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, number requests, structure requests, and collection 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 `useNoGenerators` 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).

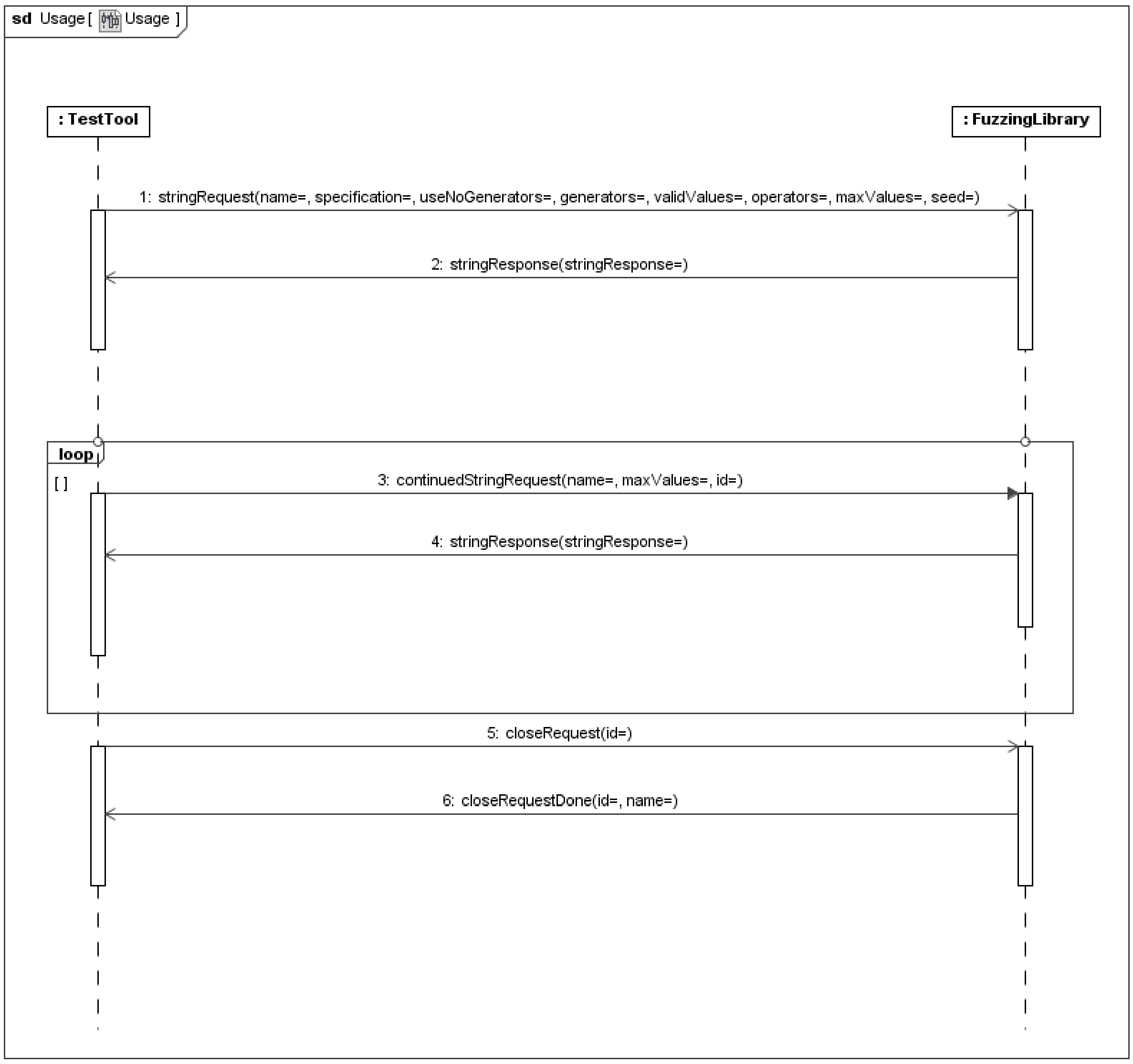
sd Usage [Usage]
: TestTool
: FuzzingLibrary
1: stringRequest(name=, specification=, useNoGenerators=, generators=, validValues=, operators=, maxValues=, seed=)
2: stringResponse(stringResponse=)
loop
[ ]
3: continuedStringRequest(name=, maxValues=, id=)
4: stringResponse(stringResponse=)
5: closeRequest(id=)
6: closeRequestDone(id=, name=)

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:collection (0..*), request:number (0..*), request:string (0..*), request:structure (0..*)

Content

none

Attributes

XML standard

Example

```
<request xmlns="http://fuzzino.fuzzing.fokus.fraunhofer.de/request">
```

```
<request xmlns="http://fuzzino.fuzzing.fokus.fraunhofer.de/request"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://fuzzino.fuzzing.fokus.fraunhofer.de/request
    ./fuzzingRequest.xsd">
```

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

The content of this element must be a valid generator 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 on page 23.

