MIX manual

<div class="MsoNormal"><span lang=EN-US style='mso-bidi-font-family:Arial'>Mix is a utility designed to randomise experimental item files according to constraints and rules written down in a script. Mix is, for example, able to set a maximum to the number of repetitions of identical items, or make sure items are a specified minimum distance apart. Mix also has a number of commands for generating blocked output or performing simple formatting.</span></div>

<div class="MsoNormal"></div>

<span lang=EN-US style='mso-bidi-font-family:Arial'>Mix can be called from the Windows command line like this:</span>

<div class="MsoNormal"></div>

<span lang=EN-US><span style='mso-tab-count:1'></span>**Mix** <script file></span> [arg] [arg] . . .

<div class="MsoNormal"></div>

<span lang=EN-US style='mso-bidi-font-family:Arial'><div class="MsoNormal"></div>Where <span lang=EN-US style='mso-bidi-font-family:Arial'>*<script file></span><span lang=EN-US style='mso-bidi-font-family:Arial'>* is the name of a file containing the commands you want Mix to execute.</span> This <div class="MsoNormal"></div><span lang=EN-US style='mso-bidi-font-family:Arial'>file can be created with any text editor that can handle plain text files (\**.txt*).</span> The optional additional arguments can be used to parameterise the script: each occurrence of ‘%1’ in the script will be replaced with the first additional argument, ‘%2’ with the second etc. This will enable you to use the same script with different settings or input.

Mix can also be started by simply double-clicking on the program. It will then present an open-file dialog, prompting the user to select the script file. In this way you do not have to work with a command-line interface.

<span lang=EN-US>Creating an item file</span>

<div class="MsoNormal"><span lang=EN-US style='mso-bidi-font-family:Arial'>Before you can start mixing, you will have to create an item file. Each line of the item file should contain one item, with all attributes belonging to the item on that same line</span><span lang=EN-US style='mso-bidi-font-family:Arial'><span style="mso-bidi-font-family: Arial">.</span></span></div> You will have to tell Mix which item file, or files, you are using by including a line in your script like this:

**ItemFile** C:\Experiment1\Items.txt

An additional number can be added before the filename(s), like this:

**ItemFile** 5 Items1.txt Items2.txt

This will now read in the content of these files 5 times.

<span lang=EN-US style='mso-bidi-font-family:Arial'>A simple item file containing 4 items could look like this:</span>

<span lang=EN-US><span style='mso-tab-count:1'></span>kiwi</span>

<span lang=EN-US><span style='mso-tab-count:1'></span>lemon</span>

<span lang=EN-US><span style='mso-tab-count:1'></span>strawberry</span>

<span lang=EN-US><span style='mso-tab-count:1'></span>banana</span>

<span lang=EN-US style='mso-bidi-font-family:Arial'>A more complicated version, containing four items with two attributes each, could be:

</span>

<span lang=EN-US><span style='mso-tab-count:1'></span>kiwi green sweet</span>

<span lang=EN-US><span style='mso-tab-count:1'></span>lemon yellow sour</span>

<span lang=EN-US><span style='mso-tab-count:1'></span>strawberry red sour</span>

<span lang=EN-US><span style='mso-tab-count:1'></span>banana yellow sweet</span>   
  <div class="MsoNormal"></div>

<span lang=EN-US style='mso-bidi-font-family:Arial'>As a general rule, Mix will interpret each line of an item file like this:</span>   
  <div class="MsoNormal"></div>

<span lang=EN-US>Field1 Field2 Field3 . . .</span>  
  <div class="example"></div>

<span lang=EN-US>Fields should be separated by spaces or tabs. If you want to use a different delimiting character, use the command **InputDelimiter**, explained further down.</span>

<span lang=EN-US>Creating a script file

</span>

M<div class="MsoNormal"><span lang=EN-US>MMost Mix commands are put in a script file. Only a few commands are used in the item file. They will be explained further on in this manual.</span></div> <span lang=EN-US>To create a script file, use any text-editor that you are comfortable with, like NotePad.</span>

The script file contains the complete ‘recipe’ for a randomisation job and provides a way of documenting the process by adding comments. Any text after a double slash, ‘//’, will be considered a comment by the program.  
  <div class="MsoNormal"></div>

<span lang=EN-US>IMPORTANT: Mix command-names contain capitals and since Mix is case sensitive please make sure you copy these correctly. This rule also applies when you use capitals in the names you create yourself, like properties and constraints.

Filenames and directory names with spaces can be used, but they will have to be enclosed in double quotes.</span>

<span lang=EN-US>Elementary Mixing</span>

Simple randomising

</span>

<div class="MsoNormal"><span lang=EN-US style='mso-bidi-font-family:Arial'>The simplest action you can perform with Mix is to shuffle the input without any constraints at all.</span></div> <span lang=EN-US style='mso-bidi-font-family:Arial'>To do this, just create a script file containing a simple definition of an output file, using the **OutputFile** command:</span>   
  <div class="MsoNormal"></div>

**ItemFile** MyItems.txt

<span lang=EN-US><span style='mso-tab-count:1'></span>**OutputFile** output.txt</span>   
  <div class="example" style="tab-stops:14.2pt"></div>

<span lang=EN-US style='mso-bidi-font-family:Arial'>Mix will now produce a file called *output.txt* in the current directory or folder. You can also specify a different directory by using a full path:</span>   
  <div class="MsoNormal"></div>

<span lang=EN-US><span style='mso-tab-count:1'></span>**OutputFile** d:\experiment1\output.txt</span>   
  <div class="MsoNormal"></div>

<span lang=EN-US style='mso-bidi-font-family:Arial'>This file will now contain the lines of your item file in a random order.

If you want several randomisations of your items, just use a script file containing:</span>   
  <div class="example" style="tab-stops:14.2pt"></div>

<span lang=EN-US><span style='mso-tab-count:1'></span>**OutputFile** out1.txt</span>

<span lang=EN-US><span style='mso-tab-count:1'></span>**OutputFile** out2.txt</span>

<span lang=EN-US><span style='mso-tab-count:1'></span>**OutputFile** out3.txt</span>   
  <div class="MsoNormal"></div>

<span lang=EN-US style='mso-bidi-font-family:Arial'>and Mix will generate 3 differently randomised versions of your item file.</span>

<span lang=EN-US>Constraining the randomisation process

</span>

<div class="MsoNormal"><span lang=EN-US style='mso-bidi-font-family:Arial'>If you want to randomise your items according to specific constraints, you will have to tell Mix what properties it should look for and what the constraints on those properties are.</span></div>   
  <div class="MsoNormal"></div>

<span lang=EN-US style='mso-bidi-font-family:Arial'>First you have to label your input with the **Property** command. You use the command in your script file, like this:</span>   
  <div class="MsoNormal"></div>

