

# **The Bat!**

## **Ritlabs AuthenticBat!**

## **Ritlabs SecureBat!**

### **Help System**

This help system contains reference information for The Bat!, Ritlabs AuthenticBat! and Ritlabs SecureBat!

**The Bat!** is a powerful, highly configurable, yet easy to use email client. It has been specially designed to help users deal with the growing volume of email as quickly and efficiently as possible, saving much of the user's precious time. The Bat! is a great email client with every feature that an advanced user needs, yet with an interface that even a novice will love. See the [General Information](#) topic for a more detailed description of The Bat!

**Ritlabs AuthenticBat!** is an e-mail client which offers all of the major features of The Bat! plus [secure authentication](#) on POP3/SMTP servers using [iKey](#) tokens. The format of the Ritlabs AuthenticBat! message base, address books and configuration files is fully compatible with The Bat!, which permits the transparent upgrade of The Bat! to Ritlabs AuthenticBat! simply by replacing the program file and configuring the iKey-related settings.

**Ritlabs SecureBat!** is an e-mail client which offers all of the major features of The Bat! plus [secure authentication](#) on POP3/SMTP servers using [iKey](#) tokens and transparent, on-the-fly encryption of the message base, address books and configuration files using the [SecureBat!ID](#). Upgrade from The Bat! or Ritlabs AuthenticBat! can be made by backing up all of the data using the Maintenance Centre of The Bat! or Ritlabs AuthenticBat! and then restoring the data from the Maintenance Centre of the Ritlabs SecureBat!. The potential area of use for Ritlabs SecureBat! is wide: from big corporations to small offices, home offices or private computers, where the role of the Security Officer is taken by the user themselves. iKey tokens for Ritlabs SecureBat! are configured by a Security Officer using a companion utility: **Ritlabs SecureBat! Token Manager**.

To avoid overhead, in the help system, Ritlabs AuthenticBat! and Ritlabs SecureBat! are referred as "The Bat!", except where explicitly stated. Everything written as for "The Bat!" also applies for Ritlabs AuthenticBat! and Ritlabs SecureBat! except where explicitly stated.

## Legal Information

Information in this help system is subject to change without notice and does not represent a commitment on the part of RITLABS S.R.L. The software and/or files described in this help system are furnished under a license agreement. The software and/or files may be used or copied only in accordance with the terms of the agreement. No part of this online help system may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording, or information and retrieval systems, for any purpose other than the purchaser's personal use, without the written permission of RITLABS S.R.L.

RITLABS S.R.L. may have patents or pending patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. The furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property rights except as expressly provided in any written license agreement from RITLABS S.R.L..

All attempts have been made to make the information in this help system complete and accurate. RITLABS S.R.L. is not responsible for any direct or indirect damages or loss of business resulting from inaccuracies or omissions. The specifications contained in this help system are subject to change without notice.

Mail Ticker™ is a trademark of RITLABS S.R.L.. iKey is a trademark of Rainbow Technologies, Inc. Microsoft Windows, Microsoft Windows NT, Windows 95, Windows 98 and Windows 2000 are trademarks of Microsoft Corporation. All other product names referenced herein are trademarks or registered trademarks of their respective manufacturers.

Copyright © 1997-2001 RITLABS S.R.L.. All rights reserved.

*Using The Bat! as a Mailing List Server* chapter copyright © 1998-1999 by Leif Gregory. All rights reserved.

*Regular Expressions Syntax (Advanced)* chapters are written by Philip Hazel and copyright by the University of Cambridge, England.

Topics related to Ritlabs SecureBat! and Ritlabs AuthenticBat! are written by RITLABS S.R.L. and edited by Marck D. Pearlstone, copyright © 2001 RITLABS S.R.L.. All rights reserved.

## Internet

The Internet is a global computer network, covering the entire globe. The Internet evolves as if it is a core, ensuring the correct interrelationship of different information networks, belonging to various organisations around the world.

The first networks were used solely for the transmission of files and e-mail, but today it resolves the more complex problems of shared access to resources. Several years ago, programs were created to support the necessary functions of network search, and access to shared information resources, and electronic archives.

The Internet, was initially only used by research groups and educational establishments whose interests spread up to the access to supercomputers. These resources became all very popular in the business world, and in no time the Internet has grown phenomenally

Big companies were seduced by this quick and easy global connection, well suited to group-work, centralised programs, and a single database over the Internet networks. They considered a global network as an addendum to their own local networks. With the low service costs (often only a fixed monthly charge for lines used, or telephone charges) users are able to gain access to commercial and non-profit information services from across the USA, Canada, Australia and many European, and other countries. Using the Internet it is possible to find information on practically any sphere of human activity, in the "free access" archives, from new scientific openings, to the weather forecast for tomorrow.

Besides all this, the Internet offers a unique possibility of cheap, reliable and confidential global communication with the whole world. This turns out to be extremely efficient for companies having various sites around the world, international corporations, and other organisations. Usually, using an Internet infrastructure for all international communications is vastly cheaper than direct computer communication through a satellite link or telephone lines.

E-mail is the most widespread service on the Internet. Sending e-mail messages over the Internet is vastly cheaper than more conventional means. Besides, messages sent on the Internet will arrive in several hours, while other messages (snail mail) can get to the addressee several days later, or even several weeks later.

## General Information

The Bat! is an Internet e-mail program. Major objectives of The Bat! are:

- Providing an efficient way of processing large message flows;
- Easy-to-use user interface;
- Allowing users to work easily with several mail accounts with the possibility of exchanging messages between them;
- Maximum reduction of delays while working. The Bat! extensively uses Window's multitasking capabilities. This means that the user can send and receive e-mail while reading or writing other mail;
- Providing truly multi-lingual interface. The Bat! can communicate with the user using various languages
- the interface's language can be chosen from the main program menu.

The Bat! also allows the user to process mail right on the mail server - without actually downloading messages to the local hard drive. The Bat! has a comfortable editor with many useful functions, Address Book and provides extensive set of tools for processing mail. When connecting to POP3/SMTP servers, the users of Ritlabs AuthenticBat! or Ritlabs SecureBat! can use hardware authentication based on the iKey token.

The program has been developed mainly for business messaging, which is why we tried to provide as comfortable an interface as possible, without messing the workplace up with excessive interface elements used rarely or for entertainment purposes.

Major features of The Bat! are:

- Real-time automatic spellchecker. As you type, it underlines questionable spellings, and a single right click will let you choose the correct spelling.
- Full 32 bit multi-threading. You can check mail on all of your accounts, and write/read messages at the same time.
- Fast, browser-independent HTML Mail viewer.
- Quick Templates allow you to save hours of typing and numerous keystrokes by inserting text blocks (even from files on your disks) and message-dependent information as well as attaching files and vCards without using menu commands.
- Threading messages by reference, subject, sender and recipient. This feature is especially handy for subscribers of discussion mailing lists.
- Mail Ticker™ is a feature seen nowhere else. It looks like a Wall Street stock ticker, but it shows you the header info from your new messages - From, To, Subject etc..
- Powerful Filtering System. Pass data to external programs, auto-responders, macro enhanced templates for auto-responders, kill filters etc..
- Filters and Macros are so powerful that they allow you to turn the e-mail client into a mailing list server. For example, you can run mailing lists as well as full-fledged discussion lists like MajorDomo and Listserv. All you have to do is check the POP3 account, and it will filter, modify, correct the To:, From: and Reply To: headers, then mails it out to everyone on the list. One of lists running this way is called "The Bat! User's Discussion List" (see home page).

- Internal OpenPGP implementation based on the award-winning OpenSSL library. Allows you to encrypt messages, sign them with digital signatures, etc. Handy manager of digital keys. Plus free plug ins for PGP v5.5 & v6.0.2.
- Multiple accounts that can be changed on the fly. There's more, each folder you create to save messages in can have it's own identity (To, From, Reply-to), macro enhanced templates etc.. This is my most favourite aspect of The Bat! You can make it do just about anything.
- Automatic mail retrieval w/auto log-off
- It can import your messages with the built in import wizard from:
  - Microsoft Outlook Express v4.xx
  - Netscape Communicator v4.xx
  - Netscape Mail v2.xx/3.xx
  - Eudora Lite/Pro
  - Pegasus Mail v2.xx
- It can import address books from any client that can export to an:
  - .INI
  - Comma Separated File
  - Tab Separated File
- Support of ten different interface languages, with the capability to switch between them on the fly.
- An unique park function, where you can make a message stay where it is at. No accidental deletions or moving it to the wrong folder.
- An address book function that let's you choose e-mail addresses very similar to the way you run programs from the Windows Start button or choose favourites from the favourites dropdown menu. The address book also supports vCard files and LDAP directories as well as being able to place actual pictures of your contacts in the address book. Another feature is the ability to define macro enabled templates for \*each\* contact in your address book. That means you can personalise your messages for each individual person.
- A built in search engine displays your results in a logical, easy to use format.
- Mail Dispatcher allows you to manage your mail directly on the POP3 server.
- Provides an easy and cheap corporate mailing network server solution (multi-user capabilities), that allows to work without a local mail server.
- Submission Forms - a tool for creating queries that should be automatically processed. A great solution for business client-server based environment.
- Dial-up networking support, mail notification, message actions, live URLs, nested folders, smart drag & drop, and more!

