



# ***SendIgence User's Manual***

*Version 1.2.*

By Igor Grešovnik, 2008



## **Contents:**

<b>1</b>	<b>Introduction.....</b>	<b>1</b>
<b>2</b>	<b>Availability.....</b>	<b>1</b>
2.1	License.....	2
2.2	Availability.....	2
<b>3</b>	<b>Use of the program.....</b>	<b>3</b>
3.1	<b>Basic use.....</b>	<b>3</b>
3.1.1	Installation .....	3
3.1.2	Running the program interactively .....	3
3.1.3	Possible actions  .....	4
3.1.4	Running the program non-interactively .....	5
3.2	<b>How-to guide.....</b>	<b>6</b>
3.2.1	Sending periodic messages to a group of participants  .....	6
<b>4</b>	<b>Database (XML) Structure and Commands .....</b>	<b>6</b>
4.1	<b>List of parameters by groups .....</b>	<b>7</b>
4.1.1	Program info .....	8
4.1.2	General parameters .....	8
4.1.3	Sending parameters.....	9
4.1.4	Recipient parameters.....	12
4.1.5	Definition of e-mail client and its configuration.....	16
4.1.6	Action list.....	20
4.2	<b>List of commands .....</b>	<b>20</b>
4.2.1	General commands.....	20
<b>5</b>	<b>Additional information .....</b>	<b>22</b>
5.1	<b>FAQ (Frequently Asked Questions) .....</b>	<b>22</b>
5.1.1	What is an incoming SMTP e-mail client? .....	22
5.1.2	What is an outgoing mail SMTP server? .....	22
5.1.3	What is necessary to relay e-mails? .....	22
<b>6</b>	<b>Development .....</b>	<b>23</b>
6.1	<b>Version history .....</b>	<b>23</b>
6.1.1	Development of versions in reverse chronological order.....	23
6.2	<b>To do (wish list) .....</b>	<b>24</b>

**SendIgence** is a freeware and comes with no warranty.

## ***1 INTRODUCTION***

*SendIgence* is a wrapper for programs for sending e-mails that are called from command-line and their operation is controlled through input command-line arguments. An example of such a program is [\*sendEmail\*](#). It is simple to use and it can be run either interactively or by using an instruction file in XML format. Instruction files can be created in interactive *SendIgence* session or edited manually (an XML editor is recommended). The program is intended to extend capabilities of e-mailing programs by sending large amount of e-mails, dividing recipients to smaller groups, timing the process of sending e-mails, extracting addresses from files, etc.

*SendIgence* offers the following capabilities:

- Use of different command line e-mail clients (although *sendEmail* is recommended).
- Sending mails to large number of addresses.
- Dividing the whole set of addresses to smaller subgroups.
- Defining pause between sending the mail to subgroups.
- Storing sessions to an XML file.
- Checking the e-mail before sending to final recipients.
- Extracting addresses from files.
- Extracting addresses from multiple files (ability to browse directories recursively).
- Eliminating repeating addresses.
- Defining exclusion lists.
- Distributing internet traffic more evenly by sending e-mails over larger periods of time.
- Choosing the number of recipients and pause between messages randomly (with specified average and maximal deviation).

The program is run from a terminal window and it is extremely simple to use. Although it is not stand-alone and needs another e-mail client, this is not a major obstacle. In the package, the free command line e-mail client called *sendEmail* is included in the package provided on the home page of the program. This e-mail client is used by default, however the program can be configured to use other e-mail clients.

## **2 AVAILABILITY**

## ***2.1 License***

SendIgence is a freeware and comes with no warranty. Anybody can use the programs for any purpose except for sending spam (see my definition of spam below). The program is provided as it is, and its authors are not liable for any kind of malfunction or misuse.

### **Definition of spam:**

Spam are annoying or disturbing e-mail messages that are sent in massive quantities and with malicious purposes, such as for spreading malicious software, excessively increasing internet traffic, for unwanted advertising, etc. As it is common in real-life, the borders between spam and non-spam are sometimes unclear. According to my definition, informing a larger group of recipients about something is not spam when the sender honestly believes that the group contains a large portion of potentially interested recipients (i.e. if the group is carefully targeted), when the sender believes that the mail will not be disturbing for recipients, if the same information is not sent out several times, and if the messages contain clear and simple means of unsubscribing from receiving further information about the subject (or, even better, if active subscription is required to receive further mailing) and if the sender does not try to camouflage the true contents and the purpose of the message in any way. I also recommend to read and to consider the Gmail's directives for sending bulk mail, which are found at the following address:

[http://www.google.com/mail/help/bulk\\_mail.html](http://www.google.com/mail/help/bulk_mail.html)

Program was created by [Igor Grešovnik](#). The author provides the program in hope to be useful to others and does not require any compensation for its use. If you feel sympathy for author's work, you can take patronage of his work in one of the following ways:

- Donation
- Inviting the author to participate in funded development or research programs

## ***2.2 Availability***

Currently the software is available only as executable for 32-bit MS Windows systems. It is planned that the program source code will be available in the future, but there are no estimations on when this might happen.

## 3 USE OF THE PROGRAM

### 3.1 *Basic use*

#### 3.1.1 Installation

Installation is simple: just unpack the files and copy executables (i.e. the *SendIgence* program and the command-line client, the `sendEmail`) to a directory from which these programs can simply be run.

In order to use the program, a command-line SMTP e-mail client (see question 5.1.1) must be installed on the system in such a way that it can be run from the directory where *SendIgence* is run. It is recommended that both the e-mail client and the

#### 3.1.2 Running the program interactively

In order to use the program interactively, open a terminal window and run the program without any arguments:

```
SendIgence
```

The program will lead you step by step to definition of various parameters, and definition and execution of actions.