<span lang=EN-US><span style='mso-tab-count:1'></span>**Property** colour 2</span>   
  <div class="MsoNormal"></div>

<span lang=EN-US style='mso-bidi-font-family:Arial'><span style="mso-bidi-font-family: Arial">Mix now knows</span> that the information on the second field of each item is called *colour*.</span>

<div class="MsoNormal"></div><span lang=EN-US style='mso-bidi-font-family:Arial'>Your item file could, for example, look like this:</span>   
  <div class="example"></div>

<span lang=EN-US><span style='mso-tab-count:1'></span>kiwi green</span>

<span lang=EN-US><span style='mso-tab-count:1'></span>lemon yellow</span>

<span lang=EN-US><span style='mso-tab-count:1'></span>strawberry red</span>

<span lang=EN-US><span style='mso-tab-count:1'></span>banana yellow</span>   
  <div class="MsoNormal"></div>

<span lang=EN-US style='mso-bidi-font-family:Arial'>The property *colour* for the first item has the value *green*, since this is the value of the second field.</span> <div class="MsoNormal"></div><span lang=EN-US style='mso-bidi-font-family:Arial'>In this example the colour value ‘*yellow’* appears twice. </span><span lang=EN-US style='mso-bidi-font-family:Arial'>Suppose you want to put a constraint on the property *colour.* For example, you do not want output in which two adjacent lines have the same *colour* value. Use the commands **Constraint** and **MaxRep**, like this in your script file:</span>   
  <div class="MsoNormal"></div>

<span lang=EN-US><span style='mso-tab-count:1'></span>**Property** colour 2</span>

<span lang=EN-US><span style='mso-tab-count:1'></span>**Constraint** colour **MaxRep** 1</span>   
  <div class="MsoNormal"></div>

<span lang=EN-US style='mso-bidi-font-family:Arial'>This means that the property *colour* cannot have more than 1 repetition of the same value.</span>   
<div class="MsoNormal"></div><span lang=EN-US style='mso-bidi-font-family:Arial'>You could also have used the commands **Constraint** and **MinDist**:</span>   
  <div class="example" style="tab-stops:14.2pt"></div>

<span lang=EN-US><span style='mso-tab-count:1'></span>**Constraint** colour **MinDist** 2</span>   
  <div class="MsoNormal"></div>

<span lang=EN-US style='mso-bidi-font-family:Arial'>This is a different constraint type, which will ensure that there will be a minimum distance of 2 between identical values.</span> B<div class="MsoNormal"></div><span lang=EN-US style='mso-bidi-font-family:Arial'>BBoth constraints will have the same effect, because excluding repetitions is the same as having a minimum distance of 2. Of course you can use other values, for example to set the number of maximum repetition to more than 1.</span>

Other constraints

There are more constraints than just maximum repetition and minimum distance. Some of these can make the mixing process rather slow, especially when used in combination. Too many constraints can also make the sequence of items too predictable. Mix will automatically detect this when it happens, and produce an additional output file with information about the sequences that are predictable.

Maximum distance constraint

This one behaves just like the minimum distance constraint, but keeps items at a maximum distance instead. Distance is defined as the actual difference between the positions of two items, so adjacent items are at a distance of precisely 1. Use this constraint like this:

**Constraint** status **MaxDist** 10

Order constraint

This one will constrain the order in which certain items can appear. An Example:

**Constraint** status **Order** A B

This will make sure that for the property 'status' all values 'B' will appear after the last appearance of an 'A'.

Numeric Constraints

This constraint can be used on numeric fields. If you only want to prevent identical numerical fields from repeating or getting to close to each other, you can use the constraints described above, since identical numerical values also have identical string values. But if you want a certain minimum difference for numerical fields between two adjacent items you will have to use one of these numerical constraints. They are used like this:

**Constraint** frequency **MinNumDiff** 50

or

**Constraint** frequency **MaxNumDiff** 50

depending on if you want a minimum, or a maximum numerical difference. This constraint only works on adjacent items.

Pattern constraint

With this constraint you can prevent repetitions of patterns in the output. For example: if you set a max repetition constraint to a low value you might end up with a very predictable sequence, like ABABABAB etc.

Mix always performs a check on all properties defined by the user to detect predictable patterns. If such patterns are detected, they are written to a file for the user to inspect. Setting a constraint will normally introduce some level of predictability, but sometimes there is more predictability than expected, or a combination of constraints has an unexpected effect. In those cases a pattern constraint could be added.

This constraint is used like this:

**Constraint** status **Pattern** 3 2

This means that Mix will check the order of the 'status' property. It will only allow a maximum of 3 occurrences of the same pattern, using a window size of 2. So, the above sequence ABABABAB will not appear anymore, since it contains more than 3 occurrences of the sequence AB. An additional parameter can be added to the pattern constraint, specifying the maximum number of patterns that reach the maximum number of occurrences.

**Constraint** status **Pattern** 3 2 5

In this case only 5 patterns are allowed to occur 3 times, and all the other ones can only occur 2 times maximally. This adds the possibilities to have a more graded change in criterion than just lowering the maximum number of occurrences by 1.

You can also add specific maximum numbers of occurrence for patterns, with the 'Max' keyword, like this:

**Constraint** status **Pattern** 3 2 **Max** A\_B 2

Now the pattern "AB" can only occur twice, instead of 3 times. The '\_' character has to be inserted in between every part of the pattern.

If you want to ensure no pattern occurs more that it should, based on the probabilities of occurrence of its part, you can use:

**Constraint** status **Pattern** **Flat** 2

The extra number specifies the window size to use.

**Sequence constraint**

This constraint will make sure that items will always appear in a certain sequence in the output.

**Constraint** status **Sequence** A B C

Items with the property ‘status’ will now appear in triplets with the values for ‘status’ being ‘A’, ‘B’ and ‘C’. No items with a ‘status’ value that is present in the sequence can appear outside of a correct sequence anymore. The minimum length of a sequence is obviously 2. There’s no maximum.

**Context constraint**

This is somewhat like the sequence constraint. It is specified like this:

**Constraint** status **Context** X A-[B]-X-C-D

Any items with a ‘status’ property of ‘X’ will now have to appear in the given context. That is, the two following items will need to have a status value of ‘C’ and ‘D’. When an item is in square brackets, like ‘[B]’, it is optional. This means that the item ‘X’ needs to be preceded with either an ‘A’ or an ‘A’ and a ‘B’.

The difference with the sequence constraint is that the items in a Context constraint can also appear elsewhere, out of context. Only the specified target item, here ‘X’, has to appear in context. This can be handy, for example to make sure that a break is always followed by a filler, without preventing fillers from also appearing elsewhere.

The Context constraint can also be used in a more flexible way, like this:

**Constraint** status **Context** X A-[B]-X-C,D,E

Now the value 'X' will have to be followed by either a 'C', a 'D' or an 'E'.