# Connection to The Internet

The Bat! supports two ways of connecting to the Internet

## Using Local Area Network or manual connection

If this way is chosen, The Bat! uses the existing connection established either with your LAN or by dialling your Internet Service Provider. In other words, The Bat! relies on that the connection to the Internet is established and the program itself does not need to do something extra. Some non-LAN systems may be configured so that ISP is dialled automatically whenever an Internet program is trying to connect to an Internet server. This can be used with The Bat! However, in some cases The Bat! needs to two TCP/IP sessions simultaneously and this may cause problems such as trying to establish two dial-up connections at the same time.

## Dial-Up Networking Connection

You need to specify this type of connection if your LAN is not connected to the Internet itself and you are using Dial-up connection provided by your ISP. Whenever The Bat! need to open TCP/IP connection, it checks whether the specified 'phone book entry is active (i.e. the established TCP/IP connection, if there is any, is done by using this 'phone book entry). If the entry is active, TCP/IP sessions starts. Otherwise, The Bat! will automatically dial the 'phone book entry to establish TCP/IP connection. In this case, multiple requests such as checking mail for all accounts and combined delivery (which requires two TCP/IP connections to be open at once: one for mail retrieval and another one for sending mail) will be handled correctly.

**Auto-disconnect after mail transmission** - Use this option if you do not want to remain on-line when The Bat! finishes dealing with mail transmission. This feature is almost useful when you are using periodical mail retrieval (especially when you have to pay for call to your ISP).

**Pause between dialling session** - This value determines the minimal time interval between two consecutive dialling sessions. Using of this feature is essential in multi-user environment with only one dial-out machine - it prevents the dial-out machine to dial the ISP unnecessarily often. Note that manual requests of mail retrieval from dial-out machine will initiate dialling even in the pause time

## Your machine status

If a machine is not a part of a Local Area Network, or all machines in your network have TCP/IP protocol installed, or you simply do not intend to read your mail from other machine within your LAN, the status of your machine is *TCP/IP Workstation*.

If a machine is connecting to the Internet by either LAN or Dial-up networking and it is intended to retrieve and send mail for all accounts defined in The Bat!, the status of your machine is *TCP/IP or Dial-out Server*.

If a machine does not have TCP/IP protocol installed and it is a part of a network with TCP/IP or Dial-out Server, it must have status of *Non-TCP/IP Workstation*.

See also:

[Working in Multi-User Environment](#)  
[The Bat! Networking Course](#)

## Logging on to a POP3 Server

POP3 (Post Office Protocol, revision 3) - the “post office” protocol is for retrieving mail from a host mail server using client software.

Once you have an e-mail address, your ISP (Internet Service Provider) should give you a user name (which is usually the same as the part of your e-mail address before the “@” symbol), a password to access your mailbox on the mail server and the address of the POP server. This should look something like “mail.host.com”.

This information should be entered in the account properties, in the appropriate fields. The password may be left blank if you want to enter your POP password every time you check for new mail.

The Bat! has the ability to securely authenticate a client with a POP3 Server. This avoids the need to pass a cleartext password over a network, keeping the information private. The users of Ritlabs AuthenticBat! or Ritlabs SecureBat! can use hardware authentication based on the iKey token.

## POP3 Authentication

POP3 authentication is configured in a special window, which is invoked from the Account Properties -- Transport tab -- Receive Mail – Authentication button. The Bat! supports four authentication schemes, ranging from the least to the most secure. The users of Ritlabs AuthenticBat! or Ritlabs SecureBat! can use hardware authentication which is the most secure mechanism.

**Regular** - this is a simple authentication mechanism that requires an e-mail client to send a username and password to the server. Username/password data is sent in cleartext form, which could be easily intercepted in transit. This authentication mechanism is supported by most RFC-1081 compliant mail servers, e.g. mail servers that support basic POP3 features.

**MD5 & APOP Challenge/Response (RFC-1939)** - this authentication mechanism avoids passing a cleartext password over a network. Instead of sending the password as cleartext, the e-mail client sends a non-reversible digest (produced by the MD5 cryptographic hash function) of the password concatenated with a unique random string (Challenge String) received from the server. Any exposure of the digest that is sent during the APOP authentication cycle does not introduce a risk, even for e-mail clients that connect frequently to POP3 servers to check for new mail, e.g. every five minutes. Please note that this authentication mechanism may not be supported by all POP3 servers.

**MD5 & CRAM-HMAC Challenge/Response (RFC-2095)** - this authentication mechanism is an improvement over the APOP standard. It also follows a Challenge/Response scheme, but uses HMAC (Keyed-Hashing) instead of the simpler digest method. While APOP requires that both the client and server systems have access to the password in cleartext form, HMAC offers a method for avoiding such cleartext storage while retaining the algorithmic simplicity of APOP in using only MD5, though in a "keyed" mode. Another reason to choose Keyed Hashing is the greater security imparted to the authentication of short passwords. Please note that this authentication mechanism may not be supported by all POP3 servers.

**iKey MD-5 CRAM-HMAC Challenge/Response** (available in Ritlabs AuthenticBat! and Ritlabs SecureBat! only) - this is a hardware implementation of the CRAM-HMAC Challenge/Response (RFC-2095) authentication. A special non-replicable hardware token, iKey1000 by Rainbow Technologies, is used to store the password and to produce the Keyed Hashing. When this authentication mechanism is used, the password will never be exposed at the client side. Once stored, the password cannot be extracted from the token and it is never transferred into the computer where the e-mail client is running. This way, no software (including Spies / Trojan Horses / Viruses) can intercept or otherwise retrieve the password. A mail server administrator may give the user an iKey token that has already had the required password stored on it, so the user won't know and won't need to know the actual password. Utilising the feature that iKey tokens can not be replicated, only the physical owner of the token will have access to the mail server, provided that he or she knows the iKey PIN. All POP3 servers that support the MD5 & CRAM-HMAC Challenge/Response (RFC-2095) authentication method support this authentication mechanism. Please note that iKey MD-5 CRAM-HMAC Challenge/Response authentication mechanism is available in Ritlabs AuthenticBat! and Ritlabs SecureBat! only.



## Choosing an SMTP Server

SMTP (Simple Mail Transfer Protocol) - is a simple protocol for mail transmission that is widely used on the Internet. The SMTP server's function is to receive inbound mail from other servers and clients, and to deliver outbound mail to other hosts and to its clients' mailboxes.

Usually, the address of an SMTP server is provided to you by your system administrator or Internet Service Provider. In the early years of Internet, a user might use other SMTP servers available on the Internet known to him/her, since SMTP did not require any special authorisation. Nowadays, most SMTP servers require authorisation based on either the IP address of the client, the username/password or some other form of authentication.

Authorisation based on IP address does not require any special settings on the client. A server just checks what point of the Internet the client has connected from, and, if this point is within the allowed IP address range, the connection continues, and the server does not restrict the client, allowing it to send messages to any destination. Otherwise, the server may require a username/password or another authentication method, and, if the client was still not authenticated then its sending capabilities would be limited, if not completely rejected. The users of Ritlabs AuthenticBat! or Ritlabs SecureBat! can use hardware authentication based on the iKey token.

In conclusion, in most cases, your ability to choose SMTP servers is limited and you will have to use the SMTP server provided by your system administrator or Internet Service Provider.

## SMTP Authentication

SMTP authentication is configured in a special window that is invoked from the Account Properties -- Transport Tab -- Send Mail -- Authentication button.

Unlike POP3 authentication, SMTP authentication is optional.

The Bat!, depending on server configuration, can either authenticate according to protocols defined in RFC-2554, and/or simply logging on to a POP3 server just before initiating an SMTP session (a.k.a POP before SMTP authentication). If you are unsure of what kind authentication to choose, you can either not use SMTP authentication at all, or contact your system administrator or Internet Service Provider to find out which you really need.

Using RFC-2554, SMTP authentication involves a username and a password being transferred from the client to the server. If the username and the password for SMTP authentication are the same as for POP3 authentication, you can check “**Use Settings of Mail Retrieval**”. Otherwise, check “**Use Specific Settings**” and specify the username and the password.

Regardless of which settings you are using, either specific or those of mail retrieval - you can avoid sending the password as cleartext by checking the “**Require secure (MD5) authentication**” box. This will activate the CRAM-HMAC Challenge/Response authentication mechanism, which, however, may not be supported by all servers.

Please note that if you check “**Require secure (MD5) authentication**” and the server does not support this secure authentication mechanism, there will be no authentication at all, and the mail session will continue as unauthenticated. Under no circumstances will The Bat! pass a cleartext password over a network during RFC-2554 authentication when “**Require secure (MD5) authentication**” is on.

Once connected to an SMTP server, The Bat! checks which of the server’s RFC-2554-authentication mechanisms is available, and chooses the most secure. Even when “**Require secure (MD5) authentication**” is off, if the server does supports it, secure authentication will take place.