The configurable program parameters define the context (program state) in which actions are performed, and can be divided into three main groups:

- **E-mail client configuration:** program name and names of its command-line options (e.g. for specifying recipient address, message body, etc.)
- **Sending configuration:** values of various e-mail header fields such as the Subject, the To, Cc and Bcc; SMTP server parameters (server name, user name and password when necessary); address parameters (a list of addresses and specification of files containing the address); grouping parameters (e.g. how many recipients are included in a single e-mail, with possibility to define a random deviation from this); timing parameters (the number of seconds before sending subsequent e-mails).
- **Action list:** defines what actions and in what order are to be performed.

##### 3.1.2.1 Interactive input of parameters

**Input of integer and real number parameters:**

- ‘?’ (a question mark + <Enter>) will print instructions for inserting a number.

- Alphabetical character (such as 'a' or 't') + <Enter> will print the current value of the parameter.
- <Enter> will keep the current value of parameter.
- Number in a valid format + <Enter> will change the parameter to that number.

For integers, valid formats are for example 3238, -34, +64, -007, 009.

For real numbers, valid formats are all integer formats + standard C number representations, e.g. 10, -42, +04, 5.56, -0.064, 6.3e-5, -5.23E-54, 0.03e+2, etc.

#### **Input of string parameters:**

- '?' (a question mark + <Enter>) will print instructions for inserting a number.
- <Enter> will keep the current value of parameter.
- e\ will insert an empty string (allocated but with no characters).
- n\ will insert a NULL string.
- a\ will switch on the append mode, inserted string will be appended to the old one. Input c\ to cancel appending.
- d\ will delete the currently inserted string (useful in append mode).
- Any sequence of characters + <Enter> will change the parameter to the inserted string.

Escape sequences (i.e. the backslash \ followed by specification character) can be used for inserting special characters:

\\ - backslash

\n - newline

\r - carriage return

\t - tab

\v - vertical tab

\f - formfeed

\d### (where each # represents a digit) – a character with ASCII code in decimal notation

\x## - character in hexadecimal notation

### **3.1.3 Possible actions**



#### **3.1.3.1 Importing parameters from an existent instruction file (XML format)**

Parameters that are defined in the instruction file will replace the current values. Other parameters will remain unaffected.

Current parameter values at the time of instruction file import can be obtained in two ways. When the program starts, some parameters are pre-set to their default values, in particular the e-mail client configuration. On the other ways, values of some parameters can be obtained by interactive user input and by eventual previous imports.

#### **3.1.3.2 Exporting settings to an instruction file**

All defined parameters are exported into a file (partial exports are currently not supported).

In order to keep partial settings, one must keep different files, each with its own group of parameters. Changes must then be applied to each file separately by interactive sessions where only one file at a time is imported and then exported after user's change of selected parameters. In this way, parameters can later be imported in separated groups in later sessions.

### 3.1.3.3 Sending e-mails

Sending of e-mails is performed according to the current configuration parameters.

## 3.1.4 Running the program non-interactively

To run the program non-interactively, simply run it with a number of command-line arguments that either determine commands to be run, or configuration data files to be imported. Execution of commands and import of data from configuration files is performed in the same order as the commands and files are listed in command-line arguments. For example, the command

```
SendIgence confserver.xml confrecipients.xml confsending.xml TestSend Send
```

will first import the data (configuration & other parameters) from the files `confserver.xml`, `confrecipients.xml` and `confsending.xml`, send a test e-mail (command `TestSend`) and then perform sending of e-mails (command `Send`) according to parameters read from these files (and also according to the default parameters that were not overwritten by the data from imported files).

The non-interactive mode can be used for different purposes, in particular for:

- Repeating a session with similar parameters (e.g. sending a different message to the same group of recipients; in this case)
- Sending e-mails without the need for user interaction, e.g. to automation of regular (in some cases periodic) tasks

The **configuration files** exported must contain all the relevant parameters needed by the program to do what is expected from it. These are XML files that contain the data in a prescribed structure. Any elements of the data may be omitted from these files, and data from files that are imported later, overwrites the data imported earlier (or the data that is set at program initialization, such as default parameters).

The data in configuration files is usually produced by user input in an interactive session after which the data are imported. However, it is also possible (and in fact quite easy, see Section 4) to edit the configuration files directly and set parameters there.

### 3.1.4.1 Non-interactive run: advanced topics



### 3.2 *How-to guide*

This section describes a number of typical uses of the program.

#### 3.2.1 Sending periodic messages to a group of participants

## 4 DATABASE (XML) STRUCTURE AND COMMANDS

All parameters that define what the program does when certain commands are executed are stored in an internal database with an XML structure. Parameters can therefore be simply exported (saved) to a file all at a time or they can be imported from a file.

It is a common practice to set groups of parameters in an interactive session and export them to a file, from which they can be imported in later runs of the program. One reason to do that is to store a set of commonly used parameters (such as outgoing mail server settings, such that they can be imported all at one time later, without the need to set each of them individually. Another reason is to enable non-interactive runs in which all relevant parameters are imported from the files and the desired actions performed automatically, without user intervention.

Any of the parameters can be undefined at any time, either within a program or in a configuration file. When importing configuration files where some of the parameters are not defined, the current values of such parameters remain unchanged, where others are overwritten by the values defined in the files. What concerns exporting, all parameters that are currently defined are exported at once. Some parameters are of array (or table) type, in such cases the whole tables are replaced when importing the data from a file.

XML is an arbitrary tree structure. Basic constituents are *elements*, each of which has a *name*, can have a *value*, can have an arbitrary number of *attributes*, and any number of *subelements*.



Parameter names coincide with the names of the corresponding XML elements in the database and are used to identify pieces of data in the database. Values of XML elements are used to define values of simple data pieces, such as options, strings, or simple elements of tables. Attributes are used as metadata and serve e.g. for type checking and for easier browsing (e.g. they specify how many sub-elements have composed parameters such as vectors). Element attributes appear in key-value pairs, i.e. each attribute has its name (key) and a value.