**Unique Constraint**

This will make sure a range of items will be unique for the given property.

**Constraint** status **Unique** 12 50

In this case the items from line 12 up to, but not including, 50 will all have unique values for the property ‘status’.

Conditional constraints

All constraints, except the Pattern constraint, can be made conditional upon a value of another constraint. For example, if you want to constrain the order of A and B trials, but only when the 'word' values are identical, you can do this:

**Constraint** status **Order** A B **ConditionalUpon** word

This will make sure that the Order constraint only activates when the two lines contain the same word.

Additionally, and extra argument can be given, like this

**Constraint** status **Order** A B **ConditionalUpon** word bicycle

In this case the Order constraint will only activate when the two items both contain the word property and it has the value ‘bicycle’. This will enable you to switch constraints on or off depending on specific properties.

You can also use a conditional constraint in the opposite way: such that the main constraint will only be triggered when the items are different on the conditional constraint. In this case you have to use the keyword **ConditionalUponNegative**. Everything else is the same.

**Constraints between two different properties**

It is even possible to specify a constraint between values of two different properties. To do this, use the **OtherProperty** keyword, like this:

**Constraint** status **MinDist 3** **OtherProperty type**

There will now be a minimal distance of 3 items between items with the same values for the 'status' and the 'type' property.

<span lang=EN-US>Making LineTypes

</span>

<div class="MsoNormal"><span lang=EN-US style='mso-bidi-font-family:Arial'>Suppose you want to randomise these items:</span></div>   
  <div class="MsoNormal"></div>

<span lang=EN-US><span style='mso-tab-count:1'></span>fruit apple green</span>

<span lang=EN-US><span style='mso-tab-count:1'></span>fruit banana yellow</span>

<span lang=EN-US><span style='mso-tab-count:1'></span>fruit strawberry red</span>

<span lang=EN-US><span style='mso-tab-count:1'></span>candy yellow</span>

<span lang=EN-US><span style='mso-tab-count:1'></span>candy red</span>

<span lang=EN-US><span style='mso-tab-count:1'></span>candy green</span>   
  <div class="MsoNormal"></div>

<span lang=EN-US style='mso-bidi-font-family:Arial'>You still want to make sure that two adjacent lines do not have the same colour, but the problem here is that the colour values are in different positions for *fruit* and *candy*.</span>   
  <div class="MsoNormal"></div>

<span lang=EN-US style='mso-bidi-font-family:Arial'>You can solve this by creating two different types of lines, using the command **LineType,** and defining the property colour separately for each of them:</span>   
  <div class="MsoNormal"></div>

<span lang=EN-US><span style='mso-tab-count:1'></span>// Define linetypes:</span>

<span lang=EN-US><span style='mso-tab-count:1'></span>**LineType** fruitline 1 fruit <span style='mso-tab-count:1'></span>// 'fruit' on position 1</span>

<span lang=EN-US><span style='mso-tab-count:1'></span>**LineType** candyline 1 candy <span style='mso-tab-count:1'></span>// 'candy' on position 1</span>

// Define properties  <div class="example"></div>

<span lang=EN-US><span style='mso-tab-count:1'></span>**Property** fruitline colour 3 <span style='mso-tab-count:1'></span>// Define ‘colour’ for fruit</span>

<span lang=EN-US><span style='mso-tab-count:1'></span>**Property** candyline colour 2 <span style='mso-tab-count:1'></span>// Define ‘colour’ for candy</span>   
  <div class="example"></div>

<span lang=EN-US><span style='mso-tab-count:1'></span>// Put a constraint on colour:</span>

<span lang=EN-US><span style='mso-tab-count:1'></span>**Constraint** colour **MaxRep** 1</span>   
  <div class="example" style="tab-stops:14.2pt"></div>

<span lang=EN-US>Mix will now recognise two different LineTypes, called ‘*fruitline’* and ‘*candyline’*, with the attribute *colour* on the third and second field, respectively. This will allow you to define a constraint for the property *colour* operating on all items, even though *colour* has a different location depending on the LineType of the item.</span>  
<div class="MsoNormal"></div><span lang=EN-US style='mso-bidi-font-family:Arial'  <div class="MsoNormal"></div

<span lang=EN-US style='mso-bidi-font-family:Arial'>A property that is not defined for a specific LineType is considered to be a *default property* and will be used on all items, except on the ones which have a specific LineType property.</span> If you do not specify a *default property* all lines without a LineType or with a LineType without a specifically defined property will *not* be assigned the property.  
  <div class="MsoNormal"></div>

<span lang=EN-US style='mso-bidi-font-family:Arial'>So the following script will generate the same results as the one above:</span>   
  <div class="MsoNormal"></div>

<span lang=EN-US><span style='mso-tab-count:1'></span>// Define linetype:</span>

<span lang=EN-US><span style='mso-tab-count:1'></span>**LineType** candyline 1 candy</span>

<div class="example"></div>

<span lang=EN-US>// Define default property colour on field 3</span>

<span lang=EN-US><span style='mso-tab-count:1'></span>**Property** colour 3</span>

<div class="example"></div>

<span lang=EN-US>// Define property colour on field 2 <div class="example"><span lang=EN-US>of candylines</span></div>

<span lang=EN-US><span style='mso-tab-count:1'></span>**Property** candyline colour 2</span>

<span lang=EN-US><span style='mso-tab-count:1'></span>// Put a constraint on colour:</span>

<span lang=EN-US><span style='mso-tab-count:1'></span>**Constraint** colour **MaxRep** 1</span>   
  <div class="example"></div>

LineTypes have to be defined in such a way that each line can only be of a single linetype. If you specify things so that one line could be more than one linetype, Mix will produce an error.

A Property can be defined for several Linetypes in one go in this way:

**Property** candyline,fruitline colour 2</span>

<span lang=EN-US>Mixing Blocks</span>

The ability to mix your items in blocks is crucial for most purposes. You might need to treat a number of items as practice or have one type of items in the first half of the experiment, and another type in the second half. Blocks will also allow you to control counterbalancing your conditions.

