

# smatrix

## Expert manual

Applicable for smatrix Windows - Version 1.9

All rights reserved especially (also in extracts) for translation, reprinting, reproduction by copying or other technical means.

dawin GmbH  
Seligenthaler Str. 5  
53721 Siegburg  
Germany

1	Introduction.....	2
2	Customization files .....	2
2.1	Locations.....	2
2.2	Language dependency.....	2
2.3	Order of precedence .....	3
2.4	Editing XML files .....	3
3	Customizing names and value ranges .....	4
3.1	ID column names.....	5
3.2	Subsample group names .....	6
3.3	Trait names.....	6
3.4	Value Ranges .....	9
3.4.1	Numeric value range .....	9
3.4.2	Text values.....	10
3.4.3	Date value range.....	11
3.4.4	Code value range .....	12
3.4.5	Generator Prefix/Suffix.....	14
3.4.6	Combined value ranges .....	16
3.5	id Attributes.....	18
3.6	Pronunciations.....	18
4	Customizing Autoconfiguration.....	19
4.1	Autoconfiguration for #subsamples.....	21
5	APPENDIX .....	21
5.1	smatrix configuration XML syntax.....	21
5.2	Pitfalls .....	24
5.3	Counting Trial .....	25

## 1 Introduction

This manual explains the aspects of the application that can be customized. You can

- define new traits or ID columns,
- adapt their pronunciations,
- define custom value ranges,
- and improve autoconfiguration.

## 2 Customization files

smatrix always uses two customization files: a “headers” file and a “values” file.

Depending on the currently used language, smatrix uses different versions of these files. There is also a fallback to default files if no customization was done by the user. Please consider the following sections to find the correct customization files.

### 2.1 Locations

smatrix considers the following locations when looking for customization files:

- “C:\ProgramData\smatrix\templates” contains the default files  
**Important: Do not change any files in this directory, because they will be overwritten when installing newer versions of smatrix or by un- / re-installing the current version!**
- “C:\ProgramData\smatrix\data”: To customize smatrix, copy the corresponding default files to this directory. It is safe to edit files there, as files stored in this directory will not be removed or overwritten when smatrix is un- / re-installed!

**Note:** The folder “C:\ProgramData\smatrix\data\configuration templates” is used to store user defined templates i.e., parameter files are stored here when a user saves a configuration as a template.

### 2.2 Language dependency

As mentioned, there are language dependent versions of the customization files:

- headers\_<language>.xml
- values\_<language>.xml

Where <language> is a two-letter language code according to this table:

de	German
en	English (used for US and UK, India & Australia)
fr	French
it	Italian
nl	Dutch
pt	Portuguese (used for Brazil and Portugal)
sp	Spanish (used for Mexico and Spain)

Note: The customization files do not differentiate between the language variants of English and Portuguese.

### 2.3 Order of precedence

smatrix will use the customized files in “C:\ProgramData\smatrix\data” if found. Otherwise, the default files in “C:\ProgramData\smatrix\templates” will be used.

Note: These files are read only once when starting smatrix. When changing the headers/values file, a re-start of smatrix is required for the changes to take effect.

### 2.4 Editing XML files

The customization files of smatrix are using XML (extensible markup language) format. This section provides a short overview on how XML works.

- XML files are regular text files. You can open them with any editor that can read “.txt” files.
  - Windows by default contains the program “Notepad” that can be used to view and edit xml files. However, it is not very comfortable to use.
  - Other free available programs like [Notepad++](#) make it easier to read/edit xml files. They support e.g., highlighting the XML syntax.
- The difference between TXT and XML is only the content of the file.
  - The content of TXT files is not structured. You can put in any text you like the way you like.
  - XML content is structured data. There are a few rules (the syntax) that need to be followed when writing XML. This allows a computer to interpret the file.
- XML Syntax
 

To keep it simple, we consider the example of describing a bike in xml.

  - The bike can be defined as an element `<bike></bike>`, where

- `<bike>` is a start-tag  
A word in angle brackets marking the beginning
- `</bike>` is the end-tag  
Same as the start-tag, but with a slash before the word. It marks the end.
- The above element `<bike></bike>` is valid xml. If that is all we want to describe, the same can be simplified writing: `<bike/>`
- However, you might want to add more detail by defining its parts (like tires). This can be done by adding further elements like `<tire></tire>` in between start-tag and end-tag of the bike. That makes clear, that they belong to the bike.
 

```
<bike>
  <tire></tire>
  <tire></tire>
</bike>
```
- With the same approach you can add further elements within others, or next to others on the same level (like the tires).
- It becomes obvious that you need to enrich the elements with information, e.g., to differentiate the tires. That can be done in two ways
  - Adding content in an element using plain text between the start-/end-tags.
 

```
<tire>front</tire>
```

Note: If you add content, you cannot add further elements within.
  - Adding attributes
 

```
<bike producer="Scott"></bike>
```