Parameters with names are arranged at most two levels in depth, they can reside either in the outermost level, or in the first level. In the latter case they are sub-elements of outer-most elements, which are *containers* for groups of parameters. Beside that, parameters can be further composed and can have themselves several levels in depth.

When parameters are exported, the resulting XML files are written in such a way that nested elements are indented, therefore it is relatively simple to edit the resulting files with any text editor. It is advisable to use an editor with syntax highlighting and useful tools such as checker of bracket matching (e.g. *EditPlus*), which reduces the possibility of errors. It is even better to use an XML editor that enables expansion and shrinkage of XML elements. For example, the MS Visual Studio 2008 that is freely available for Windows platforms (it only requires registration over the Internet, but this is free) is more than enough powerful for comfortable editing of configuration files.

The following Subsection 4.1 lists the parameters that are used by the program.

### ***4.1 List of parameters by groups***

This Subsection lists parameters by their identification as it appears in the XML data structure (internal as well as in configuration files). When parameters are read interactively, these identifications are usually printed out, and they appear as element names in the XML configuration files. Where the ID consists of two strings separated by a slash (e.g. “*grp\_name/el\_name*”), this denotes parameters that are nested in a group, i.e. they are sub-elements of a container element named by the first string (in this case *grp\_name*). If there is no string before the slash, the corresponding parameter is stored at an outer-most level.

#### **General remarks**

For string parameters, it is usually considered that they are undefined when their values are empty strings.

In the case of undefined parameters, the program will either complain and launch an error message (and possibly not finish the performed task properly), or ask the user to insert the missing parameter interactively, or assume some default values for these parameters and continue the operation normally. The behavior depends on the specific operation, and the program is designed in such a way that the behavior is as logical and expected as possible.

### 4.1.1 Program info

Data from this section provides some basic information about the program and it is not be modified by the user.

#### 4.1.1.1 /Program\_Name

Name of the program.

#### 4.1.1.2 /Program\_Version

Version of the program.

#### 4.1.1.3 /Program\_Year

Year when this version of the program was issued.

#### 4.1.1.4 /Program\_Author

Author of the program.

#### 4.1.1.5 /Program\_Home

Program home page.

### 4.1.2 General parameters

#### 4.1.2.1 Program\_Configuration/Session\_File

String. Contains the name of the session file where all program data are stored (or can be restored from) – used by commands `SaveSession` and `RestoreSession`.

This file is typically used for saving and restoring a complete set of parameters, while the file specified by *Export\_File* is typically used for storing incomplete groups of parameters. With other words, the role of saving the session data is a bit different from the role of exporting and importing the data. Saving the session is usually done in order to be able to restore the session in the case that we need to interrupt interactive work for any reason or when the system or program crashes. Therefore, the name of this file is usually not changed by the user – in this way we can always easily find data dumps that we have made by the `SaveSession` commands.

See commands `ImportData` (Section 4.2.1.2) `RestoreSession` (Section 4.2.1.4)

#### 4.1.2.2 Program\_Configuration/Export\_File

String. Defines the name of the file to which data is exported when the command `ExportData` is executed.

#### **4.1.2.3 Program\_Configuration/Import\_File\_List**

Table of strings. Defines the list of names of files from which data is imported when the command `ImportData` is executed. Data is imported from files in the same order as they are listed in the table.

#### **4.1.2.4 Program\_Configuration/Log\_File**

String. Defines the program's log file. This is a file into which the program writes reports about performed operations. In such a way, the operation can be traced and checked for errors.

After individual sends are executed, the whole send commands are written to the log file, and since these contain (in the command-line argument list) e-mails of recipients of the current send, the log file can be used for exclusion list when bulk sending (which is broken into multiple sends) is interrupted in the middle of work. In this way, one can at a later time easily send the mail to all intended recipients (whose mails are typically extracted from files), except to those to which the mail has already been sent. In order to do that, just rename the log file, list the recipient and exclude files as in the broken session, list in addition the renamed log file in the exclusion files list, and repeat the sending operation with the same values of relevant parameters.

### **4.1.3 Sending parameters**

These parameters define how the sent messages are composed and how they are sent (e.g. through which SMTP server).

#### **4.1.3.1 Sending\_Configuration / Message\_Head\_From**

String parameter.

Sender's address. This string is written in the "From" field of the message.

#### **4.1.3.2 Sending\_Configuration / Message\_Head\_ReplyTo**

String parameter.

Address for replies. This string is written in the "Reply-to" field of the message. It does not need to be specified if the address is the same as the sender's address (i.e. the address specified in the "From" field, Section 4.1.3.1)

#### **4.1.3.3 Sending\_Configuration / Test\_Recipient**

String parameter.

Address of the test recipient. Test messages are sent to the address specified by this string (command `TestSend (<recipient>)`, Section 4.2.1.10).

#### **4.1.3.4 Sending\_Configuration / Message\_Head\_To**

A table of e-mail records.

A list of addresses that will be added to the “To” field of all messages that are sent in the regular sending actions (command Send, Section 4.2.1.9).

#### **4.1.3.5 Sending\_Configuration / Message\_Head\_Cc**

A table of e-mail records.

A list of addresses that will be added to the “To” field of all messages that are sent in the regular sending actions (command Send, Section 4.2.1.9).

#### **4.1.3.6 Sending\_Configuration / Message\_Head\_Bcc**

A table of e-mail records.

A list of addresses that will be added to the “Bcc” field of all messages that are sent in the regular sending actions (command Send, Section 4.2.1.9).

#### **4.1.3.7 Sending\_Configuration / Message\_Subject**

String parameter.

The subject of messages that are sent by the program (e.g. command Send, Section 4.2.1.9, or TestSend (<recipient>), 4.2.1.10).

#### **4.1.3.8 Sending\_Configuration / Message\_Body**

String parameter.

The body of messages that are sent by the program (e.g. command Send, Section 4.2.1.9, or TestSend (<recipient>), 4.2.1.10).