Defining blocks of items

</span>

<div class="MsoNormal"><span lang=EN-US style='mso-bidi-font-family:Arial'>You can group items in an item file in blocks, using the **Block** command:</span></div>   
  <div class="MsoNormal"></div>

<span lang=EN-US>

<span lang=EN-US><span style='mso-tab-count:1'></span>**Block** part1</span>

<span lang=EN-US><span style='mso-tab-count:1'></span>show green</span>

<span lang=EN-US><span style='mso-tab-count:1'></span>show red</span>

<span lang=EN-US><span style='mso-tab-count:1'></span>show blue</span>

<span lang=EN-US><span style='mso-tab-count:1'></span>**Block** part2</span>

<span lang=EN-US><span style='mso-tab-count:1'></span>show black</span>

<span lang=EN-US><span style='mso-tab-count:1'></span>show white</span>

<span lang=EN-US><span style='mso-tab-count:1'></span>show purple</span>   
  <div class="MsoNormal"></div>

<span lang=EN-US style='mso-bidi-font-family:Arial'>Note that you have to incorporate these commands in the **item file**! Evidently, they will not be copied to the output files.</span> <div class="MsoNormal"></div><span lang=EN-US style='mso-bidi-font-family:Arial'>In your script file, you can now use these blocks to control the output to different output files:</span>   
  <div class="MsoNormal"></div>

<span lang=EN-US><span style='mso-tab-count:1'></span>**OutputFile** out1.txt</span>

<span lang=EN-US><span style='mso-tab-count:1'></span>**Block</span><span lang=EN-US>** part1 **rand** **Add** 1</span>

<span lang=EN-US><span style='mso-tab-count:1'></span>**Block** part2 **rand** **Add** 2</span>

<span lang=EN-US><span style='mso-tab-count:1'></span>**OutputFile** out2.txt</span>

<span lang=EN-US><span style='mso-tab-count:1'></span>**Block** part2 **Add** a</span>

<span lang=EN-US><span style='mso-tab-count:1'></span>**Block** part1 **Add** b</span>   
  <div class="MsoNormal"></div>

<span lang=EN-US style='mso-bidi-font-family:Arial'>The file *out1.txt* will contain all items from the input file, with the items of the block *part1* before the block *part2*. In the file *out2.txt* the items of the block *part2* will precede those of the block *part1*.</span>   
  <div class="example" style="tab-stops:36.0pt"></div>

<span lang=EN-US style='mso-bidi-font-family:Arial'>If the Block command is followed by the addition **rand**, the items of that block will be randomised according to the constraints in the script file. In the above example the items of the blocks in the file *out1.txt* will be randomised, while the items in *output2.txt* will stay in exactly the same order as they were in the item file.</span>

You can also use the statement **norand** if you want the items in a block to stay in the order they are in.   
  <div class="MsoNormal"></div>

The randomisation statements **rand** and **norand** can be used either with the **Block** command (item file) or the **OutputFile** command (script file).   
  <div class="MsoNormal"></div>

Item file:   
  <div class="MsoNormal"></div>

**Block** one **rand**

Item1

Item2

**Block** two **norand**

Item3

Item4   
  <div class="MsoNormal"></div>

Script file:   
  <div class="MsoNormal"></div>

**OutputFile** exp1.txt

**Block** one

**Block** two **rand** //Error!!   
  <div class="MsoNormal"></div>

Be careful not to include randomisation statements in both files that will contradict each other.  If the randomisation statement is missing in one of the two files, Mix will use the value in the other file. If both files lack a randomisation statement, Mix will assume that no randomisation is specified (**norand** is the default value).

The given sequence of Blocks can be repeated several times by using the **RepeatBlocks** keyword in the OutputFile definition.

<span lang=EN-US style='mso-bidi-font-family:Arial'><span style="mso-bidi-font-family: Arial">Adding extra fields

</span></span>

<div class="MsoNormal"><span lang=EN-US style='mso-bidi-font-family:Arial'>The command **Add** will add an extra field to the end of every line in a block. So, the file *out2.txt* will look like this:</span></div>   
  <div class="MsoNormal"></div>

<span lang=EN-US><span style='mso-tab-count:1'></span>show black a</span>

<span lang=EN-US><span style='mso-tab-count:1'></span>show white a</span>

<span lang=EN-US><span style='mso-tab-count:1'></span>show purple a</span>

<span lang=EN-US><span style='mso-tab-count:1'></span>show green b</span>

<span lang=EN-US><span style='mso-tab-count:1'></span>show red b</span>

<span lang=EN-US><span style='mso-tab-count:1'></span>show blue b</span>   
  <div class="MsoNormal"></div>

<span lang=EN-US>As you can see, the order of the items in the blocks is unchanged and an '*a'* or '*b'* was added to the output.</span>

<span lang=EN-US>Randomising over blocks

</span>

<div class="MsoNormal"><span lang=EN-US style='mso-bidi-font-family:Arial'>As a rule Mix only checks constraints within blocks. Thus, Mix can produce output where</span><span style='mso-char-type: symbol; mso-symbol-font-family: Symbol'>—</span><span lang=EN-US style='mso-bidi-font-family:Arial'>viewed over block boundaries</span><span style='mso-char-type: symbol; mso-symbol-font-family: Symbol'>—</span><span lang=EN-US style='mso-bidi-font-family:Arial'>your constraints will be violated. For example, if your item file looked like this:</span></div>   
  <div class="MsoNormal"></div>

<span lang=EN-US><span style='mso-tab-count:1'></span>**Block** Fruit</span>

<span lang=EN-US><span style='mso-tab-count:1'></span>Apple</span>

<span lang=EN-US><span style='mso-tab-count:1'></span>Orange</span>

<span lang=EN-US><span style='mso-tab-count:1'></span>Pear</span>

<span lang=EN-US><span style='mso-tab-count:1'></span>Apricot</span>

<span lang=EN-US><span style='mso-tab-count:1'></span>**Block** Vegetable</span>

<span lang=EN-US><span style='mso-tab-count:1'></span>Spinach</span>

<span lang=EN-US><span style='mso-tab-count:1'></span>Parsnip</span>

<span lang=EN-US><span style='mso-tab-count:1'></span>Cabbage</span>

<span lang=EN-US><span style='mso-tab-count:1'></span>Cauliflower</span>   
  <div class="MsoNormal"></div>

<span lang=EN-US style='mso-bidi-font-family:Arial'>and you did not want two items starting with the same letter to be in a row, the script:</span>   
  <div class="example" style="tab-stops:36.0pt"></div>  <span lang=EN-US>

// Only use the first character</span>

<span lang=EN-US><span style='mso-tab-count:1'></span>**Property** FirstCharOfName 1 **Sub** 1</span>   
  <div class="example"></div>

<span lang=EN-US><span style='mso-tab-count:1'></span>**Constraint** FirstCharOfName **MaxRep** 1</span>

<span lang=EN-US><span style='mso-tab-count:1'></span>**OutputFile** out1.txt</span>

<span lang=EN-US><span style='mso-tab-count:1'></span>**Block</span><span lang=EN-US>** fruit **rand**</span>

<span lang=EN-US>**Block** vegetable **rand**</span>   
  <div class="MsoNormal"></div>

<span lang=EN-US>could produce an output file like this:</span>   
  <div class="MsoNormal"></div>