    - Attributes are name-value pairs written as `name="value"`
    - They are added within the start marker, before the closing bracket
    - Note: multiple attributes can be added to an element, separated by comma: `<bike producer="Scott", productionYear="2019"></bike>`
- Putting the above together, the example can look like this:
 

```
<bike producer="Scott", productionYear="2019">
  <tire>front</tire>
  <tire>rear</tire>
</bike>
```

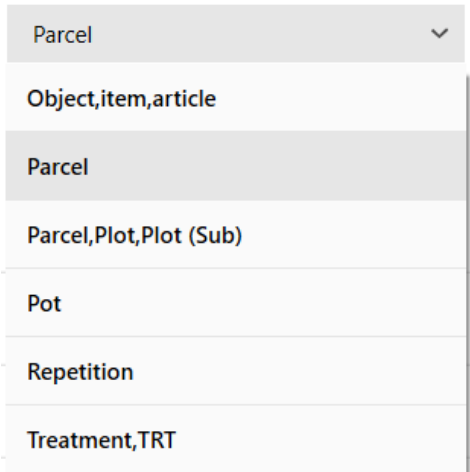
### 3 Customizing names and value ranges

During data collection, one or more trait is evaluated for a particular trial unit (plot, plant, pot, or any surveyed object).

smatrix is agnostic about what type of trial unit you are surveying. During the configuration process, you select from a drop-down menu the desired name. This name combined with the identifiers

embedded in the sheet identifies the trial units. Therefore, the list of available ID column names can be configured.

**Column A** - Name



A trait can be the presence of a disease or insects, the effect of a treatment, a characteristic of a plant or many other things. The actual trait depends on the line of business of the smatrix user. Therefore, the list of Trait names can be customized.

A trait evaluation results in a count, a percentage, a code, or a text. Therefore, also the available value ranges are customizable.

As all these customizable items can be used in speech commands, their definitions are language dependent.

**Note:** Any change will impact all users on the same PC/tablet, as this configuration is shared.

### 3.1 ID column names

ID column names are stored in the headers\_<language>.xml file.

Each ID column name is represented by a <subject> element with two attributes:

- **id:** The value must fulfill the conditions described in [3.5](#)
- **name:** The value serves multiple purposes:
  - For display in the drop-down lists in the smatrix user interface
  - As pronunciation if no <pronounce> elements are defined, see [3.6](#)
  - It affects the autoconfiguration, see [4](#)

Example: <subject id="S2" name="Parcel"/>

All ID column names are grouped under a `<topic>` xml-element with the reserved id “`{SAMPLE}`”.

```
<topic id="{SAMPLE}" name="Sample">
    ... put all ID column name <subject> entries here
</topic>
```

### 3.2 Subsample group names

Subsample group names are stored in the headers\_<language>.xml file.

Similar as for ID column names, also subsample groups can be named using `<subject>` xml-elements with two attributes:

- **id**: The value must fulfill the conditions described in [3.5](#)
- **name**: The value is used for display in the drop-down lists in the smatrix user interface
  - Note: For counting trials the name needs to be pronounceable

```
Example: <subject id="SSGMYOWN1" name="My special subsample group"/>
```

They are grouped under a `<topic>` xml-element with the reserved id “`{SUBSAMPLEGROUP}`”.

```
<topic id="{SUBSAMPLEGROUP}" name="Sample groups">
    ... put all subsample group name <subject> entries here
</topic>
```

### 3.3 Trait names

Trait names are stored in the headers\_<language>.xml file.

Traits are represented by a `<subject>` xml-element with the following attributes:

- **id**: The value must fulfill the conditions described in [3.5](#)
- **name**: The value serves multiple purposes:
  - For display in the drop-down lists in the smatrix user interface
  - As pronunciation if no `<pronounce>` elements are defined, see [3.6](#)
  - It affects the autoconfiguration, see [4](#)
- **range**: (optional) Defines a default value range for this trait. If specified:
  - the value must equal an **id** of a value range in the values file, see [3.4](#)

- when selecting this trait in the configuration, the value range will be set automatically to its default value range. Note that the value range can be changed by the user afterwards.

Example: `<subject id="CHAR1" name="Characteristic 1" range="NUMBER_0_100"> </subject>`

As the list of traits can grow large, traits are grouped into different categories. The categories can be chosen freely, e.g., to group by trial types or customers. Each category is represented by a `<topic>` xml-element with the following attributes:

- **id**: The value must fulfill the conditions described in [3.5](#)
- **name**: The value is only used for display in the drop-down lists in the smatrix user interface

Example:

```
<topic id="GENERIC" name="Generic">
  <subject id="CHAR1" name="Characteristic 1" range="NUMBER_0_100"/>
  ... other subjects can be added ...
</topic>
```

Example: Defining two categories (using `<topic>`) with two traits each (using `<subject>`)

## <headers>

```
<topic id="GENERIC" name="Generic">
```

```
<subject id="CHAR1" name="Characteristic 1" range="NUMBER_0_100"/>
```

<subject id="CHAR2" name="Characteristic 2"/>

... other subjects can be added ...