The secure (MD5) authentication mechanism avoids passing a cleartext password over a network, ensuring that it cannot be captured and used by anyone else. Instead of sending the password as cleartext, The Bat! client sends a non-reversible digest (produced by the MD5 cryptographic hash function), as defined in the HMAC (Keyed-Hashing) standard, of the password and a unique random string (Challenge String) as received from server. Even if the digest being sent is exposed during SMTP authentication, there is no risk involved, even for e-mail clients that connect frequently to SMTP servers to send new mail. Please note that this authentication mechanism may not be supported by all SMTP servers.

You may also choose the option to “**Store password on iKey**” (available in Ritlabs AuthenticBat! and Ritlabs SecureBat! only). This will activate a hardware implementation of the CRAM-HMAC Challenge/Response (RFC-2095) authentication. A special non-replicable hardware token, iKey1000 by Rainbow Technologies, is used to store the password and to produce Keyed Hashing. When this authentication mechanism is chosen, the password will never be exposed at the client end. Once stored, the password cannot be extracted from the token and it is never transferred into the computer where the e-mail client is running. This way, no software (including Spies / Trojan Horses / Viruses) can intercept or otherwise retrieve the password. A mail server administrator may give the user an iKey token that has already had the required password stored on it, so the user won’t even know and won’t need to know the actual password. Utilising the feature that iKey tokens cannot be replicated, only the physical owner of the token will have access to the SMTP server, provided that he or she knows the iKey PIN. All SMTP servers that support MD5 & CRAM-HMAC Challenge/Response authentication support this authentication mechanism. Please note that this authentication mechanism is available in Ritlabs AuthenticBat! and Ritlabs SecureBat! only.

## E-mail Address

On the Internet, each user has a unique address (some people have more than one e-mail address) which identifies the mail server or zone address, and the name of the mailbox. The usual structure of an e-mail address is <user name>@<mail server/zone>, e.g., john.smith@ritlabs.com. E-mail addresses cannot be anything like “John Smith” or “John Smith from Magic Island”

## Adding a New Entry to the Address Book

Invoke **Address Book** command on the **Tools** menu in the main program window, or click the appropriate button on the main window toolbar.

Invoke **New Address** entry in the **Edit** menu in Address Book.

Edit user's data, especially his name and e-mail address - all other fields are optional.

The new entry may be added to the Pop-up menu of address entry fields, if the check box Add to Pop-up menu is checked.

# Setting up an E-mail Account

Account settings are divided into the following sections:

- General settings
- Transport settings
- Advanced Transport settings
- Preferences
- Files and directories
- Message templates
- Notification of new mail arrival (sound)
- Sorting Office / Filters

Each e-mail account set-up in The Bat! has its own folder structure which is used for storing incoming, outgoing and sent messages. Other folders may have any name (depending on their use) but there always are four default folders, which cannot be deleted or renamed. These folders are:

<b>Inbox</b>	the default incoming mail folder
<b>Outbox</b>	the outgoing mail folder
<b>Sent</b>	default folder for sent mail
<b>Trash</b>	folder for storing deleted messages before they are removed permanently

## Account General Settings

This section is for setting up the title of an e-mail account, as it appears in The Bat!'s account list, and the data used for generating the sender's data in message headers.

**Name** - the name of the e-mail account. It may be any combination of characters. The only limitation is that the name must be unique within the program; in other words, there must not be another account with the same name

**From name** - the originator's name, which will be put in the header of each message created for the account

**From address** - originator's e-mail address

**Organisation** - organisation to which originator belongs (if any)

**Reply name** - the name of the person to which replies must be sent. Usually it is the same as the originator's name

**Reply address** - the e-mail address used for replies to the original e-mail. (return address, and the address to which confirmation must be sent)

### Notes:

Avoid using characters disallowed in system file names (such as “+, ”^><|”) in account’s name. Otherwise, non-default directory names for disk storage must be used.

Try to use only ASCII characters in From and Reply names - this will help to avoid problems with some mail servers which don't allow of non-ASCII characters in message headers.

## Account Transport Settings

**SMTP server** - the address of your SMTP server to which all outgoing e-mail messages from the account must be sent. The address may be represented as a traditional URL (e.g.: mail.ritlabs.com) or as an IP numeric address (e.g. 193.219.214.38); if an IP address is used you should be aware that sometimes it may be changed. The advantage of using of IP addresses is that a DNS lookup is not used, thus it is slightly faster on connecting to the server. Generally speaking, if there are several mail accounts set in the program, it is possible to use just one SMTP server for sending messages from all accounts. In some rare cases, depending of an SMTP server you've chosen, you may need to perform SMTP authentication. The users of Ritlabs AuthenticBat! or Ritlabs SecureBat! can use hardware authentication based on the iKey token.

**POP3 server** - the address of your POP3 server where the account's mailbox is located. All messages from the Internet are accumulated on POP3 servers and may be retrieved from there by the "Get new mail" command or by Message Dispatcher. The address may be represented as a traditional URL (e.g.: mail.ritlabs.com) or as an IP numeric address (e.g. 193.219.214.38); if an IP address is used you should be aware that sometimes it may be changed. The advantage of using of IP addresses is that a DNS lookup is not used, thus it is slightly faster on connecting to the server.

**User** - the POP3 user name for access to your mailbox on the server. Usually, it is the same as the part of the account's e-mail address before "@" symbol. For example, if the e-mail address is john.smith@ritlabs.com, the user's name will be john.smith

**Password** - the password for logging on to the POP3 server. You may leave this field blank - in this case you will be prompted to enter the password each time you try to receive mail.

**Authentication** – if logging to your POP3 server allows a secure authentication mechanism, you should configure it by pressing **Authentication** button that invokes a special window. The users of Ritlabs AuthenticBat! or Ritlabs SecureBat! can use hardware authentication based on the iKey token.

**8-bit characters are treated** - this radio group allows you to set the method used to handle 8-bit (non-ASCII) characters. If you are using only the English alphabet, this feature is not very significant to you because all English characters are ASCII characters and will go through the net without any problems. However, if you are using accented characters or character sets other than English, you should choose the method carefully because there still are some mail servers which do not allow appearance of 8-bit characters in e-mail messages. If you are not sure that such servers are functioning in the way you write your messages, it is better to choose either **Base64** or **Quoted-printable** encoding. The difference between these two encoding methods is that Base64 produces totally unreadable text. Quoted-printable encoding still can be read if your alphabet is almost Latin, only accented characters are encoded. If your recipient's are using The Bat! or another program which can recognise Base64 and quoted-printable encoding automatically, you may choose either of these two methods.

**Delivery** - the default delivery method. This is implemented almost for defining the function of "F2" and "Shift+F2" keystrokes in the message editor. If the delivery method is Immediate, pressing F2 within the editor will send the message immediately, whereas if Deferred is selected, the message will be queued in Outbox.

**Combined delivery** - if this check box is ticked, "Get new mail" command will also invoke sending of queued messages from Outbox and "Send queued mail" command will also invoke new mail retrieval.

**Always ask for POP3 password** - If this check box is ticked, you will always have to enter the POP3 password to retrieve new mail (this is equivalent to leaving POP3 password field blank).

# Advanced Account Transport Settings

Message management options and port numbers for servers are defined in this section

## Message management

**Delete received messages from server** - if this method is chosen, The Bat! does not leave copies of the received messages on the server. This may save some amount of disk storage of your provider (especially when your mailbox size is limited)

**Leave messages on server** - the opposite to the above method. This method is useful when you are accessing the server from several places and wish to be able to get a copy of any message from each of those places

**Keep messages on server for  $n$  days** - the maximum age of messages left on the server. When a message is older than the age stated, it will be automatically deleted from the server.

**Delete message from server when it is removed from Trash** - use this option if you are sure that you delete only messages not needed in the future.

**Receive header only if message size is greater than  $n$  K bytes** - use this option if you do not want to receive large messages from the account. This may be useful when you are away from your home and want to read only significant messages (messages of such kind are usually relatively short)

## Message Dispatcher

**Message lines to download with header** - this determines how many lines of the message text must be downloaded together with the message header to be viewed within Mail Dispatcher. This allows you to read a part of a message to decide whether you wish to download it this session

**Invoke automatically at each mail check** - if this check box is ticked, Mail Dispatcher will be invoked at each “Get new mail” command if there are new messages in the mailbox

**Show all messages left on server** - This option is available only if the previous option is set. If it is set, Mail Dispatcher will show all messages left on the server, otherwise only new messages will be shown

## Ports

Mail service port numbers can be changed sometimes to disallow other systems from the outside world to connect to servers used within the given network. In this case, you must re-define the port numbers.

SMTP	the port number used by <u>SMTP</u> protocol (default 25)
POP3	the port number used by <u>POP3</u> protocol (default 110)
IMAP4	the port number used by <u>IMAP4</u> protocol (default 143)



## Account Preferences

This section is designed for setting some miscellaneous parameters for a mail account.

**Check mailbox at startup** - if ticked, “Get new mail” command is invoked automatically each time The Bat! Starts

**Periodical checking each *n* minutes** - The Bat! will automatically examine your mailbox for new mail periodically, if this check box is ticked.