Attributes

Name	Type	Values	Description
param	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>LongStrings</generator>
<generator>BadStrings</generator>
```

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

none

Attributes

none

Example

```
<noGenerators/>
```

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*	Optional. 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:specification, request:generator (0..*), 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

The content of this element must be a valid generator 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 on page 23.

Attributes

Name	Type	Values	Description
param	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>LongStrings</generator>
<generator>BadStrings</generator>
```

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

none

Attributes

none

Example

```
<noGenerators/>
```

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*	Optional. 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:structure

This element opens a request for a data structure to be fuzzed by Fuzzino. This type determines the fuzzing generators and operators that shall be used for generating fuzzed data structures.

Parent Elements

request:request

Child Elements

request:specification, request:operator (0..*)

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.
fuzzStructure	xs:boolean	false true	Specifies if the structure of the data structure shall be fuzzed, meaning whether the order and appearance of the fields within the data structure shall be changed. Must be *true* if *fuzzValues* is *false*.
minMutations	xs:int		Optional. The minimum number of mutations applied to each fuzzed structure. If this attribute is not set, it has the default value *1*. The number of mutations refers to structural mutations. If *fuzzStructure* is *false*, this attribute is ignored and can be omitted.
maxMutations	xs:int		The maximum number of mutations applied to each fuzzed structure. The number of mutations refers to structural mutations. If *fuzzStructure* is *false*, this attribute is ignored and can be omitted.
fuzzValues	xs:boolean	false true	Specifies whether the values of the fields of the data structure shall be fuzzed. Must be *true* if *fuzzStructure* is *false*.
id	xs:NCName	*a valid id given in a response from Fuzzino*	Optional. Identifier for Fuzzino to request more values. Can only be set if an initial request has been taken place.

Name	Type	Values	Description
			If set, all child elements of this request are ignored and the values of the attributes *minMutations*, *maxMutations*, *fuzzStructure* and *fuzzValues* are ignored and can be omitted.
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 data structure to be fuzzed. The child elements are omitted for readability.

```
<structure name="uniqueName"
           minMutations="1"
           maxMutations="3"
           fuzzStructure="true"
           fuzzValues="false"
           maxValues="50">
    ...
</structure>
```

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.

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

request:collection

This element opens a request for a collection to be fuzzed by Fuzzino. This type determines the fuzzing generators and operators that shall be used for generating fuzzed collections.

Parent Elements

request:request

Child Elements

request:specification, request:validCollections (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.
minMutations	xs:int		Optional. The minimum number of mutations applied to each fuzzed collection. If this attribute is not set, it has the default value *1*.
maxMutations	xs:int		The maximum number of mutations applied to each fuzzed collection.
fuzzCollection	xs:boolean	false true	Specifies if the collection itself shall be fuzzed, meaning whether the order and appearance of the elements within the collection shall be changed. Must be *true* if *fuzzValues* is *false*.
fuzzValues	xs:boolean	false true	Specifies whether the values of elements of the collection shall be fuzzed. Must be *true* if *fuzzCollection* is *false*.
id	xs:NCName	*a valid id given in a response from Fuzzino*	Optional. 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 and the values of the attributes *minMutations*, *maxMutations*, *fuzzCollection* and *fuzzValues* are ignored and can be omitted.
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 collection to be fuzzed. The child elements are omitted for readability.

```
<collection name="uniqueName"
            maxValues="50"
            minMutations="1"
            maxMutations="3"
            fuzzCollection="true"
            fuzzValues="true">
    ...
</collection>
```

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.

```
<collection 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. For a *request:string* this element is optional.

Parent Elements

request:collection , request:number, request:string, request:structure

Child Elements

none for *request:collection*, *request:number*, *request:string*
request:field (1..*) for *request:structure*

Content

If the parent element is *request:string* and the value of the attribute *type* is either *RegExValid* or *RegExInvalid* the content is a regular expression (currently under the constraint that the regular expression may not allow a variable length).

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 section *Contents* for more information). *RegExInvalid* is set to describe invalid values with a regular expression in the content of this *request:specification* element (see section *Contents* 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

Name	**Type**	**Values**	**Description**
			conveyed to the command line. *Date* and *Time* refers to strings for date and time specification. *IPAddress* refers to addresses according 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*.

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

Name	**Type**	**Values**	**Description**
type	xs:NCName	integer float	Specifies whether the number is an integer or a decimal number.
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*.

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

Name	**Type**	**Values**	**Description**
ordered	xs:boolean	false true	Specifies whether the order of the *request:field*s within the data structure is relevant.

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

Name	Type	Values	Description
ref	xs:IDREF		References another request as a specification for this field. The value of this attribute must match the name of another request. It may not reference the enclosing request.
unique	xs:boolean	false true	Specifies if each value of the collection must be unique.
ordered	xs:Boolean	false true	Specifies if the order of values within the collection is relevant.
minLength maxLength	xs:int		Specifies the minimal respectively the maximal number of elements in the collection.

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.

A valid specification for *request:structure* could be the following (the child elements are omitted for readability):