<span lang=EN-US><span style='mso-tab-count:1'></span>Apricot</span>

<span lang=EN-US><span style='mso-tab-count:1'></span>Orange</span>

<span lang=EN-US><span style='mso-tab-count:1'></span>Apple</span>

<span lang=EN-US><span style='mso-tab-count:1'></span>**P**ear</span>

<span lang=EN-US><span style='mso-tab-count:1'></span>**P**arsnip</span>

<span lang=EN-US><span style='mso-tab-count:1'></span>Cabbage</span>

<span lang=EN-US><span style='mso-tab-count:1'></span>Spinach</span>

<span lang=EN-US><span style='mso-tab-count:1'></span>Cauliflower</span>   
  <div class="example"></div>

<span lang=EN-US style='mso-bidi-font-family:Arial'>where your constraint is violated.</span>   
  <div class="MsoNormal"></div>

<span lang=EN-US style='mso-bidi-font-family:Arial'>If you want Mix to have a global view on constraints</span><span style='mso-char-type: symbol; mso-symbol-font-family: Symbol'>, </span><span lang=EN-US style='mso-bidi-font-family:Arial'>ignoring the fact that items can be in different blocks</span><span style='mso-char-type: symbol; mso-symbol-font-family: Symbol'>, </span><span lang=EN-US style='mso-bidi-font-family:Arial'>use the command **OverBlocks** in the script file. It is important to realise that Mix will only check constraints within the range of each block without this command!</span>

<span lang=EN-US>Properties on parts of fields

</span>

<div class="MsoNormal"><span lang=EN-US style='mso-bidi-font-family:Arial'>Sometimes you want to randomise using the first character of a field only, or the first two or three characters, or maybe the last two or three.</span></div> <div class="MsoNormal"></div><span lang=EN-US style='mso-bidi-font-family:Arial'>You can do so by using the **Property** command together with the **Sub** command while defining a property.</span>   
  <div class="MsoNormal"></div>

<span lang=EN-US style='mso-bidi-font-family:Arial'>Assume your items look like this:</span>   
  <div class="example" style="tab-stops:14.2pt"></div>

<span lang=EN-US><span style='mso-tab-count:1'></span>Bach</span>

<span lang=EN-US><span style='mso-tab-count:1'></span>Beethoven</span>

<span lang=EN-US><span style='mso-tab-count:1'></span>Mozart</span>

<span lang=EN-US><span style='mso-tab-count:1'></span>Monteverdi</span>

<span lang=EN-US><span style='mso-tab-count:1'></span>Verdi</span>   
  <div class="MsoNormal"></div>

<span lang=EN-US style='mso-bidi-font-family:Arial'>and for some reason you do not want adjacent items starting with the same character.</span>   
  <div class="MsoNormal"></div>

<span lang=EN-US style='mso-bidi-font-family:Arial'>Your script will have to look like this:</span>

<span lang=EN-US></span>

<span lang=EN-US>// Only use first character</span>

<span lang=EN-US><span style='mso-tab-count:1'></span>**Property** FirstCharOfName 1 **Sub** 1</span> <div class="example"></div>

<span lang=EN-US><span style='mso-tab-count:1'></span>**Constraint** FirstCharOfName **MaxRep** 1</span>   
  <div class="example" style="tab-stops:14.2pt"></div>

<span lang=EN-US style='mso-bidi-font-family:Arial'>Now Bach and Beethoven will not be next to each other.</span>

<div class="MsoNormal"></div><span lang=EN-US>In the above example, the **Property** command creates the variable *FirstCharOfName* for the item on the first field (hence the number *1*) of a line, while the **Sub** command specifies that this variable only concerns the first character of this field (hence the second number *1*).</span>   
  <div class="example" style="tab-stops:14.2pt"></div>

<span lang=EN-US style='mso-bidi-font-family:Arial'>With respect to counting characters in a field, Mix uses the following format:</span>   
  <div class="MsoNormal"></div>

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Field character | B | A | C | H |
| Forward counting | 1 | 2 | 3 | 4 |
| Backward counting | -4 | -3 | -2 | -1 |

<div class="MsoNormal"></div>

<span lang=EN-US style='mso-bidi-font-family:Arial'>This means you can count from the beginning of a field, starting with the first character on position *1*, or from the end of a field, starting with the last character on position *-1*. Thus, you could describe a whole field by using: 1:-1.</span>

<span lang=EN-US style='mso-bidi-font-family:Arial'>You can also specify a range of characters by using two numbers separated by a colon. Thus, *1:3* would indicate the first three characters of a field, and the range *-2:-1* will mean the last two characters. For example, *-3:-2* out of *Beethoven* would be *ve.*</span>   
  <div class="MsoNormal"></div>

So,<span lang=EN-US style='mso-bidi-font-family:Arial'>SoS the script:</span>   
<div class="MsoNormal"></div><span lang=EN-US>

// Only use first 2 chars</span>

<span lang=EN-US><span style='mso-tab-count:1'></span>**Property** FirstTwoCharOfName 1 **Sub** 1:2</span>

<span lang=EN-US><span style='mso-tab-count:1'></span>**Constraint** FirstTwoCharOfName **MaxRep** 1</span>   
  <div class="MsoNormal"></div>

<span lang=EN-US style='mso-bidi-font-family:Arial'>would only keep *Mozart* and *Monteverdi* apart (both '*Mo*'), </span><div class="MsoNormal"></div><span lang=EN-US style='mso-bidi-font-family:Arial'>while:</span>   
  <div class="MsoNormal"></div>

<span lang=EN-US><span style='mso-tab-count:1'></span>// Only use last 4 chars</span>

<span lang=EN-US><span style='mso-tab-count:1'></span>**Property** LastFour 1 **Sub** -4:-1<span style='mso-tab-count:1'></span></span>

<span lang=EN-US><span style='mso-tab-count:1'></span>**Constraint** LastFour **MaxRep** 1</span>   
  <div class="MsoNormal"></div>

<span lang=EN-US style='mso-bidi-font-family:Arial'>would keep *Verdi* and *Monteverdi* apart (because they both end in '*erdi*').</span>

<span lang=EN-US>Constraints on specific values

</span>

<div class="MsoNormal"><span lang=EN-US style='mso-bidi-font-family:Arial'>Sometimes you need to have different constraints for different values. </span><span lang=EN-US style='mso-bidi-font-family:Arial'>Say, you want to Mix the following items containing words and non-words:</span></div>   
  <div class="MsoNormal"></div>

<span lang=EN-US><span style='mso-tab-count:1'></span>word ball</span>

<span lang=EN-US><span style='mso-tab-count:1'></span>word tree</span>

<span lang=EN-US><span style='mso-tab-count:1'></span>word water</span>

<span lang=EN-US><span style='mso-tab-count:1'></span>word house</span>

<span lang=EN-US><span style='mso-tab-count:1'></span>nonword blosp</span>

<span lang=EN-US><span style='mso-tab-count:1'></span>nonword plest</span>

<span lang=EN-US><span style='mso-tab-count:1'></span>nonword huso</span>

<span lang=EN-US><span style='mso-tab-count:1'></span>nonword krup</span>   
  <div class="MsoNormal"></div>