**Empty Trash folder on exit** - trash folder of the account will be automatically emptied each time you exit The Bat!

**Time of reading to mark a message as read** - within this time interval, unread messages will retain their status. As soon as an unread message is marked as read, some actions may take place (Read Mail sorting rules, generation of Reading confirmation)

**Maximum Log file size** - the account’s activity Log file is stored in the account’s home directory and called ACCOUNT.LOG. To prevent this file from getting too big, you may limit its size with this parameter

**Ignore “Check All Accounts” request** - use this option only if your account is not password-protected. It must not be checked for all your other mailboxes (for example, when POP3 server only works within a specific time interval)

## Account Files and Directories

**Home Directory** - the path to the directory where all files related to the given account are stored. This directory is also used for creating default sub-directories (each mail folder keeps its messages in MESSAGES.MSB file in its mail-folder directory, which by default is placed in the home directory of the account it belongs to)

### Attachment management

**Default encoding** - the default encoding type for attachments (used by the button on the toolbar of the message editor. It can be either **Base64** (MIME standard) or **UU-encode**

You may choose the storage method for received file attachments. They may be either **kept in a separate directory** (so you can copy them from there using explorer, or DOS. This method is the most preferable if you receive large files), or **in message bodies**. If you choose to store attached files in the *<default>* directory, files will be stored in the sub-directory named ATTACH in the account's home directory.

Option **Delete attached files when the message is deleted from Trash** makes The Bat! keep your disk storage in order, so you don't keep file attachments to messages you no longer need.

## Cookies

Cookies are some funny phrases (aphorisms, quotations etc.) that could be inserted randomly in your messages if you are using `%cookie macro` in your message template. Use of cookies may entertain your correspondents. The more cookies you use - the more colourful your messages look.

Another way to use cookies is to use them for greetings or one-line signatures.

These are some examples of cookies:

Editing is a rewording activity

If you save the world too often, it begins to expect it

I like kids, but I don't think I could eat a whole one

## New Mail Notification (Sound)

The Bat! can notify you about new mail arrival with a custom sound (WAV files are used, so you need a sound card installed on your machine). Account's sound settings are set to messages arriving to any account's folder by default. It is possible to define additional sound for user defined mail folders so you may hear what folder new mail came in to.

Sometimes it is necessary to disable sound within a certain time interval (for example, if a PC is used at home usually you do not want to hear it reporting to you about new mail in the middle of the night). **Allowance time interval** is designed specially for such cases. Example: 9:00-20:00.

## Sorting Office (Message Filters)

Sorting Office is implemented in The Bat! for auto-classification of **incoming**, **outgoing**, **read** and **replied** mail. Each of the above mentioned message flows can be sorted by folder, accordingly to its set of rules.

Each rule is determined by its *name*, the *source folder* (for incoming mail it is always Inbox, for outgoing - Outbox and this cannot be changed), the *target folder*, sets of signal strings and defined actions. To create a new rule, open “**Account | Sorting Office/Filters**” dialogue, select the desired rule set (depending on what type of messages you want to classify) in the tree, and press **New** button.

Related topics:

[Signal strings](#)

[Filtering actions](#)

[Using special syntax in signal strings](#)

[Using The Bat! as a Mailing List Server](#)

[Regular Expressions](#)

## Signal strings

Each set of signal strings is a list of strings to be present or absent in a certain part of a message (it can be one of: sender, recipient, subject, whole message header (kludges), message text, or whole message (kludges and text together). A message which contains (or does not contain - depending on the presence flag for a particular string) ALL the strings included in the set, satisfies the condition defined by the set.

To define several signal string sets for one rule, use the *Alternatives* page of the Sorting Office dialogue box. If a message satisfies any of the signal string sets for signal strings, the message will be moved from the source folder to the destination folder and the actions defined by the rule will be executed

Example:

String: [John Smith], Location: [Sender], Presence: [Yes]  
String: [beer], Location: [Text], Presence: [Yes]  
String: [orange], Location: [Anywhere], Presence: [No]

This rule set will be satisfied by any message, which originates from *John Smith* AND *beer* must be mentioned somewhere in the message text AND *orange* must not be found anywhere in the message.

See also: [Using special syntax in signal strings](#)

## Filtering actions

Each filtering rule has its own set of actions, which will be executed whenever a message satisfying filtering condition appears in the source folder. A very widely used action is moving a message to another folder for filing purposes - that is why *the target folder* is set at the very first page of rule properties. However, it is an option, therefore, you can set *the target folder* to be the same as *the source folder* to leave message at its place and proceed other rule actions which can be defined at **Actions** page of *Sorting Office* configuration dialogue.

The actions currently available are:

**Mark message as read** - can be used for incoming messages because messages filtered by other rule sets are supposed to be marked as read already.

**Delete message** - this is similar to making the Trash folder as the target folder. To remove the message from the server as well as deleting it from your message base, use Kill Filters.

**Add/delete the sender's address to Address Book** - this function allows you to maintain your address book and is especially useful for maintenance of your mailing lists.

**Send auto-reply** - automatically generate a reply message to the sender's address using a template and the message's text as the text for quotation.

**Create a message for** - automatically generate a message to specified address(es) using a template and the message's text as the text for quotation.

**Forward to** - forward the message to a specific address(es). Options allow you to use the entire message including header for quotation, send the message as an attach and skip sending of attachments. Forwarding template of the target folder is used for generating of the forwarded message.

**Redirect the message to** - automatically send a copy of the message to another address(es).

**Export message to a file** - write the message to a plain text file. Options allow you to append the text to the file or override the existing file.

**Run External Program** - plug-in hook. Enter a command line for the external processor of the message. For example: C:\Util\MyLists.EXE -c %1. This means that MyLists.EXE will be started, the message will be stored in a temporary file and the file name will be passed as the second parameter. *Create a detached process* option allows to "hide" the started Win32 from the task bar.

**Create a copy of message in another folder** - duplicate the message in additional folder. This may be useful if you want another user to read messages of specific type.

## Using special syntax in signal strings

Special syntax is used for signal strings:

**Pipe character “|”** means that signal string contains two alternatives. For example, string John|Jack means that search will be successful if John or Jack (or both) will be found in the specified location.

**Square brackets [ ]** mean that text embraced by them must be searched as a phrase. i.e. search for [Jack] will be successful for sentence “Jack is a good man”, but not for “Jackson guitars are very good for beginners”.

**Text embraced in quotation marks** is case-sensitive, i.e. “Internet” will be found in the sentence “The Internet is growing very rapidly” but not in “InTeRnEt RuLeZ”

The text containing special characters mentioned above, must be **embraced in single quotes**, like this:  
‘["New" generation mailing list]’



## Templates General Concept

Message templates are used for composing messages. They include static textual information in the message body. It may have, for example, a salutation and a signature as well as defining the location of quoted text in replies. From our point of view, the use of templates is a more flexible way of message composition than the use of just “signatures”.

It is also possible to use macros in templates to include message-dependent information such as addressee name, subject, current time, etc. With macros, it is also possible to define additional addressees and a message subject.

Templates defined in the account’s properties are used by default for all messages created within the account, but it is also possible to create special templates for each mail folder - these templates will be used when the folder is selected in the main window’s folder panel.

The current version of The Bat! supports four types of templates:

New message  
Reply to a message  
Forwarding of a message  
Reading confirmation

It is possible to add “Cookies” to a message using template macros

### Example of a message template

Hiya %tofname,

%cursor

Bye now,  
%fromfname                      mailto:%fromaddr

...%cookie

The initial look of new message addressed to Mike Brown <mike@domain.com> from John Smith <john@site.net> will be:

Hiya Mike,

| - *the editor cursor will be placed here by default*

Bye now,  
John                      mailto:john@site.net

...Put knot yore trust inn spel chequers

See also:

Macros  
Regular Expressions in Macros

## New Message Template

New message template is used whenever you create a new message. Once the message editor window is opened and you have all the header information (From, To, Subject) fields filled in correctly, text will appear in the editor window, depending on the new message template. It is also possible to define some message options for new messages such as:

- Default character set
- Confirm Receipt Request.
- Reading Confirmation Request.

Note that these options will be used only for new messages. Replies and forwards will not use them. Also, macros related to "original message" will not function (they will be replaced by empty strings).

See also:

Macros

Regular Expressions in Macros

## Reply Template

Reply template is used whenever you reply to a message you have received (“Reply” and “Reply to all” commands). In this case, you will see the initial text once Reply command is invoked. However you may change anything in the message header - in this case, the initial text will be adjusted accordingly to reflect the new information *unless you have changed the message before editing the header*. To include the original message’s text in the initial text of a reply, use `%QUOTES` macros to put it in the mail as a quotation and `%TEXT` to put it in as normal text.

The style of the quotation prefix is determined by **Sender information used for quotation** option. Example (John A. Smith is used as the sender)

Information used:	Prefix
None	>
Initials JAS	>
Name	John>
Last Name	Smith>
First Initial J	>
Full Name	John A. Smith>

See also:

Macros

Regular Expressions in Macros

## Forward Template

There are two ways of forwarding messages: with a template, which places the original text into the message text, and using MIME to attach the file. When messages are forwarded using MIME, it is possible to send several messages enveloped in one (message digests), however not all mail clients can manage MIME digests.

