Elements

response:fuzzedValueresponse:generator

Child Elements

none

Content

This element contains the type of the value. The value can be either of kind “GENERATED”, “MUTATED” or “NONE”.

Attributes

none

Example

The following example shows the kind of a fuzzedValue.

```
<kind>GENERATED</kind>
```

response:warnings (0..1)

Content

none

Attributes

Name	Type	Values	Description
name	xs:NCName	*user-defined, unique*	The user-defined identifier from the request.
id	xs:ID	*unique*	Identifier set by Fuzzino. It can be used to request further values (see request:string for more information on requesting further values).
moreValues	xs:boolean	false true	If true, more values can be requested using the value of the *id* attribute.
seed	xs:string		Optional. The used seed for getting the same results if random-based generators or operators are involved.

Example

The following example shows a response to a string request. The child elements are omitted for readability.

```
<string name="uniqueName" id="IdFromFuzzino" moreValues="true">
    ...
</string>
```

response:number

This element contains the response to a number request.

Parent Elements

response:response

Child Elements

response:generatorBased (0..*), response:operatorBased (0..*), response:value
This element contains one fuzzed value.

Parent Elements

response:fuzzedValueresponse:generator

Child Elements

none

Content

This element contains exactly one fuzzed value. The type depends on the original request (see request:string and request:number).
Non-printable and Unicode characters has to be represented in the following format:
\xnn for a 8-bit character, e.g. \x00
\unnnn for a 16-bit Unicode character, e.g. \00ef
\Unnnnnnnn for a 32-bit Unicode character
A backslash followed by a lowercase *x* or *u* or an uppercase *U* has to be printed two times, e.g. \\x.

Attributes

none

Example

The following example shows a value from a response for a string request.

```
<value xmlns:xsi="http://www.w3.rg/2001/XMLSchema-instance"
      xmlns:xs="http://www.w3.org/2001/XMLSchema" xsi:type="xs:string">
1
</value>
```

response:basedOnValue

This element contains the valid value before it was mutated by an operator.

Parent Elements

response:fuzzedValueresponse:generator

Child Elements

none

Content

This element contains exactly one valid value. The type depends on the original request (see request:string and request:number).
Non-printable and Unicode characters has to be represented in the following format:
\xnn for a 8-bit character, e.g. \x00
\unnnn for a 16-bit Unicode character, e.g. \00ef
\Unnnnnnnn for a 32-bit Unicode character
A backslash followed by a lowercase *x* or *u* or an uppercase *U* has to be printed two times, e.g. \\x.

Attributes

none

Example

The following example shows the original valid value from a response for a number request.

```
<basedOnValue xmlns:xsi="http://www.w3.rg/2001/XMLSchema-instance"
      xmlns:xs="http://www.w3.org/2001/XMLSchema" xsi:type="xs:long">
1
</basedOnValue>
```

response:kind

This element contains the type of the fuzzedValue.

Parent Elements

response:fuzzedValueresponse:generator

Child Elements

none

Content

This element contains the type of the value. The value can be either of kind “GENERATED”, “MUTATED” or “NONE”.

Attributes

none

Example

The following example shows the kind of a fuzzedValue.

```
<kind>GENERATED</kind>
```

response:warnings (0..1)

Content

none

Attributes

Name	Type	Values	Description
name	xs:NCName	*user-defined, unique*	The user-defined identifier from the request.
id	xs:ID	*unique*	Identifier set by Fuzzino. It can be used to request further values (see request:number for more information on requesting further values).
moreValues	xs:boolean	false true	If true, more values can be requested using the value of the *id* attribute.
seed	xs:string		The used seed for getting the same results if random-based generators or operators are involved.

Example

The following example shows a response to a number request. The child elements are omitted for readability.

```
<number name="uniqueName" id="IdFromFuzzino" moreValues="true">
    ...
</number>
```

response:generatorBased

This element contains all values that are created by generators.