&lt;/topic&gt;

```
<topic id="SPECIAL" name="Special">
```

```
<subject id="SPEC1" name="SZFG6">
```

<pronounce>Special 6</pronounce>

&lt;/subject&gt;

```
<subject id="SPEC2" name="SZFG7">
```

<pronounce>Special 7</pronounce>

&lt;/subject&gt;

... other subjects can be added ...

&lt;/topic&gt;

&lt;/headers&gt;

The trait configuration in smatrix populates the drop-down lists for ‘Category’ and ‘Trait’ using the values of the `name` attributes. In this example you could select from the category “Special” the trait name “SFG6”.

Column G

Category

Special

Trait

SZFG6

Category

Generic

Trait

SZFG7

### 3.4 Value Ranges

Value ranges are stored in the values\_<language>.xml file.

Each value range is represented by a <topic> xml-element with two attributes:

- **id**: The value must fulfill the conditions described in [3.5](#)
- **name**: The value is used for display in the drop-down lists in the smatrix user interface (trait configuration). It can be chosen freely but should be self-explanatory and unique. If chosen well, the user knows what he can use/speak without looking up details in the customization files.

Example:

```
<topic id="MyDataRange" name="0.0-100.0">
    ... here all values are defined ...
</topic>
```

The values supported by a value range are defined within the <topic>. They are represented by

- generators (<generator> xml-elements)
- subjects (<subject> xml-elements)
- or a mixture of both.

#### 3.4.1 Numeric value range

A numeric value range can be defined using a <generator> element. A generator element is like an instruction to generate values without having to define them one by one.

For a numeric value range the following attributes need to be defined for a generator:

- **type**: The value must be "NUMBER" to define a numeric value range
- **from**: The value defines the first number to be generated (always included)
- **to**: The value defines the last number to be generated (always included)
- **by**: (optional) The value specifies the step size of the generator.
  - The value can also have decimals after the point/comma (depending on your locale)
  - Defaults to "1" if not specified

Note: The optional attribute **decimals** was used in older smatrix versions, when **by** was still limited to natural numbers. If used, **decimals** (re)defines **by** when **by** is not defined or has no decimals after the point/comma.

i.e., **decimals="2"** together with **by="1"** is the same as **by="0.01"**

The numbers are generated by stepping from the **from**-value to the **to**-value, incrementing by the **by**-value (step size). The bordering **from**-/**to**-values will always be included.

Examples for number generators and the values they generate:

`<generator type="NUMBER" from="0" to="9" by="1"/>` → 0, 1, 2, 3, 4, 5, 6, 7, 8, 9

`<generator type="NUMBER" from="0" to="9" by="2"/>` → 0, 2, 4, 6, 8, 9

`<generator type="NUMBER" from="0" to="9" by="0.1"/>` → 0, 0.1, 0.2, ..., 8.8, 8.9, 9

`<generator type="NUMBER" from="0" to="9" by="0.8"/>` → 0, 0.8, 1.6, ..., 8, 8.8, 9

#### Notes:

- For number generators, smatrix always sends numeric values to Excel **except** if there is a prefix or suffix. Then a text string is sent to Excel.
- For details on Prefixes and Suffixes refer to section [3.4.5](#)

### 3.4.2 Text values

A value range does not have to be numeric. It can define text values (usually simple sets of words) instead. What needs to be defined, is

- the pronunciation of the text AND
- the data value, which is written into the Excel sheet

This is done using a `<subject>` element with the following attributes:

- **name**: The value is used for two purposes:
  - as pronunciation, if no `<pronounce>` statement is specified, see [3.6](#)
  - as data value, that is written to Excel (if no **id** is specified)
- **id**: (optional) If specified the value is written to Excel as the data value.

	Examples on how the definition of a text value is used	
<u>1</u>	<code>&lt;subject name="rodents"/&gt;</code>	"rodents" is pronunciation and data value
<u>2</u>	<code>&lt;subject name="rodents"&gt;</code> <code>    &lt;pronounce&gt;rats&lt;/pronounce&gt;</code> <code>    &lt;pronounce&gt;mice&lt;/pronounce&gt;</code> <code>&lt;/subject&gt;</code>	"rats" and "mice" are pronunciation, "rodents" is data value
<u>3</u>	<code>&lt;subject id="ROD1" name="rodents"/&gt;</code>	"rodents" is pronunciation, "ROD1" is data value
<u>4</u>	<code>&lt;subject id="ROD1" name="rodents"&gt;</code> <code>    &lt;pronounce&gt;rats&lt;/pronounce&gt;</code> <code>    &lt;pronounce&gt;mice&lt;/pronounce&gt;</code> <code>&lt;/subject&gt;</code>	"rats" and "mice" are pronunciation, "ROD1" is data value

### 3.4.3 Date value range

Next to numbers and text, also dates are commonly used. For example, when surveying the begin of flowering for a plant. The simplest definition of a date value range is the built-in DATE topic. It has no explicit values defined because it is built-in. It is identified by the id attribute value "DATE". The name attribute value can be freely chosen:

```
<topic id="DATE" name="Datum"/>
```

Valid pronunciations are described in the section "Speech commands" of the smatrix user manual. The data values written to Excel are DateTime objects, thus the display depends on the formatting used in the Excel sheet.