It is more usual to specify the file that contains the body (Section 4.1.3.9). In such a case this parameter should not be defined).

#### **4.1.3.9 Sending\_Configuration / Message\_Body\_File**

String parameter.

The name of the file that contains the body of messages that are sent by the program (e.g. command Send, Section 4.2.1.9, or TestSend (<recipient>), 4.2.1.10).

The body file can be a text file or a html file. In the case that it is a html file, it should contain a <html> tag at the very beginning (at least this is true for the *sendEmail* e-mail client).

When the file containing the message body is specify, the body should not be specified directly (i.e. the parameter Sending\_Configuration / Message\_Body should not be specified, see Section 4.1.3.8).

#### **4.1.3.10 Sending\_Configuration / Message\_Attachments**

Table of strings.

This parameters contains names of the files that will be attached to the messages that are sent by the program (e.g. command Send, Section 4.2.1.9, or TestSend (<recipient>), 4.2.1.10). These files do not consist the body of the message, but will arrive with the message as separate attachments that can be saved by the recipient.

#### **4.1.3.11 Sending\_Configuration / Message\_Headers**

Table of strings.

A list of additional headers that will be added to the messages that are sent by the program (e.g. command Send, Section 4.2.1.9, or TestSend (<recipient>), 4.2.1.10). These files do not

---

consist the body of the message, but will arrive with the message as separate attachments that can be saved by the recipient.

For example, if you are sending an e-mail to a larger group of recipients (e.g. more than 50), it is nice to indicate this by adding the header

`Precedence: bulk`

to the messages (according to Gmail's recommendations for sending bulk e-mail).

#### **4.1.3.12 Sending\_Configuration / Outgoing\_SMTP\_Server**

String parameter.

Specifies the outgoing e-mail SMTP server that is used for sending the messages.

The default is "*localhost:25*", but you will probably have to modify this.

In order to send out e-mails, you should specify the existing SMTP server for which you have the appropriate credentials (e.g. a server of your e-mail provider on which you have an account). In most cases you will also have to provide the user name and password for authentication with that server. Some SMTP servers don't require authentication, but restrict locations from which you can use them for sending e-mails (e.g. by permitting only a specific set of IP addresses from which you can send valid e-mail sending requests).

A typical example is that you have a **Gmail** account. In this case you can use the Gmail's SMTP server, which is "*smtp.gmail.com:587*". You will have to provide the user name and password for authentication with this server.

#### **4.1.3.13 Sending\_Configuration / Option\_Network\_Timeout**

String parameter.

This parameter should represent the timeout for network operations performed by the e-mail client, in the format as required by the e-mail client (usually it should represent an integer number of seconds).

With *sendEmail*, it is usually OK if the timeout is not specified.

#### **4.1.3.14 Sending\_Configuration / Outgoing\_SMTP\_User**

String parameter.

User name used for authentication when the SMTP server is contacted by the e-mail client to pass on the e-mails (see Sections 4.1.3.12 and 4.1.3.15).

Some servers do not require authentication and in such cases the user name should not be specified.

#### **4.1.3.15 Sending\_Configuration / Outgoing\_SMTP\_Password**

String parameter.

Password for your user account used for authentication when the SMTP server is contacted by the e-mail client to pass on the e-mails (see Sections 4.1.3.14 and 4.1.3.12).

##### **Special warnings:**

The password is visible for the time when you are typing it in (because an usual terminal input is used) and disappears from the input terminal window only when you press the final <Enter> that confirms the input.

It is not safe to store the password in the program's database (from which it can easily be unknowingly exported to data files where everybody with access to these files can read it).

The program enables you that the passwords that you input are not stored in the database, but are only remember by the program until the end of the session (until the program exits).

Every time you are asked for the password, you will be warned about this and asked whether the password you entered should be stored in the database. Do not store it in the database unless you are absolutely sure that nobody will have access to the eventually exported files with program data. Note that access privileges for files are easily loosened e.g. when creating backups.

#### **4.1.3.16 Sending\_Configuration / Sending\_Log\_File**

String parameter.

Name of the log file that will be used by the e-mail client for logging. For example, if you are using the *sendEmail* client, the program will report details about sending e-mails in this file (e.g. about exchange of requests with the SMTP server, failed requests, etc.). It is sometimes useful to check this file to be sure whether sending of all e-mails was successfully performed. A lot can also be seen from the program's log file (see Section 4.1.2.4).

#### **4.1.3.17 Sending\_Configuration / Sending\_Verbose**

Integer parameter.

If it is 1 then the e-mail client will be instructed to operate in verbose mode, outputting more information about its operation. If 0, the client will not be run with verbose mode.

#### **4.1.3.18 Sending\_Configuration / Sending\_Quiet**

Integer parameter.

If it is 1 then the e-mail client will be instructed to operate in quiet mode, outputting less or none information about its operation. If 0, the client will not be run in quiet mode.

### **4.1.4 Recipient parameters**

These parameters specify to whom the e-mails will be sent.

Note that the parameters that specify the static lists of recipients under the “*To*”, “*Cc*” and “*Bcc*” fields (which are included in every e-mail that is sent by the command *Send*, Section 4.2.1.9), were listed in Section 4.1.3 (more specifically in Sections 4.1.3.4, 4.1.3.5 and 4.1.3.6). The group name for these parameters is different.

In this Section we describe the parameters that define the list of recipients that are sent by the *Send* command (Section 4.2.1.9). Mails can be sent to these recipients in smaller groups, and parameters listed in this Section also define the number of recipients per e-mail (which can have random deviations) and the pauses that are eventually made between successive sends in order to more equally distribute the traffic (these can also be random).

Parameters listed here also specify in which field the recipients of the sent e-mails are listed (i.e. “*To*”, “*Cc*” or “*Bcc*”, see Section 4.1.4.6).

#### **4.1.4.1 Bulk\_Sending\_Configuration / Recipient\_List\_Files**

Table of strings.