In the case of a message forwarded by placing original text in the message text, the template could be:

From: %OFROMNAME <%OFROMADDR>  
To: %OTONAME <%OTOADDR>  
Subject: %OSUBJ

----- Original message text -----  
%TEXT  
----- End of Original message text -----

Hello %TOFNAME,

%CURSOR

Best regards,  
%FROMFNAME

In the case of MIME-forwarding, the template could be:

Hello %TONAME,

%CURSOR

All the best,  
%FROMNAME

See also:

Macros

Regular Expressions in Macros

## Reading Confirmation Receipt Template

Whenever you receive messages with Reading Confirmation Receipt Request, The Bat! will generate a receipt message using this template (unless it is configured to ignore these requests).

There are four options for processing Reading Confirmation Receipt Requests:

<b>Put in Outbox</b>	generate the receipt and store it in Outbox. This allows you to make changes to the receipt or even delete it if you do not want it to be sent.
<b>Send immediately</b>	generate the receipt and send it immediately
<b>Edit</b>	the same as writing a reply, but receipt template is used instead of reply template
<b>Ignore</b>	one of simplest ways to finish your correspondence
<b>Prompt before the Action</b>	option is functional only if one of the first three methods above are chosen. In this case you will be prompted when the receipt is about to be created, if the action is not confirmed, the request is ignored

See also:

Macros

Regular Expressions in Macros

## Folders

Folders are the place where messages are kept. There are two types of folder in The Bat!: *standard* (or *system*) and *user-defined* (or *custom*). The general purpose of folders is classification of messaging so that you may easily find messages related to a particular topic.

Each account has its own folder hierarchy. Folders can be named depending on their use, but any account has four *standard* folders:

<b>Inbox</b>	the default folder for incoming mail
<b>Outbox</b>	the folder for <u>outgoing</u> messages
<b>Sent</b>	the default folder for sent messages
<b>Trash</b>	Wastebasket - all deleted messages from other folders are stored here, when a message is deleted from the Trash folder, it will be erased permanently (unless you keep a copy on the server).

Standard folders cannot be deleted.

With sorting rules (Account | Sorting Office/Filters), you can determine in which folder messages should be placed after they have been received, sent, read or replied to.

For example, if you have a contact from which you receive messages often, and wish to keep track of mails from him/her, you can create a folder with a relevant name, which gives you a clue that the folder keeps messages from this contact (his/her name, for instance). You can now create a sorting rule in the Incoming Mail group with the person's e-mail address or name as a signal string, located in sender information for the sorting rule. Once done, all incoming messages from this contact will go into the folder you have created.

*User-defined* folder may also have its own templates for new messages, replies to messages from this folder and for message forwards, as well as sounds for notification about new messages.

It is possible to move and copy messages from one folder to another by a simple drag-and-drop operation: if no keys are pressed, a Move operation is performed, if *Ctrl* key is held down while dragging, the message is copied.

A folder can have sub-folders.

To re-position a folder within the account's hierarchy, press and hold Alt key and then use drag-and-drop to move the desired folder.

## Folder settings

**Name** - the name of the folder as it appears in the folders hierarchy of the account. The name is also used as the default name for the folder's message base sub-directory

**Directory** - the directory where folder's message base file (MESSAGES.MSB) will be stored. If it is set to <default>, make sure that the folder's name does not contain characters disallowed by the file system ("\", "/", "+", "|", "<", ">", ":"), otherwise you will be asked either to rename the folder or to change the folder's directory

**Maximum number of stored messages**- the maximum amount of messages allowed to be kept in the message base. If the number of messages exceeds this value, earlier messages may be automatically deleted from message base by the *Purge* command

**Keep messages in the base for *n* days** - The maximum age of a message allowed in the message base. If a message is older than this, it may be automatically deleted from message base by *Purge* command

### On Exit actions

**Remove old messages** - purging messages using the values above

**Compress the folder** - compress the folder's message base file (if there were some messages deleted or moved from the folder). This option helps you to use your disk storage more efficiently

### Identity

It is possible to re-define identity (i.e. default "From:" and "Reply-To:" fields' contents) for a particular user folder using settings located on "Identity" page of Folder Properties dialogue.

#### See also:

[Templates](#)

[New mail notification \(Sound\)](#)

## Text Editor

The Bat! provides an easy and fast way of editing messages. The text editor has number of commands and useful features. The main difference between it and most of other Windows-based e-mail message editor is that the text being edited is shown the same way as it will appear on recipient's screen - no matter what system or e-mail client is running on the recipient's computer.

The command set is described in sections below:

Block and formatting commands.

Insertion and deletion commands.

Miscellaneous editor commands.

Cursor movement commands.

Spell checking functions.

Regular Expressions

URL highlights.



## Reading Your Mail

Received messages can be read in message auto-view panel located at the bottom of the main program window, or in a separate window, which gives the capability to read subsequent messages from the folder. It is possible to list messages by pressing the space key. In this case you will read messages from page to page, and when you reach the bottom of the current message, the first page of the next message in the folder will be shown (this applies to both modes of reading).

When folder message list box is active, it is possible to view the message using Alt+Arrow keys to scroll the message text in the window. Use <Del> key to remove messages you do not want to keep in your message base. By pressing <Enter> the separate message viewer window will be opened with the current message shown. To sort messages within a folder, use the mouse to click on the appropriate header section.

It is possible to set the amount of reading time for a message, after which the message will be marked as read (note that this setting can vary from account to account) - this reduces the risk of missing an important message.

## Block and Formatting Commands

To mark a block of text, you can use the standard cursor keys, together with the <Shift> key. With the mouse you can select text by moving the mouse cursor while pressing the left mouse-button.

Action	Key combination
Mark the beginning of a block	<b>Ctrl+K B</b>
Mark the end of a block	<b>Ctrl+K K</b>
Copy the block to Clipboard	<b>Ctrl+Ins or Ctrl+C</b>
Cut the block	<b>Shift+Del or Ctrl+XI</b>
Paste the block from Clipboard	<b>Shift+Ins or Ctrl+V</b>
Paste the block from Clipboard as quotation	<b>Alt+Ins</b>
Read the block from file	<b>Ctrl+K R</b>
Write the block to a file	<b>Ctrl+K W</b>
Move the block left to right	<b>Ctrl+K I</b>
Move the block to the left	<b>Ctrl+K U</b>
Switch block type to stream (default)	<b>Ctrl+O K</b>
Switch block type to column	<b>Ctrl+O C</b>
Switch block type to linear	<b>Ctrl+O L</b>
Justify the block or paragraph	<b>Alt+J</b>
Align the block on left end	<b>Alt+L</b>
Align the block on right end	<b>Alt+R</b>

## Insertion and Deletion Commands

Command	Key combination
Toggle Insert/Overwrite Mode	<b>Ins</b>
Delete the current line	<b>Ctrl+Y</b>
Delete to the end of line	<b>Ctrl+Q Y</b>
Delete character at the left	<b>Backspace</b>
Delete character at the right	<b>Del</b>
Delete word after cursor	<b>Ctrl+T</b>
Delete word before cursor	<b>Ctrl+Backspace</b>

## Miscellaneous Editor Commands

Command	Key combination
Save message and continue editing	<b>Ctrl+S</b>
Save message in queue and exit editing	<b>Shift+F2</b>
Send message immediately	<b>F2 or Ctrl+Enter</b>
Undo	<b>Alt+Backspace</b>
Place a position marker in the text	<b>Ctrl+K n (n=0..9)</b>
Return to a marker	<b>Ctrl+Q n (n=0..9)</b>
Insert Current Date	<b>Ctrl+Q D</b>
Insert Current Time	<b>Ctrl+Q T</b>
Uppercase Current Word	<b>Ctrl+[</b>
Lowercase Current Word	<b>Ctrl+]</b>
Capitalise Current Word	<b>Ctrl+/</b>
Search for a string	<b>Ctrl+Q F</b>
Search and replace	<b>Ctrl+Q A</b>
Repeat the last search	<b>Ctrl+L</b>
Cancel operation	<b>Esc</b>

## Block Modes

Text blocks can be:

**Stream** - Standard - as they are in most Windows-based text processors

**Linear** - whole text rows are selected

**Column** - rectangular text areas are selected. This is based on the fact that plain text editors use only fixed pitch fonts. Column blocks are sensitive to the Insert mode of the editor. If Insert mode is on, column blocks are inserted in the text while shifting the original text the corresponding number of character positions to the right, otherwise the original text is replaced within the rectangular area of the inserted block

If *Persistent blocks* are used, text selection remains until you select another region, hide the selection or delete the selected text. Otherwise, selection will be automatically hidden when you move the cursor to another position, type some text in, or insert text from Clipboard.

*Override blocks* option works only if Persistent blocks are not used. If this option is selected, a text selection will be replaced with anything you type in, insert from the Clipboard, or press <Del> key, - selected text will be deleted.