<span lang=EN-US style='mso-bidi-font-family:Arial'>You would not mind having three words in a row, but three non-word items would be too distracting. To accomplish this, you could use this script:</span>   
  <div class="MsoNormal"></div>

<span lang=EN-US>// Label first field as status</span>

<span lang=EN-US><span style='mso-tab-count:1'></span>**Property** status 1</span>   
  <div class="example"></div>

<span style='mso-tab-count:1'></span><span style="mso-tab-count: 1"></span><span lang=EN-US>// Default maximum 3 items identical</span>

<span lang=EN-US><span style='mso-tab-count:1'></span>**Constraint** status **MaxRep** 3</span>   
  <div class="example"></div>

<span lang=EN-US><span style="mso-tab-count: 1"></span>// But nonword items maximum 2 times</span>

<span lang=EN-US><span style='mso-tab-count:1'></span>**Constraint** status **MaxRep** nonword 2</span>   
  <div class="MsoNormal"></div>

First you specify a default constraint: in the example above, <span lang=EN-US style='mso-bidi-font-family:Arial'>this is represented by the first **Constraint** command, which sets a default **MaxRep** value for all items with the property *status*.</span>

<span lang=EN-US style='mso-bidi-font-family:Arial'>By adding a second **Constraint** command, the constraint specialised for *status* having the value *nonword*, the result being that only two nonword items will appear in a row.</span>   
  <div class="MsoNormal"></div>

<span lang=EN-US style='mso-bidi-font-family:Arial'>If you want to put a constraint on nonword items only, you can use a script like this:</span>   
  <div class="MsoNormal"></div>

<span lang=EN-US>// Label first field as status</span>

<span lang=EN-US><span style='mso-tab-count:1'></span>**Property** status 1</span>   
  <div class="example"></div>

<span lang=EN-US>// Only nonword items maximum 2 times</span>

<span lang=EN-US><span style='mso-tab-count:1'></span>**Constraint** status **MaxRep** nonword 2</span>   
 <div class="MsoNormal"></div>

In case you want an even more specific constraint for a very small set of words Mix enables you to set specific Properties. These can be used to make items 'Friends' or 'Enemies', making it possible to keep two or more items apart from each other, or close if that is needed. Beware that in this case the actual value of the fields is being ignored and all items specified are considered identical in the context of this property!

This is done by extending the Property keyword, like this:

**Property** enemies 2 **Specific** doctor nurse

**Constraint** enemies **MinDist** 20

In this case a specific property is created for the items 'doctor' and 'nurse', and a constraint is put on that property to make sure that these two items are at least 20 positions apart in the output. The keyword Specific has to be used before the values are specified. This does not exclude also using the Sub keyword:

**Property** enemies 2 Sub 1 **Specific** e

**Constraint** enemies **MinDist** 20

This will keep all words that start with the letter 'e' at least 20 lines apart.

Pooling values

Mix can 'pool' values on a specified property, meaning that the values on this property will be randomly shuffled over all items containing the property. For example, consider this item file:

cat grey animal small

tree green plant big

dog brown animal small

whale blue animal big

rose red plant small

corn yellow plant small

If we now run the following script:

**Property** colour 2

**Property** type 3

**Constraint** type **MaxRep** 1

**Pool** colour **within** type animal

The output might look like this:

whale brown animal big

rose red plant small

dog blue animal small

corn yellow plant small

cat grey animal small

tree green plant big

As you can see the colour values have been shuffled around within the animals, but not within the plants. The shuffle algorithm does not guarantee that all values will be changed, by the way, so 'cat' is assigned its original colour, just by coincidence.

The 'within' clause is optional: when left out all colours will be shuffled. When no specific value is given for the 'within', like this:

**Pool** colour **within** type

the shuffling will be limited to all items that possess the property 'type', irrespective of its value. It is also possible to give more than one value:

**Pool** colour **within** type animal plant

in which case all lines with property 'type' of one of the given values will be included in the shuffling process. For the example file both of these last version will have identical results, as there are only two values for the property 'type'.

<div class="MsoNormal"></div>

<span lang=EN-US>Fixing items</span>

<div class="MsoNormal"></div>

<span lang=EN-US style='mso-bidi-font-family:Arial'>Suppose you want a certain type of items to be left on their location in the output file. For example, these are the items:</span>   
  <div class="MsoNormal"></div>

<span lang=EN-US><span style='mso-tab-count:1'></span>Start</span>

<span lang=EN-US><span style='mso-tab-count:1'></span>show green</span>

<span lang=EN-US><span style='mso-tab-count:1'></span>show red</span>

<span lang=EN-US><span style='mso-tab-count:1'></span>show blue</span>

<span lang=EN-US><span style='mso-tab-count:1'></span>show black</span>

<span lang=EN-US><span style='mso-tab-count:1'></span>show white</span>

<span lang=EN-US><span style='mso-tab-count:1'></span>End</span>   
  <div class="MsoNormal"></div>

<span lang=EN-US style='mso-bidi-font-family:Arial'>You would like the *Start-* and *End lines* to stay in place, but shuffle all the other ones. Use the **FixType** and the **LineType** command, like this:</span>   
  <div class="example" style="tab-stops:14.2pt"></div>

<span lang=EN-US><span style='mso-tab-count:1'></span>// Define LineTypes:</span>

<span lang=EN-US><span style='mso-tab-count:1'></span>**LineType** startitem 1 Start</span>

<span lang=EN-US><span style='mso-tab-count:1'></span>**LineType** enditem <span style="mso-spacerun: yes"></span>1 End</span>   
  <div class="example"></div>

<span lang=EN-US><span style='mso-tab-count:1'></span>// Fix both of them:</span>

<span lang=EN-US><span style='mso-tab-count:1'></span>**FixType** startitem enditem</span>   
  <div class="MsoNormal"></div>

<span lang=EN-US style='mso-bidi-font-family:Arial'>**FixType** will fix both *startitem* and e*nditem* at their current place in the output file. Note that you can fix multiple LineTypes in one go.</span>   
  <div class="MsoNormal"></div>

<span lang=EN-US style='mso-bidi-font-family:Arial'>Items that are fixed can still have properties on which constraints will operate. For example, if your fixed item has the property *colour*, you can still prevent the next item in your output file having the same colour value, provided that one isn't fixed as well, of course.</span>

You can also fix all items with a certain value for a given property with the command **FixProp**, like this:

Property Type 5

FixProp Type break

This will make sure that items that have the property ‘Type’ and a value of ‘break’ for that property will remain fixed.