Specifies names of the from which recipient addresses are extracted. These must be text files, but apart from this e-mail addresses can be extracted from files which are formatted very differently.

When preparing the list of all recipients to which the mail will be sent by the *Send* command (Section 4.2.1.9), e-mails are extracted from all the files whose names are specified in this list.

File names can include wildcats, such as "*conf\*.eml*". Such filenames are correctly resolved (usually to multiple file names) before addresses are extracted from all files defined by the file names.

When specifying paths with file names, Unix path separators can also be used on Windows (i.e. the slash, '/', instead of the backslash, '\').

It is also possible to use recursive browsing of directories for the specified files (see Section 4.1.4.3).

#### 4.1.4.2 Bulk\_Sending\_Configuration / Exclude\_List\_Files

Table of strings.

Specifies names of the files from which addresses for exclusion list are extracted. When the list of recipients is formed, all recipients that are on the exclusion list are removed from the recipient list before the e-mails are sent by the *Send* command (Section 4.2.1.9). This enables one to easily maintain e.g. the unsubscribing service (one must just put all e-mail addresses from which unsubscribe requests were sent, to an exclusion list file).

Similar rules as for recipient list files are also valid for exclusion list files (including the possibility of using wildcards and recursive directory listing - see Section 4.1.4.3).

#### 4.1.4.3 Bulk\_Sending\_Configuration / Recursive\_Address\_File\_Listing

Integer parameter.

Specifies whether files in the recipient file list (Section 4.1.4.1) and files in the exclusion file list (Section 4.2.1.9) are searched for by recursively listing the containing directories and all of their subdirectories.

If the value is non-zero then files will be resolved by recursively listing the subdirectories, otherwise (if value of the parameter is 0) only the specified directory will be looked up for the files.

For example, if recursive listing is enabled (e.g. the parameter has value 1) and the list of files contains

*"my\_mail\_account/bus\*.eml"*

then this file name will match all files whose names match the wildcard specification *bus\*.eml* and are contained in the directory *my\_mail\_account* (contained in the current directory) and **all its subdirectories**. For example, the following files would all match this specification if they existed:

*my\_mail\_account/business\_january1.eml* ,  
*my\_mail\_account/business\_week.eml* ,  
*my\_mail\_account/real\_estate/business.eml* ,  
*my\_mail\_account/real\_estate/business\_housess.eml* ,  
*my\_mail\_account/vehicles/cars/business1.eml* .

If such a name is contained in the list of recipient files or exclusion files, the name will first be resolved and then all files that match the name in the above described sense are worked (i.e. e-mails are extracted from them and added to the appropriate list, either the recipient or exclusion list).

#### 4.1.4.4 Bulk\_Sending\_Configuration / Recipient\_List

Table of e-mail records.

Specifies a list of recipients that is directly stored in the program's database (i.e. in contrary to specifying the files from which recipient addresses are extracted – see Section 4.1.4.1).

When the final list of recipients is prepared, recipients extracted from the files are added to recipients that are in this list, and finally addresses that are contained in the joint exclusion list (addresses extracted from exclusion files plus those listed directly in the database) are removed from the list. This is all done before the e-mails are sent by the *Send* command (Section 4.2.1.9).

#### 4.1.4.5 Bulk\_Sending\_Configuration / Exclude\_List

Table of e-mail records.

Specifies an exclusion list of addresses that is directly stored in the program's database (i.e. in contrary to specifying the files from which excluded addresses are extracted – see Section 4.1.4.2).

When the final list of recipients is prepared, recipients extracted from the files are added to recipients that are in the recipient list stored in the database, and finally addresses that are contained in the joint exclusion list (addresses extracted from exclusion files plus those listed directly in the database) are removed from the list. This is done before the e-mails are sent by the *Send* command (Section 4.2.1.9).

#### 4.1.4.6 Bulk\_Sending\_Configuration / Recipient\_Field

Integer parameters.

Specified under which field of the sent e-mail messages the recipients are listed:

- 0 – “Bcc”
- 1 – “To”
- 2 – “Cc”

This concerns only the addresses that are contained in the cleaned list of recipient (e.g. with addresses from the exclusion list removed, and prepared according to other parameters defining appearance, order and other things). It does not affect static addresses that are always added to the specified fields (see Sections 4.1.3.4, 4.1.3.5 and 4.1.3.6).

If you don't want recipient of your message to see who else are the recipients, then the recipients must be listed in the “Bcc” field.

This parameter affects sending with the *Send* command (Section 4.2.1.9).

#### 4.1.4.7 Bulk\_Sending\_Configuration / Random\_Order\_of\_Recipients

Integer parameter.

If non-zero then recipients of the e-mail will be listed in random order.

Otherwise (if the parameter value is 0), the recipients will be listed either by some internally specified order (e.g. by alphabet) or by the order in which they were added to the recipient list. The latter can be true only when there is no exclusion list.

#### 4.1.4.8 Bulk\_Sending\_Configuration / Remove\_Duplicate\_Recipients

Integer parameter.



If non-zero then duplicate e-mail addresses are automatically removed from the list of recipients.

#### **4.1.4.9 Bulk\_Sending\_Configuration / Include\_Recipient\_Names**

Integer parameter.

If non-zero then the display names are included in the recipient field together with e-mail addresses. For example, some address may appear as

*"harry.potter@whichcraft.com"*

when the value of this parameter is 0 (or when the parameter is not specified), or as

*"Harry Potter <harry.potter@whichcraft.com>"*

When the value of the parameter is different than 0.

##### **Note:**

When e-mail addresses for recipient lists are extracted from files, display names may not be extracted correctly. From this perspective, it is safer to set this parameter to 0 because it may be inconvenient if somebody receives a mail where his name is stated wrong, for example

*"This is the address of the student <harry.potter@whichcraft.com>"*

#### **4.1.4.10 Bulk\_Sending\_Configuration / Maximal\_Number\_of\_Recipients\_per\_Mail**

Integer parameter.