For most cases, the above definition is sufficient. However, there is also a more powerful way of specifying dates using a `<generator>` element with the attributes

- **type**: The value must be "DATE"
- **format**: defines how the date is written to Excel as the data value. Valid values are:
  - "Date": A date value (same as for the built-in topic above)
  - "DayOfYear": The day of year as numerical value
  - "d/MMM/yy": Formatted: 7/Jun/18
  - "d": Formatted: 07/06/2018
  - "dd.MM.yyyy": Formatted: 07.06.2018

**Important:** The pronunciations of dates are always the same, independent of the used format!

Example for a date value range, that writes the day of the year as numerical value to Excel

```
<topic id="MyDates1" name="Date as DayOfYear">
    <generator type="DATE" format="DayOfYear"/>
</topic>
```

Note: Prefixes and Suffixes can be added to all generator types - see section [3.4.5](#).

### 3.4.4 Code value range

The last type of `<generator>` is designed to enter a code as data value. There are only few special cases where this type is required. Due to the complexity of its definition, it is preferable to use the generators and text values described above, or a combination of those (see [3.4.6](#)).

Background:

- A code is a sequence of symbols: letters, numbers, and in very special cases other characters
- What we want to achieve is
  - to pronounce a sequence of words
  - where each word is each representing one of the code symbols
  - and enter the corresponding code into Excel
- An example would be to enter the code "PBT" we want to use the NATO alphabet and say "Papa Bravo Tango"

Code value ranges use a `<generator>` element with the attributes

- **type**: The value must be "CODE"
- **format**: defines a list of code patterns, separated by comma
- **action**: defines how the data value is written to excel. Can be
  - "word": code symbols are entered in Excel separated by space
  - "concatenate": code symbols are entered in Excel concatenated without space
- Several attributes where the name is a single **letter**: (optional) If defined, the letters in the format do not stand for themselves, but for a set of possible code symbols.

Let us first have a look at the specified **format**. It contains one or more patterns (separated by comma) defining how the code is made up. A pattern is a sequence of letters or symbols. A letter stands either for itself or represents a set of symbols. A symbol that is not a letter always represents only itself.

**Examples:**

<code>format="A"</code>	The code is made up of exactly one symbol <b>A</b>
<code>format="AA"</code>	The code is made up of exactly two symbols <b>A</b>
<code>format="A,AA"</code>	The code is made up of either one or two symbols <b>A</b>
<code>format="A,BA"</code>	The code is made up of either one symbol <b>A</b> , or two symbols where the first is <b>B</b> and the second one is <b>A</b> .

Without any further attributes, each letter stands for itself: **A** for the code symbol "A" and **B** for the code symbol "B".

When a letter represents a set of code symbols, that set is described by an attribute of that name. The value of the attribute can be

- A list of individual code symbols, separated by comma e.g. "A,B,C" or "g,h,i"
- A range of letters e.g., "A-C"
- A list of numbers e.g., "3,4,5,6"
- A range of numbers e.g., "3-5"
- A combination of numbers and number ranges "3-5,8,9"

**Examples:**

<code>format="A", A="A,E,I,O,U"</code>	The code is made up of exactly one symbol, which must be A,E,I,O or U
<code>format="AA" A="N,H,D,K,G,U,C,B,F,S"</code>	The code is made up of exactly two symbols, allowing any combination of two symbols in the set. e.g., "NN", "NH", "HN", "FS" ...
<code>format="A,AA,AAA" A="N,H,D,K,G,U,C,B,F,S"</code>	The code is made up of one, two or three symbols. It allows any combination of two symbols in the set.  valid: „N“, „KG“, „DSF“      not valid: „X“, „CBHD“
<code>format="AB" A="A-Z" B="1-5"</code>	The code is made up of exactly two symbols, first one is an uppercase letter, second one is a number between 1 and 5 e.g. "A4", "K2", "X5" ...

**Note:** If you want to include the **comma** symbol in a format or code set, you have to precede it with a backslash.

With this definition, we can define codes of a certain length and specify which code symbols are allowed at each position. Looking back at the example of the NATO alphabet, the last remaining thing to configure is the pronunciation for each code symbol.