**Fix command with Blocks  
  <div class="MsoNormal"></div>**

<span lang=EN-US style='mso-bidi-font-family:Arial'>If your blocks contain items that you do not want to be moved or randomised, you will have to add a **fix** statement to the **Block** command.</span>   
  <div class="MsoNormal"></div>

<span lang=EN-US style='mso-bidi-font-family:Arial'>For example, if the first item in a block is a practice item that should always precede the other items, your script should look like this:</span>   
  <div class="example" style="tab-stops:36.0pt"></div>

<span lang=EN-US><span style='mso-tab-count:1'></span>**Block** part1 **fix** 1</span>

<span lang=EN-US><span style='mso-tab-count:1'></span>show practice</span>

<span lang=EN-US><span style='mso-tab-count:1'></span>show red</span>

<span lang=EN-US><span style='mso-tab-count:1'></span>show blue</span>   
  <div class="MsoNormal"></div>

<span lang=EN-US>The practice item now always will be on the first line. </span>   
  <div class="MsoNormal"></div>

<span lang=EN-US>If you need to fix more items at the beginning of your output file, simply increase the number after **fix**; i.e. 2 will fix the first two lines, 3 the first three, etc. <span lang=EN-US>Note that the **fix** statement described here can only be used with the Block command, and will only fix items at the beginning of your output file.</span>

LineFix and LineBan commands

Sometimes you want lines to be in certain places, like pauses in an experiment. If you want your pauses every 50 items, and you have 200 items in total, you could put your pauses in the right place like this:

**LineFix** PauseItem 50 101 152

This will make sure that the items on lines 50, 101 and 152 are of the linetype PauseItem. Of course you have to provide precisely three lines of the correct type for this to work. Too many PauseItems and they will be in places where you don’t expect them, too few and Mix will fail with an error message.

Likewise you can use the command **LineBan** to make sure that items in a certain location are not of a specific type.

You can even fix items on, or ban items from locations within a block using this:

**BlockLineFix** Condition1 PauseItem 50 101 152

Now the items will be on lines 50, 101 and 152 from the beginning of the block 'Condition1'. If you need to fix a whole series of items you could use the 'Sequence' keyword, like this:

**LineFix** Break **Sequence** 1 12 30

which will fix 12 'Break' items on positions 30 items apart, starting from position 1.<div class="MsoNormal"></div>

Header and Footer commands   
  <div class="MsoNormal"></div>

<span lang=EN-US style='mso-bidi-font-family:Arial'>If you want every output file to start with the same block, without randomising this block or adding extra fields, it is easier to write this block down in a separate file (for instance, *head.txt*) and use the **Header** command in your script file:</span>   
  <div class="MsoNormal"></div>

<span lang=EN-US><span style='mso-tab-count:1'></span>**Header** head.txt </span>   
  <div class="MsoNormal"></div>

<span lang=EN-US>Now, every output file will start with the content of the file *head.txt*. More than one file can be specified.

The command **Footer** works just like **Header**, but adds the content of the file to the end of the output</span><div class="MsoNormal"></div>.

Additionally, the existing header of an item file can be kept by using “**KeepHeader**”. The header line will be ignored during the actual mixing process and added to the top of all outputfiles unchanged. To only ignore a header, and not include it in the output, use **“IgnoreHeader”**.

Include statement   
  <div class="MsoNormal"></div>

<span lang=EN-US style='mso-bidi-font-family:Arial'>A useful command for generating item files is the **include** statement. With this you can include the content of other files in your item file.</span>   
  <div class="MsoNormal"></div>

<span lang=EN-US style='mso-bidi-font-family:Arial'>For example: assume that you have three files with three different groups of items for which you want to make different blocks. The files are named *group1*, *group2* and *group3* and are in the folder *C:\items*. Instead of copying the content of those three files into an item file you can write an item file like this:</span>   
  <div class="MsoNormal"></div>

<span lang=EN-US><span style='mso-tab-count:1'></span>**Block** b1</span>

<span lang=EN-US><span style='mso-tab-count:1'></span>**include** C:\items\group1</span>

<span lang=EN-US><span style='mso-tab-count:1'></span>**Block** b2</span>

<span lang=EN-US><span style='mso-tab-count:1'></span>**include** C:\items\group2</span>

<span lang=EN-US><span style='mso-tab-count:1'></span>**Block** b3</span>

<span lang=EN-US><span style='mso-tab-count:1'></span>**include** C:\items\group3 grp3 $3 $5</span>   
  <div class="MsoNormal"></div>

<span lang=EN-US>and Mix will create three blocks using the items in these three files. Mix will treat this item file as if the</span><span lang=EN-US style='mso-bidi-font-family:Arial'> content of the three files was actually in the item file. This way you do not have to use Mix statements and items together in one file.</span>

The third line shows an extra feature of the include command: you can select certain fields from the given file, and add fields if you need to. In this case all items in block b3 will have “grp3” as their first field, and then the third and fifth field from the given file as their second and third field.

<span lang=EN-US>Defining input delimiters

</span>

<div class="MsoNormal"><span lang=EN-US>Fields in the item file have to be separated by a delimiter. Without further specification, Mix will recognise a space or tab-character as a delimiting character.</span></div>

<span lang=EN-US>If you want Mix to recognise other delimiting characters as well, you will have to use the command **InputDelimiter**. The syntax for this command is:</span>   
  <div class="MsoNormal"></div>

<span lang=EN-US>**InputDelimiter "**<delimiters>**"**</span>   
  <div class="MsoNormal"></div>

<span lang=EN-US>You can specify multiple delimiting characters in one go:</span>   
  <div class="MsoNormal"></div>

<span lang=EN-US>**InputDelimiter "**,#;**"**</span>   
  <div class="MsoNormal"></div>

<span lang=EN-US>Mix will now use commas, hashes and semicolons as delimiting characters. You can also specify special characters, provided they are preceded by a backslash, such as a tab-stop: "\t" and the backslash itself: "\\".</span>

You can also make Mix recognise empty fields by using the keyword **DisAllowMultipleInputDelimiters**. By default Mix will interpret multiple occurrences of a delimiter as a single separation between two fields, but with this keyword you can force it to see empty fields.

Counting property occurrences

<div class="MsoNormal"><span lang=EN-US>The command **Occurrence** counts how many times a property has already occurred within each block. For example:</span></div>   
  <div class="MsoNormal"></div>

<span lang=EN-US>**Property** colour 1</span>

<span lang=EN-US>**Occurrence** colour</span>   
  <div class="MsoNormal"></div>

<span lang=EN-US>will generate this output:</span>   
  <div class="MsoNormal"></div>

<span lang=EN-US>green 1</span>

<span lang=EN-US>blue 1</span>

<span lang=EN-US>green 2</span>

<span lang=EN-US>white 1</span>

<span lang=EN-US>green 3</span>

<span lang=EN-US>white 2</span>

<span lang=EN-US>blue 2</span>   
  <div class="MsoNormal"></div>

<span lang=EN-US>You can add any number of properties to be counted to the **Occurrence** command:</span>   
  <div class="MsoNormal"></div>

<span lang=EN-US>**Occurrence** <property <property> ...</span>   
  <div class="MsoNormal"></div>

The order in which the counters are added to the output, will be the same as the order in which the properties were listed with the **Occurrence** command.