Specifies the maximal number of recipients included in sending of a single mail, in the case that sending is done in groups rather than to all recipients at once.

Note that this may be important also because some SMTP servers may reject e-mails that are sent to too many recipients. When an e-mail is sent by the *Send* command (Section 4.2.1.9) and there are many recipients, the mail is sent to smaller groups of recipients one after another.

##### **Warning:**

Recipients that are specified statically for given address field of e-mails are not taken into account in this number (Sections 4.1.3.4, 4.1.3.5 and 4.1.3.6))

#### **4.1.4.11 Bulk\_Sending\_Configuration / Number\_of\_Recipients\_per\_Mail**

Integer parameter.

Specifies the number (or average number) of recipients included in sending of a single mail. Actual number of recipients per e-mail is always limited by the maximal number (Section 4.1.4.10).

When there are many recipients, sending is done in groups rather than to all recipients at once (but you can change this by specifying a very large number of recipients and maximal number of recipients).

##### **Warning:**

Recipients that are specified statically for given address field of e-mails are not taken into account in this number (Sections 4.1.3.4, 4.1.3.5 and 4.1.3.6))

#### **4.1.4.12 Bulk\_Sending\_Configuration / Deviation\_of\_Number\_of\_Recipients\_per\_Mail**

Integer parameter.

Specifies the maximal deviation of number of recipients included in sending of a single mail. Actual number of recipients per e-mail is always limited by the maximal number (Section 4.1.4.10) and bounded below by 1.

If all mails should be sent to the same number of recipients (except perhaps the last one, which is sent to the remaining number of recipients) then this parameter should be set to 0.

#### **4.1.4.13 Bulk\_Sending\_Configuration / Maximal\_Pause\_Between\_Sends\_in\_Seconds**

Integer parameter.

Specifies the maximal pause length (in seconds) between sending consecutive e-mails (in a series of e-mails that are sent to the divided groups of recipients) by the *Send* command (Section 4.2.1.9).

#### **4.1.4.14 Bulk\_Sending\_Configuration / Minimal\_Pause\_Between\_Sends\_in\_Seconds**

Integer parameter.

Specifies the minimal pause length (in seconds) between sending consecutive e-mails (in a series of e-mails that are sent to the divided groups of recipients) by the *Send* command (Section 4.2.1.9).

#### **4.1.4.15 Bulk\_Sending\_Configuration / Pause\_Between\_Sends\_in\_Seconds**

Integer parameter.

Specifies the pause length or average pause length (in seconds) between sending consecutive e-mails (in a series of e-mails that are sent to the divided groups of recipients) by the *Send* command (Section 4.2.1.9).

#### **4.1.4.16 Bulk\_Sending\_Configuration / Deviation\_of\_Pause\_Between\_Sends**

Integer parameter.

Specifies the maximal deviation of the pause lengths (in seconds) between sending consecutive e-mails (in a series of e-mails that are sent to the divided groups of recipients) by the *Send* command (Section 4.2.1.9).

If you want that all pauses between consecutive sends are of the same length then the value of this parameter should be 0 (or it should be undefined). Otherwise, random pause length in the prescribed limits will be taken.

### **4.1.5 Definition of e-mail client and its configuration**

The default e-mail sending program is the *sendEmail* command-line SMTP client. However, any other command-line SMTP client can be used. The client and its use are completely defined by a set of parameters that are stored in the database. When the program is run, these parameters are initialized in such a way that the *sendEmail* program is utilized as the e-mail client. In order to change this, the e-mail client parameters should be set accordingly. This can be done by executing the *ReadParameters* command (Section 4.2.1.8) and choosing the appropriate subcommand for e-mail client configuratin.

The e-mail sending client (which performs the actual sending of e-mails) is invoked by executing the appropriate system command (a standard way of running a program). The command is handled by the operating system and can include a number of arguments. These arguments are used to specify all the relevant parameters to the e-mail client. The e-mail client is capable of recognizing different kinds of parameters that must be specified by command-line arguments in the

---

required way. In order to correctly interpret the meaning of parameters, their values must be introduced by the special option strings that are recognized by the program.

For example, the *sendEmail* program can be called by the following command (on Unix; in MS Windows, command can not be divided across multiple lines by appending ‘\’ at the end of each line):

```
sendEmail -f myaddress@isp.net \
          -t myfriend@isp.net   \
          -s relay.isp.net      \
          -u "Test email"       \
          -m "Hi buddy, this is a test email."
```

In the above command, strings *-f*, *-t*, *-s*, *-u* and *-m* are option specifications that tell the program about the meaning of the following argument. For example, command-line arguments “*-f myaddress@isp.net*” instruct the program (e-mail sending client) that the “*From*” field (i.e. the sender’s address) of the message should be set to *myaddress@isp.net*, and it is the option string (argument “*-f*”) that informs the program about the meaning of the corresponding parameter (argument “*myaddress@isp.net*”).

Below is the list of relevant parameters.

#### 4.1.5.1 Client\_Configuration / Email\_Client\_Program

String parameter.

Name of the e-mail client program. It must represent a valid program name under which the operating system can execute that program (you can try to call the program manually before specifying this parameter).

Program name can include the absolute path to the program, for example:

```
/usr/bin/sendEmail
```

(Unix-like) or

```
C:\Program Files\misc\sendEmail.exe
```

(Windows-like). In the cases similar to this latter one, please check that the path name containing spaces is interpreted correctly, otherwise try to move the program elsewhere and get rid of the spaces.

Note that you do not need so specify the complete absolute path of the program’s executable in the case that it is located in a directory that is included in the system path.

#### 4.1.5.2 Client\_Configuration / Option\_Separator

String parameter.

A separator that separates individual command-line arguments (options). Usually a space (“ ”) is OK, and this is also the default (used in the case that the parameter is not specified).

#### 4.1.5.3 Client\_Configuration / Address\_Separator

String parameter.