## Cursor Movement Commands

Left to one character	<b>Left arrow</b>
Right to one character	<b>Right arrow</b>
Word left	<b>Ctrl + Left arrow</b>
Word right	<b>Ctrl + Right arrow</b>
Up to one row	<b>Up arrow</b>
Down to one row	<b>Down arrow</b>
To the start of the row	<b>Home</b>
To the end of the row	<b>End</b>
Scroll up to one row	<b>Alt + Up arrow</b>
Scroll down to one row	<b>Alt + Down arrow</b>
Up to one screen height	<b>PgUp</b>
Down to one screen height	<b>PgDn</b>
To the start of the text	<b>Ctrl+Home or Ctrl+PgUp</b>
To the end of the text	<b>Ctrl+End or Ctrl+PgDn</b>
To the selection start	<b>Ctrl+Q B</b>
To the selection end	<b>Ctrl+Q K</b>

## Automatic Text Formatting

When typing text, you may use the auto-formatting options of the editor. Note that text in the plain text editor is not re-formatted automatically when you modify the text you have typed in already - in this case you should use block formatting commands

**Auto wrap mode**-if the length of the row you are typing in exceeds the value set as Right margin, the text will be automatically wrapped to the next row

**Auto indent mode**- positions the cursor under the first nonblank character of the preceding nonblank line when you press Enter.

**Backspace unindents** - aligns the insertion point with the previous indentation level when you press Backspace, if the cursor is on the first nonblank character of a line.

**Justify on wrap** - when a text row is automatically wrapped while typing, the text in the row is justified accordingly with the Right margin value

## Spell Checking

The Bat! implements two most commonly used spell checking engines - Sentry Spelling-Checker Engine (SSCE) and Common Speller API (CSAPI) for Standard and Just-In-Time Spell Checking.

### Sentry Spelling-Checker Engine (SSCE)

SSCE is shipped inside The Bat! installation package and comes with superb 100,000-word American English and British English dictionaries with comprehensive coverage of general words, contractions, abbreviations, and capitalised proper names. It is VERY fast - checks over 100,000 words per minute on a modest hardware (66Mhz 80486). The major advantages of SSCE are:

- Advanced User Dictionaries - add your own words to any of up to 30 user dictionaries. Words can be added to or removed from user dictionaries at run time. Words in user dictionaries may be offered as suggestions for misspelled words.
- Intelligent suggestions for misspelled words - SSCE can locate suggested replacements using either typographical (looks like) or phonetic (sounds like) matching. Suggestions are scored by the degree of closeness with the misspelled word, and the list of suggestions is returned in decreasing score order. The most commonly misspelled words are underlined by square-wave-line and can be corrected with mouse double-click on this word.
- Case sensitive - incorrect capitalisation (e.g., canada instead of Canada) can be reported as a misspelling. Case sensitivity can be disabled if necessary via a simple run-time option setting. Dictionaries included with SSCE contain correct capitalisation forms.
- Low resource consumption - SSCE typically uses a maximum of 450K of memory at run time and 400K of disk space. You will really appreciate this feature!
- Works with other applications' user dictionaries - SSCE can read and write user dictionaries created by other applications, including Microsoft Word and Microsoft Office (\*.dic files).

### Common Speller API (CSAPI)

The CSAPI is intended for use by all Microsoft applications, which include spell-checking. CSAPI is currently provided by many vendors and used by Microsoft Office family and many other applications. CSAPI engine is not shipped inside The Bat! installation package, but if you have Microsoft Office installed, The Bat! does automatically detect the presence of CSAPI and uses it. CSAPI is not as fast as SSCE, but the major advantage of CSAPI is good support of multiple languages. Available CSAPI dictionaries are displayed in "Language" submenu of "Spell Checker" menu below a horizontal splitting line. The absence of a splitting line in "Language" submenu indicates that no CSAPI engine is installed.

### Spell Checking Functions of The Bat!

While in Message Editor Window, you can enter Spell Checker menu. "Automatic Checking" option activates Just-In-Time Spell Checker that underlines misspelled words without any speed degradation.

Right-click on a highlighted word and a popup menu displays, letting you choose from the suggestions for the correct spelling of the word. Or, choose Ignore All to ignore the current and all subsequent occurrences of the word. Or, choose Add to add the word to the user dictionary. If it is no suggestions in its dictionary for the correct spelling of the misspelled word, suggestions are not listed in the popup menu.

It is also possible to check the text using Standard Spell Checking dialogue box (press "Check Entire Text" submenu of "Spell Checker" menu or simply F4). Below is the description of Spell Checking dialogue box buttons:



- Ignore: skips the current word (word displayed in “Not in Directory” field) without changing it and goes to the next misspelled word.
- Ignore All: skips all occurrences of the current word in the message without changing it and goes to the next misspelled word. This word would be treated as spelled correctly during the entire current The Bat! session.
- Change: replaces the current word with the correct word taken from “Change To” field.
- Change All: replaces all occurrences of the selected word in the message with the correct word taken from “Change To” field.
- Add: adds the current word to the current user directory. The name of the current user directory is displayed in “Add Words To:” drop-down list.
- Suggest: gets the list of similarly spelled words for the current word (word displayed in “Not in Directory” field).
- Undo: takes the selected word from Message Editor and places it to “Not in Directory” field.
- Options: invokes Spell Checking Engine specific options dialogue box.
- Dictionaries: invokes Spell Checking Engine specific user directories configuration dialogue box.

### **User Dictionaries Configuration**

User dictionary is a file with is a set of words in text form, and the usual extension is \*.tlx for SSCE and \*.dic for CSAPI. Words can be added to or removed from a user dictionary. User dictionaries are generally fast to access, but they take up more space per word than main dictionaries. User dictionaries can be configured using Edit Dictionaries dialogue box (press “Dictionaries” submenu of “Spell Checker” menu or pop-up menu (Right Mouse Button or Alt-F10) on misspelled word or “Dictionaries” button of Standard Spell Checking dialogue box). Edit Dictionaries dialogue box displays the entire word list and the language of current dictionary, you can easily manipulate its contents here. The file name (without path) of a current dictionary is displayed in Dictionary/File drop-down list. This list shows all active files that are used for spell checking. You can also add an existing or create a new dictionary file by pressing “New File” or “Add File” buttons. If you press “Remove” button, the current file will be discarded from the list of active dictionaries. Clicking “Import” / ”Export” buttons imports / exports a user dictionary to a text file, one word per line.

The main purpose of user dictionary is allowing user to add custom words in it. There are three ways of adding a word: from Standard Spell Checking dialogue box, from Edit dictionaries dialogue box and from pop-up menu (Right Mouse Button or Alt-F10) on misspelled word.

The Bat! checks the only text that you has entered plus your template text. If your template includes unknown words for spell checker, you can simply add them to the default user directory. Note that the user directories for different languages may vary. The quoted text and URL links are not checked.

## Character Translation Tables

Character Translation Tables are designed to manage countless national character encoding tables. For example, in there are three different encoding tables used in Russia (DOS Code Page 866, Windows 1251 and KOI-8), the same situation arises in many other countries where a non-Latin alphabet is used or a Latin alphabet with accented characters.

Translation tables determine what characters of a particular alphabet correspond to their equivalents in Windows national code page. To add a translation table to The Bat! from [Options | XLAT tables](#) dialogue, press Add button, select the file containing the translation table, then you will need to enter the name of the table being added as it appears in the program menus and the name of the character set determined by the table and as it appears in *charset* field of message header (for example, “koi8-r” for Russian KOI-8 character set).

The format of a file containing a Translation table is fairly simple. It is a stream of character pairs: *<national character 1><Windows equivalent 1><national character 2><Windows equivalent 2>...<national character n><Windows equivalent n>*

## Interface Language

The Bat! is an international program, the language of its interface can be changed from Options | Language sub-menu. Language file TheBat.LNG must be placed in the same directory where the program executable is located. This file determines the contents of Options | Language sub-menu.

## Creating a New Message

A new message can be created in four different ways:

1. By clicking on **New message** button or invoking **New message** command in the main program window menu. In this case you will have to enter the addressee manually or use the address pop-up menu.
2. By selecting the address needed from Address Book and invoking **New message** command from there.
3. When viewing a message if an e-mail address is highlighted, click the right mouse button and choose **Write a message...** command - the highlighted address will automatically be placed in the "To:" field.
4. By clicking Ctrl+N

You can assign a priority to outgoing messages and request a Confirm Receipt and Reading Confirmation.

## Message Priority

You can assign a priority to outgoing messages. The priority is only for you and your recipients – it does not affect the way mail transport systems handle the messages. New messages are created with a Normal priority. To change the priority of the current message, in Message Editor Window, use the Priority popup or Options menu. Unread received messages are displayed in the list with differently-coloured envelopes: high priority messages are red, normal priority messages are yellow and low priority messages are blue.

## Confirm Receipt and Reading Confirmation

You can request that your recipient's mail server notify you when your message has arrived to the user's mailbox on the server. To do this, in Message Editor, click on the **Confirm Receipt** in the pop-up or Options menu. The Confirm Receipt options may or may not work as described, depending on your recipient's SMTP server software.

You can also request that your recipient notify you when he have read your message. To do this, in Message Editor, click on the **Reading Confirmation** in the pop-up or Options menu. When your recipients open the message and have read it, a dialog is displayed asking them to create a confirmation message, depending on Reading Confirmation Options. The Reading Confirmation options may or may not work as described, depending on your recipient's e-mail client software.