Parent Elements

response:number, response:string

Child Elements

response:generator (0..*)

Content

none

Attributes

none

Example

```
<generatorBased>
    ...
</generatorBased>
```

response:generator

This element contains all fuzzed values that are created by the same generator.

Parent Elements

response:generatorBased

Child Elements

response:fuzzedValue (0..*)

Content

none

Attributes

Name	Type	Values	Description
name	xs:NCName		The name of the generator all contained fuzzed values are created by (see List of Generators on page 16).

Example

```
<generator name="LongStrings">
    ...
</generator>
```

response:operatorBased

This element contains all values that are created by operators.

Parent Elements

response:number, response:string

Child Elements

response:operator (1..*)

Content

none

Attributes

none

Example

```
<operatorBased>
    ...
</operatorBased>
```

response:operator

This element contains all fuzzed values that are created by the same operator that was applied on the same valid value in different ways.

Parent Elements

response:operatorBased

Child Elements

response:fuzzedValue (0..*)

Content

none

Attributes

Name	Type	Values	Description
name	xs:NCName		The name of the operator all contained fuzzed values are created by (see List of Operators on page 24).
basedOn	xs:string		The operator specified in the *name* attribute was applied on this valid value that was taken from the original request.

Example

```
<operator name="StringCase" basedOn="ABC">
    ...
</operator>
```

response:fuzzedValue

This element contains the fuzzed value. Every fuzzed value contains an element with the generated or mutated value and an element with the kind of the value. If the value was mutated by an operator, then there is also an element with the unmodified value.

Parent Elements

response:generator

Child Elements

response:value (1), response:basedOnValue (0..1), response:kind (1)

Content

none

Attributes

none

Example

```
<fuzzedValue>
    ...
</fuzzedValue>
```

response:value

This element contains one fuzzed value.

Parent Elements

response:fuzzedValueresponse:generator

Child Elements

none

Content

This element contains exactly one fuzzed value. The type depends on the original request (see request:string and request:number).
Non-printable and Unicode characters has to be represented in the following format:
\x*nn* for a 8-bit character, e.g. \x00
\u*nnnn* for a 16-bit Unicode character, e.g. \00ef
\U*nnnnnnnn* for a 32-bit Unicode character
A backslash followed by a lowercase *x* or *u* or an uppercase *U* has to be printed two times, e.g. \\x.

Attributes

none

Example

The following example shows a value from a response for a string request.

```
<value xmlns:xsi="http://www.w3.rg/2001/XMLSchema-instance"
      xmlns:xs="http://www.w3.org/2001/XMLSchema" xsi:type="xs:string">
1
</value>
```

response:basedOnValue

This element contains the valid value before it was mutated by an operator.

Parent Elements

response:fuzzedValueresponse:generator

Child Elements

none

Content

This element contains exactly one valid value. The type depends on the original request (see request:string and request:number).
Non-printable and Unicode characters has to be represented in the following format:
\x*nn* for a 8-bit character, e.g. \x00
\u*nnnn* for a 16-bit Unicode character, e.g. \00ef
\U*nnnnnnnn* for a 32-bit Unicode character
A backslash followed by a lowercase *x* or *u* or an uppercase *U* has to be printed two times, e.g. \\x.

Attributes

none

Example

The following example shows the original valid value from a response for a number request.

```
<basedOnValue xmlns:xsi="http://www.w3.rg/2001/XMLSchema-instance"
      xmlns:xs="http://www.w3.org/2001/XMLSchema" xsi:type="xs:long">
1
</basedOnValue>
```

response:kind

This element contains the type of the fuzzedValue.

Parent Elements

response:fuzzedValueresponse:generator

Child Elements

none

Content

This element contains the type of the value. The value can be either of kind “GENERATED”, “MUTATED” or “NONE”.

Attributes

none

Example

The following example shows the kind of a fuzzedValue.

```
<kind>GENERATED</kind>
```

response:warnings

This element contains warnings resulting from malformed requests.