A separator that separates individual addresses in the recipient list transferred by command-line arguments. Usually a space (" ") is OK, and this is also the default (used in the case that the parameter is not specified).

#### **4.1.5.4 Client\_Configuration / Email\_Client\_Success\_Code**

Integer parameter.

The return code returned by the client in the case of successful execution (without errors). If you are not sure, you can leave this parameter unspecified or set it to 0.

#### **4.1.5.5 Client\_Configuration / Option\_Head\_From**

String parameter.

Option string for the "From" field of the message (sender's address).

#### **4.1.5.6 Client\_Configuration / Option\_Head\_ReplyTo**

String parameter.

Option string for the "Reply-to" field of the message (address used to reply to the message).

#### **4.1.5.7 Client\_Configuration / Option\_Head\_To**

String parameter.

Option string for the "To" field of the message (addresses of recipients).

#### **4.1.5.8 Client\_Configuration / Option\_Head\_Cc**

String parameter.

Option string for the "Cc" field of the message (meaning "Carbon copy" - addresses of recipients to whom a copy of the message is also sent).

#### **4.1.5.9 Client\_Configuration / Option\_Head\_Bcc**

String parameter.

Option string for the "Bcc" field of the message (meaning "Clank carbon copy" - addresses of recipients to whom a copy of the message is also sent but are not visible to other recipients).

#### **4.1.5.10 Client\_Configuration / Option\_Message\_Subject**

String parameter.

Option string for the subject of the e-mail (the "Subject" field of the message).

#### **4.1.5.11 Client\_Configuration / Option\_Message\_Body**

String parameter.

Option string for the body (main text) of the e-mail message (in the case that it is directly specified as command-line argument).

#### **4.1.5.12 Client\_Configuration / Option\_Message\_Body\_File**

String parameter.

Option string for the file containing the body of the e-mail message.

**4.1.5.13 Client\_Configuration / Option\_Message\_Attachments**

String parameter.

Option string for attachments – files attached as whole to the e-mail message.

**4.1.5.14 Client\_Configuration / Option\_Message\_Headers**

String parameter.

Option string for eventual additional headers of the e-mail message that are specified in addition to the standard headers.

**4.1.5.15 Client\_Configuration / Option\_Outgoing\_SMTP\_Server**

String parameter.

Option string for the outgoing mail SMTP server used to relay the e-mails.

**4.1.5.16 Client\_Configuration / Option\_Network\_Timeout**

String parameter.

Option string for the timeout of network operations performed by the e-mail client.

**4.1.5.17 Client\_Configuration / Option\_Outgoing\_SMTP\_User**

String parameter.

Option string for the outgoing mail SMTP server *user*.

**4.1.5.18 Client\_Configuration / Option\_Outgoing\_SMTP\_Password**

String parameter.

Option string for the outgoing mail SMTP server user's *password*.

**4.1.5.19 Client\_Configuration / Option\_Sending\_Log\_File**

String parameter.

Option string for the e-mail client log file.

**4.1.5.20 Client\_Configuration / Option\_Verbose**

String parameter.

Option string for requesting verbose operation of the e-mail client.

**4.1.5.21 Client\_Configuration / Option\_Quiet**

String parameter.

Option string for requesting quiet operation of the e-mail client.

**4.1.5.22 Client\_Configuration / Option\_Additional**

String parameters.

Literal representation of eventual additional options that are always appended to the command-line string when the e-mail client is called.

### 4.1.6 Action list

#### 4.1.6.1 Action\_Configuration/ Command\_List

List of commands to be executed when the command *RunList* is executed (Section 4.2.1.7).

## 4.2 List of commands

### 4.2.1 General commaneds

The commands listed in this Section are available in interactive as well as in non-interactive operation. For some commands, a string argument can be specified. In this case, the argument (if specified) must be stated in round brackets without any spaces between the argument and the bracket and without quotes. For example, if the argument is stated as (f.xml), this will produce a string "f.xml", but if it is stated as ( f.xml ), this will produce a string " f.xml " (and probably the wrong file name). This may be subject to change in the future.

When the argument is not specified, the corresponding data is taken from the database (if stored there) or in some cases the user is asked to input it (with respect to this, commands may behave differently in interactive and non-interactive mode).

Commands are triggered by similar strings in interactive and non-interactive mode (in the latter, commands are simply listed as command-line arguments in the order of execution). However, there are two small differences. First, in interactive mode, command names can be replaced by the consecutive numbers from the command list (command IDs), which is not possible in non-interactive mode. Another difference is that in the non-interactive mode, the command string (command with eventual argument) must not contain any spaces (e.g. between the command and parentheses that contain a command argument), or it must be put in quotes. This is because strings containing spaces are treated as a set of separate space-delimited command-line arguments by the operating system, which executes the program and passes it the command-line arguments. In the non-interactive mode, there can be any number of spaces between the command and the eventual parentheses containing its argument.

#### 4.2.1.1 ExportData ( <filename> )

Export the complete program database to the file specified at the location  
*Program\_Configuration/Export\_File* .

#### 4.2.1.2 ImportData ( <filename> )

Imports program configuration data from the file whose name is specified in the database at the location

*Program\_Configuration/Import\_File\_List* .

This function re-sets those parameters in the program database that are defined in the configuration file and leaves other data unchanged (this is the difference with *RestoreSession*, Section 4.2.1.4).

#### 4.2.1.3 SaveSession

Saves the complete program's database to the file whose name is specified in the database at the location

*Program\_Configuration/Session\_File* .

This is in fact very similar to *ExportData*, except that the location of the default export file name in the database is different and is considered more static (i.e. the user is not expected to modify it). Therefore, this function is provided as a pair to *RestoreSession* (which is considerably different from the *ImportData*, is a pair to *ExportData*).

#### 4.2.1.4 RestoreSession

Restores the program database by reading it from the file whose name is specified in the database at the location

*Program\_Configuration/Session\_File* .