By default, the default pronunciation is used for every code symbol. To allow other pronunciations, `<pronounce>` statements can be used similar as in the definition of trait names or ID column names. These pronunciations need to be specified with `<subject>` element representing a code symbol. The `<subject>` requires one attribute:

- **id**: The value of this attribute needs to be the code symbol

Example:

```
<generator type="CODE" format="AAA" A="A-Z" action="word">
  <subject id="A">                                → "A" is code symbol
    <pronounce>alfa</pronounce>                    → "alfa" and "a" are pronunciations
    <pronounce>a</pronounce>
  </subject>
  <subject id="B">                                → "A" is code symbol
    <pronounce>bravo</pronounce>                   → "bravo" and "b" are pronunciations
    <pronounce>b</pronounce>
  </subject>
  ... add definitions of other code symbols here ...
</generator>
```

Note: Prefixes and Suffixes can be added to all generator types - see section [3.4.5](#).

### 3.4.5 Generator Prefix/Suffix

Each `<generator>` element defines a range of supported values. Regardless of the generator type, it can be extended by a common prefix/suffix for all values, impacting

- the pronunciation of the values AND/OR
- their data value (written to Excel)

This is done by one `<prefix>` and/or one `<suffix>` element within the `<generator>`. The attributes and their impact on the pronunciation/data value is identical for prefix and suffix. The only difference is

that they are added before (<prefix>) or after (<suffix>) the values of the generator. Available attributes are:

- **id**: The value is used as part of the data value (if required by the **include** attribute)
- **name**: The value is used for two purposes:
  - as pronunciation if no <pronounce> element is specified, see [3.6](#)
  - as part of the data value if no **id** is specified and required by **include**
- **optional**: Defines if the prefix/suffix needs to be pronounced. Values can be
  - **"true"**: The prefix/suffix must be pronounced
  - **"false"**: The prefix/suffix does not have to be pronounced
- **include**: (optional) Defines if the prefix/suffix is part of the data value. Possible values are:
  - **"yes"**: Always adds prefix/suffix regardless if it was said
  - **"no"**: (default) Never adds prefix/suffix regardless if it was said
  - **"IfSaid"**: Adds prefix/suffix to the data value if it was said
- **action**: (optional) Defines how prefix/suffix are combined with the value. Possible values are:
  - **"concatenate"**: prefix/suffix are concatenated with the value without space
  - **"word"**: prefix/suffix are concatenated with the value, separated by space

For better understanding, we have a look at a number generator with values from 0 to 100. We want to extend it by a “percent” suffix.

```
<generator type="NUMBER" from="0" to="100" by="1">
```

... here the suffix is added ...

```
</generator>
```

Examples: For different definitions of the <suffix>, the impact of the used attributes on pronunciation (left column) and data value (right column) is shown for the value 80.

Used pronunciations	Used data values
<suffix name="percent" optional="true" include="no"/>	
"80"	80
"80 percent"	80
<suffix name="percent" optional="false" include="no"/>	
"80 percent"	80
"80" will not be recognized!	Nothing is written to Excel
<suffix id="%" name="percent" optional="true" include="yes" action="concatenate"/>	
"80"	"80%"
"80 percent"	"80%"
<suffix id="%" name="percent" optional="true" include="yes" action="word"/>	
"80"	"80 %"
"80 percent"	"80 %"
<suffix name="percent" optional="true" include="yes" action="word"/>	
"80"	"80 percent"
"80 percent"	"80 percent"
<suffix name="percent" optional="true" include="IfSaid" action="word"/>	
"80"	80
"80 percent"	"80 percent"

### 3.4.6 Combined value ranges

Value ranges do not need to be limited to one of the described generator types or text values. If desired, a combination can be used. Consider the following two examples for value ranges that are made up of text values and generators.

Example1: Definition of a value range for entering a percentage, but

- For values 0-10 we allow fine granularity (one decimal after the point)
- For values between 10 and 100 we only allow step-size 10
- We add a text “100” to add perfect and gorgeous as synonyms for the value 100.
- The word “percent” is an optional suffix for all numbers

The complete value range definition then looks like this:

```
<topic id=" MyDataRange " name="0.0-9.9, 10, 20, ... 100 % + Perfect">
  <generator type="NUMBER" from="0" to="10" by="1" decimals="1">
    <suffix name="percent" optional="true"/>
  </generator>
  <generator type="NUMBER" from="20" to="100" by="10">
    <suffix name="percent" optional="true"/>
  </generator>
  <subject name="100">
    <pronounce>perfect</pronounce>
    <pronounce>gorgeous</pronounce>
  </subject>
</topic>
```

Example 2:

In [3.4.3](#) we described a `<generator>` of type ‘DATE’, that stores the entered Date as a number using the format “DayOfYear”. The supported pronunciations however only allow dates like ‘Tomorrow’ or ‘4<sup>th</sup> of July’. If we want to allow the user to say either the date, or the day number directly, we can use two generators:

```
<topic id="MyDates" name="Date as DayOfYear">
  <generator type="DATE" format="DayOfYear"/>
  <generator type="NUMBER" from="1" to="366" by="1"/>
</topic>
```

### 3.5 id Attributes

Various elements have **id** attributes. The values for these must be chosen carefully and fulfill the following conditions:

- They must be unique within the file
- They must be different from the reserved ids "{SAMPLE}", "{SUBSAMPLEGROUP}", "{SUBSAMPLE}" and "{#SUBSAMPLES}"
- They must be kept unchanged! Once a configuration has been made based on a values/headers file, the ids are used to link sheet columns to these definitions. Changing them will make the configuration incorrect or unusable.
- They should be consistent between languages. If you want to be able to share the same configured sheet between operators speaking a different language, language specific header files should be consistent!