Parent Elements

response:number, response:string, response:closeRequestDone

Child Elements

response:illegalGenerator (0..*), response:illegalOperator (0..*), response:illegalRequestFormat (0..*)

Content

none

Attributes

Name	Type	Values	Description
hasMoreValues	xs:boolean	false, true	This attribute indicates that a request for more values is unsuccessful because there are no more values available. This could be the case for two reasons: either because all values are already requested or the request was closed using the element request:closeRequest.

Example

```
<string ...>
    ...
    <warnings>
        <!-- some warnings here -->
    </warnings
</string>
```

response:illegalGenerator

This element denotes a generator that was illegally requested – either because it is unknown or not applicable in respect to the specification of the request.

Parent Elements

response:value

This element contains one fuzzed value.

Parent Elements

response:fuzzedValueresponse:generator

Child Elements

none

Content

This element contains exactly one fuzzed value. The type depends on the original request (see request:string and request:number).
Non-printable and Unicode characters has to be represented in the following format:
\x*nn* for a 8-bit character, e.g. \x00
\u*nnnn* for a 16-bit Unicode character, e.g. \00ef
\U*nnnnnnnn* for a 32-bit Unicode character
A backslash followed by a lowercase *x* or *u* or an uppercase *U* has to be printed two times, e.g. \\x.

Attributes

none

Example

The following example shows a value from a response for a string request.

```
<value xmlns:xsi="http://www.w3.rg/2001/XMLSchema-instance"
      xmlns:xs="http://www.w3.org/2001/XMLSchema" xsi:type="xs:string">
1
</value>
```

response:basedOnValue

This element contains the valid value before it was mutated by an operator.

Parent Elements

response:fuzzedValueresponse:generator

Child Elements

none

Content

This element contains exactly one valid value. The type depends on the original request (see request:string and request:number).
Non-printable and Unicode characters has to be represented in the following format:
\x*nn* for a 8-bit character, e.g. \x00
\u*nnnn* for a 16-bit Unicode character, e.g. \00ef
\U*nnnnnnnn* for a 32-bit Unicode character
A backslash followed by a lowercase *x* or *u* or an uppercase *U* has to be printed two times, e.g. \\x.

Attributes

none

Example

The following example shows the original valid value from a response for a number request.

```
<basedOnValue xmlns:xsi="http://www.w3.rg/2001/XMLSchema-instance"
      xmlns:xs="http://www.w3.org/2001/XMLSchema" xsi:type="xs:long">
1
</basedOnValue>
```

response:kind

This element contains the type of the fuzzedValue.

Parent Elements

response:fuzzedValueresponse:generator

Child Elements

none

Content

This element contains the type of the value. The value can be either of kind “GENERATED”, “MUTATED” or “NONE”.

Attributes

none

Example

The following example shows the kind of a fuzzedValue.

```
<kind>GENERATED</kind>
```

response:warnings

Child Elements

none

Content

none

Attributes

Name	Type	Values	Description
name	xs:string		The name of the illegal generator.
reason	xs:NCName	unknown notApplicable	The reason why the generator was illegal.

Example

```
<illegalGenerator name="unknownGeneratorName" reason="unknown"/>
```

```
<illegalGenerator name="UnicodeBomStrings" reason="notApplicable"/>
```

response:illegalOperator

This element denotes an operator that was illegally requested – either because it is unknown or not applicable in respect to the specification of the request.

Parent Elements

response:value

This element contains one fuzzed value.

Parent Elements

response:fuzzedValueresponse:generator

Child Elements

none

Content

This element contains exactly one fuzzed value. The type depends on the original request (see request:string and request:number).
Non-printable and Unicode characters has to be represented in the following format:
\x*nn* for a 8-bit character, e.g. \x00
\u*nnnn* for a 16-bit Unicode character, e.g. \00ef
\U*nnnnnnnn* for a 32-bit Unicode character
A backslash followed by a lowercase *x* or *u* or an uppercase *U* has to be printed two times, e.g. \\x.