```
<specification ordered="true">
    …
</specification>
```

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

The content of this element must be a valid generator 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 on page 23.

Attributes

Name	Type	Values	Description
param	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>LongStrings</generator>
<generator>BadStrings</generator>
```

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

none

Attributes

none

Example

```
<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:validCollections

This element indicates that valid collections are available for Fuzzino. If this element is present, at least one valid collection 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:collection

Child Elements

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

Content

none

Attributes

none

Example

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

```
<validCollections>
    <validCollection name="myCollectionName">
    </collection>
</validCollections>
```

request:validCollection

This element represents a valid collection made up of its child elements.

Parent Elements

request:validCollections

Child Elements

request:value (0..*)

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.

Example

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

```
<validCollection name="uniqueName" />
```

request:value

This element contains a valid value of a request.

Parent Elements

request:validCollection, 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

The content of this element must be a valid generator 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 on page 23.

Attributes

Name	Type	Values	Description
param	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>LongStrings</generator>
<generator>BadStrings</generator>
```

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

none

Attributes

none

Example

```
<noGenerators/>
```

request:validValues

Child Elements

none

Content

The content of this element is exactly one valid value of a request.
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

```
<value>A\x00A\x00A\x00</value>
```

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:structure, request:validCollections, 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

The content of this element must be a valid generator 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 on page 23.

Attributes

Name	Type	Values	Description
param	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>LongStrings</generator>
<generator>BadStrings</generator>
```

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

none

Attributes

none

Example

```
<noGenerators/>
```

request:validValues

Child Elements

none

Content

The content of this element 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 on page 28.

Attributes

Name	Type	Values	Description
param	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>StringCase</operator>
<operator param="5">NumericalVariance</operator>
```

request:field

This element specifies one field of a data structure. It refers to request for collection, number, string or structure by their *name* attribute.

Parent Elements

request:specification

Child Elements

none

Content

none

Attributes

Name	Type	Values	Description
ref	xs:IDREF	*reference to a name of a request*	References another request as a specification for this field. The value of this attribute must match the *name* attribute of another request. It may not reference the enclosing request.
fuzz	xs:boolean	false true *(default)*	Optional. Specifies if the value of this field shall be fuzzed. If this attribute is not set, it has the default value *true*.

Example

The following example shows a field specification of a structure request. It refers to another request with the name *nameOfARequest* that specifies the type of the field

```
<field ref="nameOfARequest" fuzz="false" />
```

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	String SQL Number
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*

Delimiter
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*

LongStrings
This generator creates just very long strings, e.g. 10,240 “A”s. It is 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*

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*

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

for request:number

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*

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*

StringRepetition
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*

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

request:collection (0..*), request:number (0..*), request:string (0..*), request:structure (0..*), response:error (0..*)

Content

none

Attributes

None

Example

```
<response xmlns="http://fuzzino.fuzzing.fokus.fraunhofer.de/response">
```

```
<response xmlns="http://fuzzino.fuzzing.fokus.fraunhofer.de/response"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://fuzzino.fuzzing.fokus.fraunhofer.de/response
    ./fuzzingResponse.xsd">
```

response:string

This element contains the response to a string request.

Parent Elements

response:response

Child Elements

response:generatorBased (0..*), response:operatorBased (0..*), 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 on page 5 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 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: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 on page 6 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:structure

This element contains the response to a structure request.

Parent Elements

response:response

Child Elements

response:fuzzedStructure (0..*), 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:structure on page 7 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 structure request. The child elements are omitted for readability.

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

response:collection

This element contains the response to a collection request.

Parent Elements

response:response

Child Elements

response:fuzzedCollection (0..*), 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:collection on page 9 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 collection request. The child elements are omitted for readability.

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

response:generatorBased

This element contains all values that are created by generators.

Parent Elements

response:number, response:string

Child Elements

response:generator (1..*)

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 (1..*)

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 23).

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 (1..*)

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 28).
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:fuzzedStructure

This element contains one fuzzed structure.