### 3.6 Pronunciations

Everything that can be used in speech commands needs to be pronounceable. That is the case for

- Traits (to navigate to the trait)
- ID columns (to navigate to a certain trial unit)
- values of value ranges (to enter a data value)

By default, the value of the **name** attribute is used as the pronunciation. Sometimes, the chosen names are abbreviations, acronyms, or codes (e.g., EPPO codes) that are not pronounceable. Or the names are very long, and you want to use a shorter version as pronunciation.

To overcome this issue, smatrix allows the definition of `<pronounce>` elements within the parent xml element. If one or more `<pronounce>` elements are added, **only those** will be valid pronunciations – the name attribute will not be used as pronunciation.

Examples: Pronunciations for a trait depending on the defined <code>&lt;pronounce&gt;</code> elements	
<code>&lt;subject id="SPEC1" name="PHYTX"&gt;&lt;/subject&gt;</code>	Uses "PHYTX" (bad!)
<code>&lt;subject id="SPEC1" name="PHYTX"&gt;     &lt;pronounce&gt;Phytotoxicity&lt;/pronounce&gt; &lt;/subject&gt;</code>	Uses "Phytotoxicity"
<code>&lt;subject id="SPEC1" name="PHYTX"&gt;     &lt;pronounce&gt;Phytotoxicity&lt;/pronounce&gt;     &lt;pronounce&gt;Phyto&lt;/pronounce&gt; &lt;/subject&gt;</code>	Allows "Phytotoxicity" or "Phyto"
<code>&lt;subject id="SPEC1" name="PHYTX"&gt;     &lt;pronounce&gt;p h y t x&lt;/pronounce&gt; &lt;/subject&gt;</code>	Supports spelling "p h y t x" (not recommended). To use spelling, you need assign a <code>&lt;pronounce&gt;</code> element, with the individual letters <u>in lowercase</u>

Note: Always the first pronunciation will be used by smatrix for feedback prompts.

#### 4 Customizing Autoconfiguration

When an Excel sheet was never configured before (no parameter file is found in the directory of the Excel file), smatrix will try to autoconfigure the sheet. To make this autoconfiguration more successful, the *headers.xml* file can be enhanced with additional information.

The content of the `name` attribute is used by smatrix for the autoconfiguration. You can use different aliases within the name to improve it. Have a look at the following description to understand the impact on the name value and how it matches Excel cells.

- Cell content matches the value (case insensitive)
  - "Treatment" will match "treatment" and "Treatment"
- A part of the value (split by comma) matches the cell content
  - "Treatment,TRT" matches cells "Treatment", "treatment", "TRT" and "trt"
  - The comma separated parts are aliases.
- Cell content **ends with** one of the aliases of the value
  - "TRT" matches also "009TRT"
- Spaces within an alias match any white space in the cell content
  - "A B" will match "AnewlineB", "A B", "AtabB", "A B", and many more

- Note: “A\nB” is an older notation that matches only exactly “AnewlineB”. It is better to use the new notation with a space instead.

Important: When extending the value of the name attribute, make sure to add a <pronounce> element to keep a simple pronunciation.

Example:

```
<subject id="S4" name="treatment"/>
```

Can be written as

```
<subject id="S4" name="Treatment,TRT">
```

```
  <pronounce>treatment</pronounce>
```

```
</subject>
```

Both definitions use ‘treatment’ as pronunciation. Hence, the user can navigate by saying e.g., ‘treatment 55’

The name contains several aliases (Treatment, TRT) separated by comma. This allows smatrix to easily detect Excel cells using one of those aliases. Autoconfiguration will map the column to the ID column name “Treatment,TRT”.

This logic applies to name attributes in the following definitions (all contained in the headers file):

- ID column names in a <subject> xml-Element within <topic> of id "{SAMPLE}"  

```
<subject id="IDCOL4" name="Parcel,Plot,Plot (Sub)"/>
```
- Trait names in a <subject> elements that represent trait names.  

```
<subject id="CHAR1" name="Characteristic 1"/>
```
- Subsample ID column in the <subject> xml elements within <topic> of id "{SUBSAMPLE}"  

```
<topic id="{SUBSAMPLE}" name="Subsample">
  <subject name="Subsample"/>
  <subject name="Sub"/>
</topic>
```