Attributes

none

Example

The following example shows a value from a response for a string request.

```
<value xmlns:xsi="http://www.w3.rg/2001/XMLSchema-instance"
      xmlns:xs="http://www.w3.org/2001/XMLSchema" xsi:type="xs:string">
1
</value>
```

response:basedOnValue

This element contains the valid value before it was mutated by an operator.

Parent Elements

response:fuzzedValueresponse:generator

Child Elements

none

Content

This element contains exactly one valid value. The type depends on the original request (see request:string and request:number).
Non-printable and Unicode characters has to be represented in the following format:
\x*nn* for a 8-bit character, e.g. \x00
\u*nnnn* for a 16-bit Unicode character, e.g. \00ef
\U*nnnnnnnn* for a 32-bit Unicode character
A backslash followed by a lowercase *x* or *u* or an uppercase *U* has to be printed two times, e.g. \\x.

Attributes

none

Example

The following example shows the original valid value from a response for a number request.

```
<basedOnValue xmlns:xsi="http://www.w3.rg/2001/XMLSchema-instance"
      xmlns:xs="http://www.w3.org/2001/XMLSchema" xsi:type="xs:long">
1
</basedOnValue>
```

response:kind

This element contains the type of the fuzzedValue.

Parent Elements

response:fuzzedValueresponse:generator

Child Elements

none

Content

This element contains the type of the value. The value can be either of kind “GENERATED”, “MUTATED” or “NONE”.

Attributes

none

Example

The following example shows the kind of a fuzzedValue.

```
<kind>GENERATED</kind>
```

response:warnings

Child Elements

none

Content

none

Attributes

Name	Type	Values	Description
name	xs:string		The name of the illegal operator.
reason	xs:NCName	unknown notApplicable	The reason why the operator was illegal.

Example

```
<illegalGenerator name="unknownOperatorName" reason="unknown"/>
```

```
<illegalGenerator name="DelimiterOperator" reason="notApplicable"/>
```

response:illegalRequestFormat

This element indicates an invalid request and denotes the elements and the attribute that was malformed or that is missing.

Parent Elements

response:value

This element contains one fuzzed value.

Parent Elements

response:fuzzedValueresponse:generator

Child Elements

none

Content

This element contains exactly one fuzzed value. The type depends on the original request (see request:string and request:number).
Non-printable and Unicode characters has to be represented in the following format:
\x*nn* for a 8-bit character, e.g. \x00
\u*nnnn* for a 16-bit Unicode character, e.g. \00ef
\U*nnnnnnnn* for a 32-bit Unicode character
A backslash followed by a lowercase *x* or *u* or an uppercase *U* has to be printed two times, e.g. \\x.

Attributes

none

Example

The following example shows a value from a response for a string request.

```
<value xmlns:xsi="http://www.w3.rg/2001/XMLSchema-instance"
      xmlns:xs="http://www.w3.org/2001/XMLSchema" xsi:type="xs:string">
1
</value>
```

response:basedOnValue

This element contains the valid value before it was mutated by an operator.

Parent Elements

response:fuzzedValueresponse:generator

Child Elements

none

Content

This element contains exactly one valid value. The type depends on the original request (see request:string and request:number).
Non-printable and Unicode characters has to be represented in the following format:
\x*nn* for a 8-bit character, e.g. \x00
\u*nnnn* for a 16-bit Unicode character, e.g. \00ef
\U*nnnnnnnn* for a 32-bit Unicode character
A backslash followed by a lowercase *x* or *u* or an uppercase *U* has to be printed two times, e.g. \\x.

Attributes

none

Example

The following example shows the original valid value from a response for a number request.

```
<basedOnValue xmlns:xsi="http://www.w3.rg/2001/XMLSchema-instance"
      xmlns:xs="http://www.w3.org/2001/XMLSchema" xsi:type="xs:long">
1
</basedOnValue>
```

response:kind

This element contains the type of the fuzzedValue.

Parent Elements

response:fuzzedValueresponse:generator

Child Elements

none

Content

This element contains the type of the value. The value can be either of kind “GENERATED”, “MUTATED” or “NONE”.