Any eventual data in the database are destroyed, therefore the state of the program is reverted to the state at the moment of execution of the corresponding *SaveSession* command, except for a few things that are eventually not stored in the database (one such example is the password input by the user, which is in some cases not stored in the database).

#### 4.2.1.5 RunCommand

Runs a single command. The command (and eventually its data) is obtained interactively by user input.

#### 4.2.1.6 Interactive

Performs an interactive run of commands.

#### 4.2.1.7 RunList

Runs the list of commands that are stored in the database at the location

*Action\_Configuration/Command\_List* .

#### 4.2.1.8 ReadParameters

Performs interactive input of configuration parameters. Parameters can be input all at once or in individual groups.

#### 4.2.1.9 Send

Performs sending of e-mails according to parameters in the program database.

#### 4.2.1.10 TestSend (<recipient>)

Performs a test send. Instead of using the actual recipient list, the mail is just sent to the test address. The test address must be a valid string for the address field. Its place in the database location is

*Sending\_Configuration/Test\_Recipient* .

The test address can also be specified as an argument *recipient* of the command.

## 5 ADDITIONAL INFORMATION

### 5.1 *FAQ (Frequently Asked Questions)*

#### 5.1.1 What is an incoming SMTP e-mail client?

An outgoing SMTP e-mail client is a program that can send out e-mails. In order to do that, the e-mail client must connect to an outgoing mail SMTP server (see question 5.1.2). Therefore, an e-mail client requires the address of the SMTP server (either as IP address or server name, such as smtp.gmail.com) as a part of its configuration (see question 5.1.3).

Thunderbird and Outlook are common example of e-mail clients. These programs also perform a number of other functions such as retrieving incoming messages or archiving the messages.

The SendIgence program needs a command-line incoming SMTP client that can be run non-interactively, in such a way that all necessary parameters are provided via command line arguments. By default, the the sendEmail is used for this purpose, which is freeware that can be simply [obtained on the Internet](#), but is also included in the precompiled SendIgence packages.

#### 5.1.2 What is an outgoing mail SMTP server?

An outgoing mail SMTP server is a software that relays e-mail messages such that they can reach the intended recipients. The term also refers to a computer on the Internet that runs such a software.

SMTP stands for **Simple Mail Transfer Protocol**, which is a standard for e-mail transmissions across the Internet.

#### 5.1.3 What is necessary to relay e-mails?

In order to send e-mails, the e-mail client (question 5.1.1) must establish a connection with an outgoing SMTP server (question 5.1.2) that relays the messages. The e-mail client must run on a computer connected to the Internet, and it must know the server address and eventual account parameters (i.e. the user name and password) if these are required by the server.



One of the ways to obtain these parameters is to use your usual account settings that you use for receiving mail. The server name and user name (or login name) can usually be retrieved from the server settings in an e-mail client (such as MS Outlook or Mozilla Firefox).

## **6 DEVELOPMENT**

### ***6.1 Version history***

#### **6.1.1 Development of versions in reverse chronological order**

##### **6.1.1.1 Version 1.2**

- Bulk sending functionality expanded
- A special command block is implemented for input of parameters
- Reading of parameters is further separated in categories
- Enabled random pause length between successive sends
- Error reporting recorded in log file (when defined)
- Cleaning of code, some bugs removed

##### **6.1.1.2 Version 1.1**

- Added random sorting of recipients
- Enabled file attachments
- Enabled random number of recipients per mail
- Added possibility of reporting of program operation to a log file; reporting is done if the log file is specified, and it is not specified by default (at program initialization)

##### **6.1.1.3 Version 1.0**

- Recipients list acquired from a file is combined by a manually specified recipient list
- User input of parameters is divided by categories
- Non-interactive function is enabled
- Unknown command functionality is added
- When triggering commands, names of configuration files cause reading of configuration data
- Added random sorting of recipients

Added logging for e-mail client  
Implemented test e-mail composition

#### **6.1.1.4 Version 0.5**

Support to other e-mail sending clients is added  
Command-line options can be manually specified and are stored in the database. The sendEmail configuration is set up at the beginning

#### **6.1.1.5 Version 0.4**

System of commands implemented  
A list of address files can be specified  
Configuration data can be imported from multiple files during the session  
Bulk sending options added

#### **6.1.1.6 Version 0.3**

Database in XML format added  
A whole session can be exported and imported

#### **6.1.1.7 Version 0.2**

Specifying multiple address files by using wildcats  
Exclusion lists enabled  
User can choose the message field where recipient addresses are put (To, Cc or Bcc)

#### **6.1.1.8 Version 0.1**

Automatic exclusion of double repeating addresses.  
Improved acquisition of addresses from files (a wide range of text formats can be read).  
Message body can be read-in from a file.

#### **6.1.1.9 Version 0.0**

Can read recipient addresses from a file in a special format, addresses put to Bcc field, one To address can be specified manually, all mails sent at once, using the sendEmail client. Message subject and body are specified as strings.

## ***6.2 To do (wish list)***

Executing nested commands, such as ReadParameters/Recipients. This should be implemented by the unknown function handler, which would parse the nested command and would

---

execute the final command (In this case, Recipients) in the subgroup of commands, which would be represented on its own stack. Yet a better option is to have a special command (named e.g. "Nested") on each command stack, with a default command data set to a stack of sub-command stack, and this command would then handle such nested commands.

Address center for handling address lists (databases). It could be implemented within the Read (or Readparameters) group.

- Exporting of addresses to a separate file (exclusion as well as recipient addresses)
- Implement migrating lists of recipient addresses between the database and a file

Possibility of defining custom header fields (list of fields!)

Inclusion of bulk option (Gmail directives): either initialize in such a way that the bulk option is included, or add this possibility as an additional option in input functions

Log files for bulk sending

- logs are appended
- before every send, information about time and number of recipients is printed
- after every send, the complete command is printed (such that addresses to which the mail was actually sent can be extracted from the log file)

Character encoding instruction to mail client