→ uses "Subsample" and "Sub" for autoconfiguration
- #subsamples header aliases in <subject> xml elements within <topic> of id "{#SUBSAMPLES}"  

```
<topic id="{#SUBSAMPLES}" name="#Subsamples">
  <subject name="# Subsamples"/>
  <subject name="Subsamples"/>
</topic>
```

#### 4.1 Autoconfiguration for #subsamples

When working with subsamples, it is sometimes necessary to have a different number of subsamples for some Trait. This information is stored in the sheet as a separate header line. Example:

	A	B	C	D	E
1	Subsamples	5	10	5	1
2					
3					
4		Characteristic 1	Characteristic 2	Characteristic 3	Characteristic 4
5	Parcel	0-100%	1,3,5,7,9	50-120cm	1,3,5,7,9
6	100 1				
7	100 2				
8	100 3				
9	100 4				

Header line for #subsamples

For smatrix to recognize this header line, the specific topic with id “{#SUBSAMPLES}” is used.

## 5 APPENDIX

### 5.1 smatrix configuration XML syntax

smatrix is configured with two files: a headers file and a values file. Please refer to [2](#) for the file locations.

Reminder: The default files in “C:\ProgramData\smatrix\templates” should **never** be modified because a re-installation of smatrix will overwrite them. Copy the file you want to change to “C:\ProgramData\smatrix\data” and modify it there.

#### The headers file

This file is used to configure the autoconfigure feature and to describe the various Trait to which columns can be mapped.

XML-element	attribute	Content/Values	Comment
<headers>		<topic>	
<topic>		<subject>	A topic represents a subject category.
	id	text	Id, unique in file. The following topics are used for the autoconfiguration feature: {SAMPLE}, {SUBSAMPLE}, {#SUBSAMPLES}, {SUBSAMPLEGROUP} The other topics define Trait categories.
	name	text	Name to be displayed in drop-down list except for reserved topics “{...}”
<subject>		<pronounce>	One or more pronunciations. Necessary if subject name is not pronounceable.

			If there are no <pronounce> elements, name will be used for pronunciation.
	id	<i>text</i>	Id, unique within the enclosing topic
	name	<i>text</i>	Name to be displayed in drop-down list except if in reserved topics "{...}" If a name constitutes a list of names separated by commas, all names are treated as aliases. This is used during autoconfiguration.
	range	<i>text</i>	Optional attribute. Id of a topic in the values file representing a value range. This value range will be the default range for this trait.

### The values file

This file is used to define value ranges where each <topic> represents a range.

Xml-element	attribute	Content/Values	Comment
<values>		<topic>	Each topic represents a value range
<topic>		<subject>, <generator>	A <subject> represent a single text value. A <generator> represents a collection of values generated by the system.
	id	<i>text</i>	Id, unique in file
	name	<i>text</i>	Name to be displayed in drop-down list.
<subject>		<pronounce>	One or more pronunciations. Necessary if subject name is not pronounceable.
	id	<i>text</i>	Id, unique in topic. Represents the value of this subject.
	name	<i>text</i>	Name to be displayed in drop-down list. Represents the value of this subject when id is undefined.
<generator>		<prefix>, <suffix>, <subject>	prefix and suffix are optional, only 1 prefix and/or 1 suffix per generator can be defined. Subjects inside a CODE generator are used to define pronunciations for codes.
	type	"DATE" or "NUMBER" or "CODE"	
	from	<i>number with optional decimals</i>	<b>For type NUMBER</b> , begin (inclusive) of the range
	to	<i>number with optional decimals</i>	<b>For type NUMBER</b> , end (inclusive) of the range
	by	<i>number with optional decimals</i>	<b>For type NUMBER</b> , increment

	decimals	<i>Integer number</i>	<b>For type NUMBER</b> , number of digits after the comma
	format	"Date" "DayOfYear" "d/MMM/yy" "d" "dd.MM.yyyy"	<b>For type DATE</b> , one of these values: A date value The day of year numerical value Formatted: 7/Jun/18 Formatted: 07/06/2018 Formatted: 07.06.2018
	format	Example: "A,AA,AAA"	<b>For type CODE</b> , a list of code patterns separated by ',' Each letter symbolizes either a set of code symbols or itself.
	A	Example: "1-3,7,9"	<b>For type CODE</b> , a list of letters, numbers or symbols that are part of the code set A. Either comma separated symbols or ranges x-y can be used.
	action	"word" or "concatenate"	<b>For type CODE</b> , is the code symbol included in the result as a separate word or concatenated to the other symbols?
<prefix>		<pronounce>	One or more pronunciations. Necessary if prefix name is not pronounceable.
	id	<i>text</i>	Prefix value
	name	<i>text</i>	Prefix value if no id present
	optional	"true" or "false"	Do you have to pronounce the prefix?
	include	"no", „yes" or "ifsaid"	Is the prefix value included in the result? No = never, Yes = always, IfSaid = only if the suffix was pronounced?
	action	"word" or "concatenate"	Is the prefix value included in the result as a separate word or concatenated to the generator value?
<suffix>		<pronounce>	One or more pronunciations. Necessary if suffix name is not pronounceable.
	id	<i>text</i>	Suffix value
	name	<i>text</i>	Suffix value if no id present
	optional	"true" or "false"	Do you have to pronounce the suffix?
	include	"no", „yes" or "ifsaid"	Is the suffix value included in the result? No = never, Yes = always, IfSaid = only if the suffix was pronounced?
	action	"word" or "concatenate"	Is the suffix value included in the result as a separate word or concatenated to the generator value?
<pronounce>		<i>text</i>	How to pronounce the containing subject, prefix, or suffix