Attributes

none

Example

The following example shows the kind of a fuzzedValue.

```
<kind>GENERATED</kind>
```

response:warnings

Child Elements

none

Content

none

Attributes

Name	Type	Values	Description
element	xs:NCName		Optional. The name of the malformed element.
attribute	xs:NCName		Optional. The name of the malformed attribute.
missingElement	xs:NCName		Optional. The name of the missing element in the request.
missingAttribute	xs:NCName		Optional. The name of the missing element in the request.

Example

The following denotes an illegal request where the attribute bits of the specification element has in illegal value:

```
<illegalRequest element="specification" attribute="bits" />
```

The following denotes an illegal request where the attribute bits of the specification element is missing:

```
<illegalRequest element="specification" missingAttribute="bits" />
```

response:errorResponse

This element denotes that the request was malformed so that it can't be read by the XML parser. This element contains the error message of the parser.

Parent Elements

response:response

Child Elements

none

Content

none

Attributes

Name	Type	Values	Description
reason	xs:string	*specific*	Optional. The reason of the error.
message	xs:string	*specific*	Optional. The error message.
stackTrace	xs:string	*specific*	Optional. The Java stacktrace of the error.

Example

The following example shows an errorResponse where the content is left out for readability.

```
<errorResponse message="..." stackTrace="..."/>
```

response:closeRequestDone

This element indicates that the denoted request was successfully closed.

Parent Elements

response:response

Child Elements

response:value

This element contains one fuzzed value.

Parent Elements

response:fuzzedValueresponse:generator

Child Elements

none

Content

This element contains exactly one fuzzed value. The type depends on the original request (see request:string and request:number).
Non-printable and Unicode characters has to be represented in the following format:
\x*nn* for a 8-bit character, e.g. \x00
\u*nnnn* for a 16-bit Unicode character, e.g. \00ef
\U*nnnnnnnn* for a 32-bit Unicode character
A backslash followed by a lowercase *x* or *u* or an uppercase *U* has to be printed two times, e.g. \\x.

Attributes

none

Example

The following example shows a value from a response for a string request.

```
<value xmlns:xsi="http://www.w3.rg/2001/XMLSchema-instance"
      xmlns:xs="http://www.w3.org/2001/XMLSchema" xsi:type="xs:string">
1
</value>
```

response:basedOnValue

This element contains the valid value before it was mutated by an operator.

Parent Elements

response:fuzzedValueresponse:generator

Child Elements

none

Content

This element contains exactly one valid value. The type depends on the original request (see request:string and request:number).
Non-printable and Unicode characters has to be represented in the following format:
\x*nn* for a 8-bit character, e.g. \x00
\u*nnnn* for a 16-bit Unicode character, e.g. \00ef
\U*nnnnnnnn* for a 32-bit Unicode character
A backslash followed by a lowercase *x* or *u* or an uppercase *U* has to be printed two times, e.g. \\x.

Attributes

none

Example

The following example shows the original valid value from a response for a number request.

```
<basedOnValue xmlns:xsi="http://www.w3.rg/2001/XMLSchema-instance"
      xmlns:xs="http://www.w3.org/2001/XMLSchema" xsi:type="xs:long">
1
</basedOnValue>
```

response:kind

This element contains the type of the fuzzedValue.

Parent Elements

response:fuzzedValueresponse:generator

Child Elements

none

Content

This element contains the type of the value. The value can be either of kind “GENERATED”, “MUTATED” or “NONE”.

Attributes

none

Example

The following example shows the kind of a fuzzedValue.

```
<kind>GENERATED</kind>
```

response:warnings (0..1)

Content

none

Attributes

Name	Type	Values	Description
name	xs:NCName		The unique name of the request from the user of Fuzzino.
id	xs:NCName		The unique identifier for the request from Fuzzino.

Example

```
<closeRequestDone name="uniqueName" id="IdFromFuzzino" />
```