Parent Elements

response:structure

Child Elements

response:field (0..*)

Content

none

Attributes

Name	Type	Values	Description
operators	xs:NCName		The value of this attribute is a comma-separated list of applied operators.
mutations	xs:int		This attribute indicates how many times the operators were applied to the original structure. The value is at least the number of operators that are given in the attribute *operators*.

Example

```
<fuzzedStructure operators="RemoveField,RepeatField" mutations="2">
    ...
</fuzzedStructure>
```

response:field

This element specifies one field of a fuzzed data structure. It refers to a collection, number, string or structure by their *name* attribute.

Parent Elements

response:fuzzedStructure

Child Elements

none

Content

none

Attributes

Name	Type	Values	Description
ref	xs:IDREF	*reference to a name of a response*	References a response for fuzzed values. The value of this attribute matches the *name* attribute of another response.
fuzz	xs:boolean	false true	Specifies if the value of this field is fuzzed. The value is identical to the one of the request.

Example

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

```
<field ref="nameOfARequest" fuzz="false" />
```

response:fuzzedCollection

This element contains all elements of a fuzzed collection.

Parent Elements

response:collection

Child Elements

response:value (0..*)

Content

none

Attributes

Name	Type	Values	Description
operators	xs:NCName		The value of this attribute is a comma-separated list of applied operators.
mutations	xs:int		This attribute indicates how many times the operators were applied to the original collection. The value is at least the number of operators that are given in the attribute *operators*.
basedOn	xs:string		Refers to the *name* attribute of a valid collection from the request (see request:validCollection on page 18). The referred collection was fuzzed by applying the operators specified in the *operators* attribute as often as denoted in the *mutations* attribute.

Example

```
<fuzzedCollection operators="CollectionLength" mutations="1"
                  basedOn="nameOfAValidCollection">
    ...
</fuzzedCollection>
```

response:fuzzedValue

This element contains one fuzzed value.

Parent Elements

response:generator

Child Elements

none

Content

This element contains exactly one fuzzed value. The type depends on the original request (see request:string on page 5 and request:number on page 6).
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

```
<fuzzedValue>A\x00A\x00</fuzzedValue>
```

response:value

This element contains one value of a collection – either fuzzed or not.

Parent Elements

response:fuzzedCollection

Child Elements

none

Content

This element contains exactly one value of a collection. The type depends on the original request (see request:string on page 5 and request:number on page 6).
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

Name	Type	Values	Description
generator	xs:NCName		The name of the generator that created this value (see List of Generators on page 23).
basedOn	xs:string		The operator specified in the *operator* attribute was applied on this value that was taken from the original request.
operator	xs:NCName		The name of the operator that created this value by being applied to the valid value denoted in the *basedOn* attribute. (see List of Operators on page 28).

Example

The following example shows a fuzzed collection with 5 values where the first 3 values wasn't fuzzed, the fourth value is fuzzed by applying the operator *StringCase* to the value *Value3* and the fifth value was created by the generator *BadStrings*.

```
<fuzzedCollection operators="CollectionLength"
                  mutations="2"
                  basedOn="nameOfAValidCollection">
    <value>Value1</value>
    <value>Value2</value>
    <value>Value3</value>
    <value basedOn="Value3" operator="StringCase">vALue3</value>
    <value generator="BadStrings">
        A\x00A\x00
    </value>
</fuzzedCollection>
```

response:warnings

This element contains warnings resulting from malformed requests.

Parent Elements

response:collection, response:number, response:string, response:structure

Child Elements

response:illegalGenerator (0..*), response:illegalOperator (0..*),
response:illegalRequestFormat (0..*), response:noMoreValuesAvailable (0..1)

Content

none

Attributes

none

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:warnings

Child Elements

none

Content

The name of the generator that was illegally requested.

Attributes

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

Example

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

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

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:warnings

Child Elements

none

Content

The name of the operator that was illegally requested.

Attributes

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

Example

```
<illegalOperator reason="unknown">
    unknownGeneratorName
</illegalOperator>
```

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: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:noMoreValuesAvailable

This element 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 (see page 22).

Parent Elements

response:warnings

Child Elements

none

Content

none

Attributes

none

Example

response:error

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

The error message of the parser.

Attributes

Name	Type	Values	Description
line	xs:int		Optional. The line number where the error occurred.
column	xs:int		Optional. The column number where the error occurred.

Example

The following error denotes a malformed request where the element *validValue* is not correctly closed.

```
<error line="11" column="7">
    The element type "validValues" must be terminated by the matching
    end-tag "&lt;/validValues&gt;".
</error>
```

response:closeRequestDone

This element indicates that the denoted request was successfully closed.

Parent Elements

response:response

Child Elements

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" />
```