## Sending Mail

To send messages to their destination, you must establish a connection to the Internet. If you are in the message editor window and have just finished editing a message, use the **Send now** command. If there are any problems with sending your message to the SMTP server, the message editor window will re-appear on the screen, otherwise you may see what was wrong with the message in the log panel of the main program window. If you have queued mail in Outbox, use **Send queued mail** command, and if the messages are not sent (i.e. the messages are not moved from Outbox folder to Sent mail or whatever folder is determined by Sorting rules), again, examine the account's log.

## File Attachments

To send files attached to an e-mail message:

Open message editor and enter all the information you need (i.e. addressee information, subject, and message text)

From sub-menu **Utilities** | **Attach a file** choose the encoding type, or click the right mouse button on attachments panel and choose the encoding type from there, or click **attach a file** button on the editor window toolbar. In this case the encoding will be the one you set as default for the account you are writing from).

In the subsequent dialogue, select the file you would like to send, and press the **Open** button.



## **Viewing and Opening Received File Attachments**

Files attached to a particular message are shown on the left of the message text as icons. You may use right mouse click to invoke the pop-up menu with possible actions for the file you have selected, or just double-click on the selected file to open it (note that the appropriate file associations must be defined in your system).

## Address Book

The Address Book provides you an easy way to maintain the list of contacts you have. It is possible to create groups dedicated to a particular classification or use as a mailing list. There are four possible ways to add a record to the Address Book:

1. When you see an e-mail address highlighted while viewing a message, click the right mouse button and then choose **Add to Address Book** command in the pop-up menu.
2. When you are filling in the “To:” field in the message editor, press the button with an arrow on it - e-mail addresses contained in “To:” field will be added to Address Book.
3. Using **Add to address Book** command of message list local pop-up menu.
4. Using **New address entry** menu command in Address Book.

It is possible to sort Address Book entries automatically using “**Edit | Sort**” sub-menu of Address book or manually using drag-and-drop operations.

## Addressing Your Message

The Bat! requires an address to be able to send your messages the same way as the postal service needs an address to be able to deliver a letter. The destination e-mail addresses must be entered in “To:”, “CC:” or “BCC:” fields. This is possible to do using either “direct” typing or Address Book aliases or Address Pop-Up menu (available by right mouse button click or clicking a button at the extreme right of the input field). Multiple addresses should be divided by a semicolon.

## **Message Parking**

Message parking is designed to prevent occasional deletion. When a message is parked, it will not be moved to another folder, deleted or purged automatically. It is also impossible to delete a folder that contains one or more parked messages.

## Working in Multi-User Environment

If you want to share the access to the Internet mail between several users, you may need to set access rights for particular accounts or account groups.

Users are classified by two categories:

**Administrators** - these users can see all accounts defined in the running copy of The Bat! and can change all account properties, program preferences and settings

**Ordinary Users** - these users cannot see all defined accounts. The properties they may change are determined by Administrators

If at least one ordinary user is defined, initial Log on dialogue is invoked at the program's start-up and a user has to enter his/her account name and the password (if any) to have access to his/her account. The password for a particular account can be set using "**Account | Set Access Password**" command from the main program window.

To allow an ordinary user to have several accounts and to see them simultaneously, the Administrator must define the account group for such a user.

Within a network, it is possible to give access to a particular account from a machine connected to the network. All the Administrator needs to do is to set as the account's home directory a directory accessible within the network and to create a copy of the account on machine from where the account is intended to be accessed. Even if machine does not have TCP/IP protocol installed on it, it is still possible to use The Bat! - just define the machine as *Non-TCP/IP Workstation* and working directory of the copy of The Bat! running on the machine must be the same as on the machine which is running The Bat! with status of *TCP/IP or Dial-out Server*.

See also:

[The Bat! Networking Course](#)

# The Bat! Networking Course

## Introduction to E-Mail Exchange

The Internet is a global computer network covering the entire globe. The Internet evolves as if it is a core, ensuring the correct interrelationship of different information networks, belonging to various organisations around the world. The first networks were used solely for the transmission of files and e-mail, but today it resolves the more complex problems of shared access to resources. By now, E-mail is the most widespread service on the Internet.

In order to process E-Mail exchange within either Internet or corporate network, each E-mail client should have either permanent or on-demand connection with its mail (POP and SMTP) server. A mail server is a computer that processes distribution and storage of E-Mail messages until they reach the destination address. A generic E-mail client supports at least two most commonly used E-mail exchange protocols: POP and SMTP. POP (Post Office Protocol) is used for retrieving mail from host by client. Particularity of protocol - mail message exchange is initiated by the client's request. SMTP (Simple Mail Transfer Protocol) is a simple protocol for mail transmission which is widely used on the Internet. The SMTP server's function is to receive mail from other servers and clients, and to deliver mail to other hosts and to its clients' mailboxes.

## Networking Modes of The Bat!

The Bat! can work either as a stand-alone program or as a replacement of a mail (POP/SMTP) server within a local network (e.g. Windows Workgroup, Windows NT Domain or Novell Netware) also providing a client part. There are three network modes in which The Bat! can function: *stand-alone* (TCP/IP Workstation), *server* (TCP/IP or Dial-Out Server) or *client* (Non-TCP/IP Workstation).

### TCP/IP Workstation: the *stand-alone* mode

The Bat! acts as a generic e-mail client in this mode. A generic e-mail client must have a connection with its mail (POP/SMTP) server, which can be located either inside in the corporate network or somewhere in the Internet. The Bat! can connect to the Internet using two ways:

**Local Area Network or manual connection.** If this way is chosen, The Bat! uses the connection to the Internet already established either via your LAN or by dialling your Internet Service Provider. In other words, The Bat! itself does not need to do something extra to connect to the Internet. Some non-LAN systems may be configured so that ISP is dialled automatically whenever a program is trying to connect to an Internet server and this method can be used with The Bat! However, in some cases (combined delivery or multiple account operations) The Bat! has to establish two or more TCP/IP sessions simultaneously and this may cause problems such as trying to establish two dial-up connections at the same time. We recommend to use the second way if you have a Dial-up connection to your ISP

**Dial-Up Networking Connection.** You should specify this type of connection if your LAN is not connected to the Internet itself and you are using a Dial-up connection provided by your ISP. Whenever The Bat! needs to open TCP/IP connection, it checks whether the specified 'phone book entry is active (i.e. the established TCP/IP connection, if there is any, is made using this 'phone book entry). If the entry is active, TCP/IP sessions starts. Otherwise, The Bat! will automatically dial the 'phone book entry to establish TCP/IP connection. In this case, multiple requests such as checking mail for all accounts and combined delivery (which requires two TCP/IP connections to be open at once: one for mail retrieval and another one for sending mail) will be handled correctly.

### TCP/IP or Dial-out Server: the *server* mode

It is possible that client computers of a local network have no access to the Internet or there are some restrictions applied preventing these computers use mail transfer protocols. In order to provide to users of such machines E-Mail exchange within the Internet and/or corporate network, there must be a mail (POP/SMTP) server inside the workgroup or domain.

The Bat! in the *server* mode can replace a mail server for such a network. This means that it allows you not to install a mail server inside of your local network and moreover, it gives users a possibility to process E-Mail exchange over the Internet without having their own Internet connection. A computer with The Bat! in the *server* mode is basically the same as in the *stand-alone* mode plus the use of its own Internet connection to provide E-Mail exchange of computers with The Bat! in the *client* mode over the network and/or the Internet.

### **Non-TCP/IP Workstation: the *client* mode**

The Bat! in the *client* mode uses neither Internet nor local connection to a mail (POP/SMTP) server for E-Mail exchange. It uses The Bat! in the *server* mode within the network to which it belongs. There is no limitation of the number of computers running The Bat! in the *client* mode within a network as long as the *server* machine is unique for each group of the *client* machines. The latter also means that it is possible to have several groups of *client* machines using different *servers* but those groups must not intersect.

### **Setting up The Bat! Networking**

#### **The Bat! Networking: How Does This All Work**

All account data and message bases that are used by computers in the *client* mode are located on the *server* computer and the data is being accessed using the local network.

When the user of a *client* machine wishes to send a message to an Internet address, it places the message into the Outbox which is physically located on the *server* machine and invokes the command that makes the *server* send the outgoing messages using the *server's* connection to the Internet.

When a *client* wishes to check its POP account, it invokes the command to the server to check the *client's* mailbox on the mail server. The server connects to the POP server using its own Internet connection, receives the messages, stores the messages to the *client's* message base and invokes the command to the *client* to re-read its message base contents. When the *client* receives this command, it notifies the user that the new mail has arrived. The new message arrival is indicated in the folder tree screen of the *client*, and on the MailTicker™, if the MailTicker™ is configured.

To turn on the MailTicker™ (there must be some unread messages though), use "Options | MailTicker™ | Show Always" or "Show Automatically" options of the main window menu.