## 5.2 Pitfalls

The customization presented in this document gives much freedom to the user. However, customization can also lead to some pitfalls, e.g., due to ambiguities.

A few of the possible pitfalls are:

- The values of `id` attributes are not unique, see [3.5](#)
  - That can happen when copying an existing trait/ID column/value range definition in the header/values files.
    - ✓ Please make sure to adapt the ID to a new unique value when creating a copy!
- The same pronunciation is allowed for e.g., two different traits
  - If you have two traits “Black rust” and “Yellow rust” and both allow for the pronunciation “rust”. If they would be used in the same trial, smatrix would randomly navigate to one or the other trait, as the pronunciations are valid for both.
    - ✓ Avoid using same pronounce elements across different traits/ID columns
- One name starts with the name of another e.g., trait
  - If you define two traits, “maturity” and “maturity date” it can lead to a similar problem in the recognition. If both traits are used in a trial and the user says, “maturity date”, but makes a bit of a pause between the two words. The recognizer can interpret this as a complete command and navigate to “maturity” instead.
    - ✓ Try to avoid this situation by using e.g., “maturity level” and “maturity date” resolving this overlap.
- Different data values for the same e.g., number
  - Consider the word “dozen” and the number “12”. If you want support numbers from 0 to 100, and allow to say “dozen” you can define:
 

```
<generator type="NUMBER" from="0" to="100" by="1"/>
<subject name="dozen"/>
```

 This would allow the correct pronunciation, but if you said “12” the number will be used as data value, if you said “dozen”, the word “dozen” is the data value.
    - ✓ To get consistent data values, make sure the data values are using numbers as well, and use a pronounce statement for the word:
 

```
<subject name="12">
    <pronounce>dozen</pronounce>
</subject>
```
- Phonetic ambiguities of different words (homophones)
  - Exemplarily you can think of the words “no” and “know”. Both words are pronounced the same way but mean something different. If both would be text subjects in a value range, the recognition is completely ambiguous.
    - ✓ Try avoiding this situation by using synonyms that are not ambiguous, e.g., “nay” instead of “no”, or “knowledge” instead of “know”.
- Phonetic confusability (synophones)

- Some words do not have the same, but very similar pronunciations. If not spoken clearly, they could be confused in the recognition
  - ✓ Words like „thing“ and „think“ sound very similar and could be confused. Try to avoid using such similar sounding words and use synonyms instead e.g., “item” instead of “thing” for this abstract example.
  - ✓ This can also be the case for number like „fourteen“ and „forty“, that are very close and could be confused. It is not possible to use synonyms for those in the configuration. Instead, the user can avoid issues by specifying the number digit-by-digit (“one four” for 14 and “four zero” for 40).

### 5.3 Counting Trial

Counting trials differ much from the other supported types of trial. With respect to the configuration there are two things that need to be customized carefully:

- The same Subsample group needs to be used for all Traits
- The used value ranges may not overlap!

**Subsample group:** If you want to define your own subsample group, refer to [3.2](#) on the creation. There you can also assign pronunciations, which are used for feedback.

**Value Ranges:** The used value ranges may not overlap, because they are recognizable in parallel! Recognition results would be ambiguous leading to unpredictable behavior. As an example, if the value “5” would be part of two value ranges, it would not be clear if the value which class-count should be incremented!

If the value ranges do not overlap, also combined value ranges can be used in counting trial. Consider the example below, where next to numbers also class names are defined.

Example: We define two value ranges for two classes “0-5” and “greater 5”, also allowing the class names as pronunciations. With the following definition,

- The count for class X is incremented when the user says a number between 0-5 or “class x”
- The count for class X is incremented when the user says: a number between 6-100 or “class y”

```
<topic id="COUNT_X" name="Class X, 0-5">
  <generator type="NUMBER" from="0" to="5"/>
  <subject name="class x"/>
</topic>
<topic id="COUNT_Y" name="Class Y, greater 5">
  <generator type="NUMBER" from="6" to="100"/>
  <subject name="class y">
    <pronounce>more than 5</pronounce>
    <pronounce>class y</pronounce>
  </subject>
</topic>
```

Note: The user manual shows an example with four different classes. Possible value range definitions for those can be seen in the default values.xml file in  
 C:\ProgramData\smatrix\templates\values\_en.xml