Adding numbers

Adding block numbers

<div class="MsoNormal"><span lang=EN-US>The command **BlockNumbers** will add the number of a block to the end of every line. Block numbers will always be added after occurrence numbers. Mix starts numbering at 1 and counts blocks in the order in which they appear in the output file.</span></div>

<span lang=EN-US>Adding line numbers

</span>

<div class="MsoNormal">The command:</div>   
  <div class="MsoNormal"></div>

<span lang=EN-US>**LineNumbers** <**Prefix**/**Postfix>**</span>   
  <div class="MsoNormal"></div>

<span lang=EN-US>will add a line number to each line in the output file. This is the number of the line as it occurs in the output file. Mix will not count empty lines. By adding either **Prefix** or **Postfix** to the command, you can choose whether the line number will appear at the beginning of each line or at the end. Line numbers are always the first or the last field of a line.</span>

<span lang=EN-US>Aligning output</span>

<div class="MsoNormal"></div>

<span lang=EN-US style='mso-bidi-font-family:Arial'>Suppose that after mixing your output file looks like this:</span>   
  <div class="MsoNormal"></div>

<span lang=EN-US><span style='mso-tab-count:1'></span>practice tomato</span>

<span lang=EN-US><span style='mso-tab-count:1'></span>test fruit yellow 25 3</span>

<span lang=EN-US><span style='mso-tab-count:1'></span>trial condition3 red 250 2</span>

<span lang=EN-US><span style='mso-tab-count:1'></span>trial baseline green</span>

<span lang=EN-US><span style='mso-tab-count:1'></span>end wait</span>   
  <div class="example" style="tab-stops:14.2pt"></div>

<span lang=EN-US style='mso-bidi-font-family:Arial'><span style="mso-bidi-font-family: Arial">Perhaps you </span></span><span lang=EN-US style='mso-bidi-font-family:Arial'>would like your output to be in nice straight columns. To accomplish this, you can use the command </span> <span lang=EN-US style='mso-bidi-font-family:Arial'><span style="mso-bidi-font-family: Arial">**C</span></span><span lang=EN-US style='mso-bidi-font-family:Arial'>omb** in your script file:</span>   
  <div class="MsoNormal"></div>

<span lang=EN-US><span style='mso-tab-count:1'></span>**Comb** 3</span>   
  <div class="MsoNormal"></div>

<span lang=EN-US style='mso-bidi-font-family:Arial'>Your output file will now look like:</span>   
  <div class="MsoNormal"></div>

<span lang=EN-US><span style='mso-tab-count:1'></span>practice   tomato</span>

<span lang=EN-US><span style='mso-tab-count:1'></span>test       fruit        yellow   25    3</span>

<span lang=EN-US><span style='mso-tab-count:1'></span>trial      condition3   red      250   2</span>

<span lang=EN-US><span style='mso-tab-count:1'></span>trial      baseline     green</span>

<span lang=EN-US><span style='mso-tab-count:1'></span>end        wait</span>   
  <div class="MsoNormal"></div>    
  <div class="MsoNormal"></div>

<span lang=EN-US style='mso-bidi-font-family:Arial'>The number added to the Comb command is the minimal number of spaces that should be inserted. The default value for this statement is *1*; at least 1 space will always be inserted in between items.</span> <span lang=EN-US style='mso-bidi-font-family:Arial'><span style="mso-bidi-font-family: Arial">All counters (occurrence, block number or line number) are also aligned by the Comb command.   
  <div class="MsoNormal"></div>

If you used the **InputDelimiter** command, **Comb** will ignore all the characters you have used in this command and replace them with spaces. If you want your output lines to be exactly the same as your input lines, then do not use Comb.

Lines that are converted (see next section) cannot be 'combed'.</span></span>

Converting the output

Mix can convert every line as it is written to the output file. For example, you can use this:

**Convert** StimLine PracticeLine

+%2 <%% 35> "+" / \* "%3" ; \line

**End Convert**

To convert an input line that looks like this:

VLD 143 Ladybird

To an output line that looks like:

+143 <% 35> "+" / \* "Ladybird” ; \line

As you can see, every occurrence of ‘%2’ or ‘%3’ is replaced by the second or third field of the input line. If you need a ‘%’ character in the output line, just use ‘%%’.

It is possible to define conversion for several line types in one go. The example above will convert lines of type 'StimLine' and type 'PracticeLine'. More than one convert command can be specified, to convert different line types in different ways. If no line type is provided all lines will be converted, except the ones that appear in another specific ‘Convert’ command. The conversion definition has to be closed with the 'End Convert' line, and embedded conversion definitions are not allowed.

A special case is the form "%%%3" which means that the number of the field to be inserted is found on field number 3. In this way you can control which field ends up in the output by having different values on field 3. Rarely needed, of course.

Sadly, adding occurrences, line numbers and block numbers or ‘Comb’-ing the output doesn’t work when you use the Convert option.

<span lang=EN-US>Repeating the same randomisation

</span>

<div class="MsoNormal"><span lang=EN-US style='mso-bidi-font-family:Arial'>If you want to repeat the same randomisation several times, you can use the **seed** command in your script file. For example:</span></div>   
  <div class="example" style="tab-stops:14.2pt"></div>

<span lang=EN-US><span style='mso-tab-count:1'></span>**seed** 6</span>   
  <div class="MsoNormal"></div>

<span lang=EN-US style='mso-bidi-font-family:Arial'>makes sure that each call to Mix with the same item file(s) and the same script file containing this statement will each time produce the same output. You can change the number after **seed** to generate several different—but repeatable—randomisations.</span>

Checking all your definitions with a labelled output file

Mix script can get complicated, and sometimes it is very practical to be able to see which lines in the item file will be assigned which LineTypes and Properties. To do this us the following command:

**ProduceLabelledOutputFile** <filename>

Which will write all your items to the specified filename, but complete with all LineType and Property information, looking like this:

ant stim word tri46 29

WordLine

Word : ant

Type : stim

Status : word

Signal : tri46

StimNum : 29

FirstChar : a

LastChar : t

This output shows first a copy of the original item-line followed by a number of indented lines indicating that the type of this line was 'WordLine' and that it had 7 properties, some of which were clearly defined on a part of an input field. This will provide valuable information, enabling you to check the correctness of your scripts.

Getting help

There’s a Mix website, where the program can be downloaded. It can be found on the MRC-CBU website. Additionally, there’s a Mix email list. Just mail to [Mix@mrc-cbu.cam.ac.uk](mailto:Mix@mrc-cbu.cam.ac.uk) for more information.

Copyright Maarten van Casteren, 2008.