#### **Setting up a TCP/IP or Dial-out Server (*server*), Step by Step**

1. Within your local network, select a computer, which is connected to the Internet. This computer will be a "TCP/IP or Dial-out Server" (*server*).
2. Using "My Computer" desktop icon, choose the local drive or directory with The Bat! installed or where you are planning to install The Bat!. This location will be used to store message bases. Share this path to the network users that will run The Bat! in *client* mode on their computers. If you are already using a network drive, e.g. on a dedicated file server, you may skip this and the next step.
3. Map the resource you have shared to a network drive's letter. Other computers that will act as *clients* within your network using this network drive to exchange the information with the server.
4. If you do not have The Bat! installed yet, install it to the mapped network drive in order to machines with The Bat! in the *client* mode can access it. Run The Bat! from the network drive that you have mapped.
5. Click "Network & Administration" item of "Options" menu to make sure that this machine is in "TCP/IP or Dial-Out Server" (server) mode.
6. Create the accounts that will be used by *client* machines. Make sure that the home directories of these accounts

are located on the network drive so the *client* users can access them.

7. Invoke "Network & Administration" dialogue box again and manage the privileges of accounts you have created.

### **Setting up a Non-TCP/IP Workstation (*client*), Step by Step**

1. Map the resource that server has shared to the same drive letter as it is mapped on the server.

2. At the installation of The Bat! on a *client* machine, select the program's working directory on the network drive letter that you just have mapped. The working directory must be the same as on the server.

3. If you have The Bat! already installed on this computer, please proceed the following steps, otherwise, if you just have installed The Bat! accordingly to the previous step, the setup is finished.

These sub-steps describe how to change the machine mode from *stand-alone* to *client* without reinstalling The Bat! If you are not familiar with RegEdit, it would be better for you to uninstall The Bat! and to go to the step "Installing The Bat! on a *client* computer"

a) Enter "Network & Administration" dialogue box and make sure that this machine is "Non-TCP/IP Workstation".

b) Exit The Bat!

c) Change the mail directory using RegEdit. HKEY\_CURRENT\_USER\Software\RIT\The Bat!, Value name: Working Directory, and the value data is the path to mail directory, which should be on a network drive, and it should be the same as on the server.

### **Using The Bat! As a Core Element of E-Mail Exchange within a Corporate Network**

There may be some reasons to establish E-Mail Exchange for local corporate networks that have neither POP/SMTP servers nor Internet access. In this case, there should be a technology of a local message delivery, i.e. the messages must be routed, delivered and stored locally without using POP/SMTP servers.

Local Delivery Technology is provided by The Bat! with the appropriate option turned on. To activate it, open "Network & Administration" dialogue box on server, and tick the "Allow Local Delivery" checkbox. This option allows several machines to work in a local network exchanging messages without using SMTP/POP server at all. This feature brings an efficient solution for building corporate mail-exchange networks without using Internet Mail Transfer Protocols. When "Allow Local Delivery" is on, the outgoing messages addressed to local users, are dispatched directly and put into appropriate message bases instead of sending them to SMTP server.

See also:

[Working in Multi-User Environment](#)  
[Account Groups](#)



## Account Groups

If a particular user wants to have access to several e-mail accounts, it is possible to define a group of accounts to be displayed whenever such a users logs on to The Bat!

To define an account group, use “**Options | Network and Administration**” command from the main program window.

To log into The Bat! using access to an account group, just enter the group name and password instead of an account’s logon data in the login screen.

See also:

[Working in Multi-User Environment](#)

[The Bat! Networking Course](#)

## Common Errors

### Mail cannot be sent

- Is your connection to the Internet is established?
- Make sure that TCP/IP protocol is installed on your system.
- Check network or modem cables connection to your computer
- Does your Internet Service provider have SMTP and POP3 servers?
- If all above is set up correctly and present but the problem persists, contact your network administrator or Internet Service provider.

### Message text is unreadable

Check that you are using correct character translation table. If you are, check the message header for field named "Content-transfer-encoding" - it is possible that the message is encoded using method unknown to The Bat!; you have to ask the sender to re-send the message using either base64 or quoted-printable encoding.

POP3 (Post Office Protocol Version 3) - a “post office” protocol is used for retrieving mail from host by client.  
Particularity of protocol - mail message exchange is initiated by the client's request.

SMTP (Simple Mail Transfer Protocol) - a simple protocol for mail transmission which is widely used on the Internet. The SMTP server's function is to receive mail from other servers and clients, and to deliver mail to other hosts and to its clients' mailboxes.

IMAP4 stands for Internet Message Access Protocol Version 4. It is a method of accessing electronic mail or bulletin board messages that are kept on a (possibly shared) mail server. In other words, it permits a "client" email program to access remote message stores as if they were local. For example, email stored on an IMAP4 server can be manipulated from a desktop computer at home, a workstation at the office, and a notebook computer while travelling, without the need to transfer messages or files back and forth between these computers. Speaking other words, IMAP4 is a superset of POP3 that provides good support for all three modes of remote mailbox access: offline, online, and disconnected.

The name of the e-mail account as it appears in account tree of the main program window. It may be any combination of characters. The only limitation is that the name must be unique within the program; in other words, there must not be another account with the same name.

The path to the directory where all files related to the given account are stored. This directory is also used for creating default sub-directories (each mail folder keeps its messages in MESSAGES.MSB file in its mail-folder directory, which by default is placed in the home directory of the account it belongs to)

Click this button if you want to specify another Home directory



Click this button to specify default Home directory

Click this button to perform the next step

Click this button to return to the previous step

The originator's name which will be put in the header of each message created for the account. For example, John Smith.

The Originator's e-mail address which will be put in the header of each message created for the account. For example, john.smith@mydomain.com

Organisation to which originator belongs (if any). Example: Mythical Software, Inc.

The address of your SMTP server to which all outgoing e-mail messages from the account must be sent. The address may be represented as a traditional Internet symbolic name (e.g.: mail.ritlabs.com) or as an IP numeric address (e.g. 193.219.214.38).

The address of your POP3 server where the account's mailbox is located. All messages from the Internet are accumulated on POP3 servers and may be retrieved from there by the Get new mail command or by Message Dispatcher. The address may be represented as a traditional URL (e.g.: mail.ritlabs.com) or as an IP numeric address (e.g. 193.219.214.38)



The POP3 user name for access to your mailbox on the server. Usually, it is the same as the part of the account's e-mail address before "@" symbol. For example, if the e-mail address is john.smith@ritlabs.com, the user's name will be john.smith

The password for logging on to the POP3 server. You may leave this field blank - in this case you will be prompted to enter the password each time you try to receive mail.

If this options is set, your messages will not be deleted from the server after their retrieval. This gives you an option to re-download a message later if you have lost its copy on your machine.

Closes this dialogue box without saving the changes you have made.

The To field indicates the primary recipient(s) of a message.

CC (Carbon Copy) recipients of a message are the persons you want to “listen in” to the message you are sending to the primary recipient. CC recipients receive the message normally, but they do not see themselves as the primary recipient of the message; nevertheless, all CC recipients are listed in the message header.

Use a BCC (Blind Carbon Copy) if you do not want the primary addressee to know that the message is sent to somebody else. This may be helpful when it is necessary to send a notifying message to a huge group of people - using BCC will seriously reduce the message size.

## Configuring The Bat!

After you have installed The Bat!, the program starts automatically. If you are installing The Bat! for the first time, you will be prompted about working directory. The working directory is the most important directory for The Bat! functioning - all account home directories, account data and address books will be stored in there by default. You also will be prompted about adding The Bat! icon to your Desktop, Start Menu and Sent-To menu. If account list file is found in the directory you have specified, The Bat! will automatically add the listed accounts into your copy of The Bat!, otherwise the **New Account Wizard** will open.

Choose the name of the account you want create and specify the home directory of this account. If the home directory contains account configuration files (file names begin with ACCOUNT), the data from those files will be used as default for your newly created account. If you choose <default> as the home directory, The Bat! creates a sub-directory of the working directory with the same name as the account.

After you have the new account name and home directory chosen, click **Next**, the *New Account Wizard* begins walking you through the fields for which you need to enter setup information. You will need to supply the following items in the panels of the *New Account Wizard* (click **Next** after filling out each page):

**Your Name** - Enter your name as you would like it to appear in the From field of all of your outgoing messages from this e-mail account, indicating to your recipients who the mail is from.

**E-mail Address** - Enter the e-mail address that has been assigned to you by your Internet Service Provider or your organisation's e-mail administrator. This is the address that other people will use to send you e-mail. E-mail addresses are generally of the form *username@domainname*, and *serg@ritlabs.com* is an example.

**Organisation** - Organisation which you represent (if any). The information about your organisation name is placed in the header of all messages you write. Mythical Software, Inc is an example. If you do not want to identify your organisation, leave this field blank.

**SMTP Server** - To send messages in The Bat!, you must have access to a computer running an SMTP (Simple Mail Transfer Protocol) server. Your outgoing messages are sent to the SMTP server, which delivers them to your recipients. The address may be represented as a traditional Internet symbolic name (e.g.: *mail.ritlabs.com*) or as an IP numeric address (e.g. 193.219.214.38).

**POP3 Server** - All of your incoming e-mail messages are delivered to your incoming e-mail account, which resides on a computer that runs your incoming e-mail server. Once your messages arrive at your mail account, The Bat! picks them up and transfers them to your PC. Your incoming e-mail server must use one of the two Internet-mail communications protocols POP (Post Office Protocol) or IMAP (Internet Message Access Protocol). In the edit box, type the full name of the computer that runs your incoming e-mail server: *mail.ritlabs.com* is an example.

**Username** - Enter the name that you will use to log in to this e-mail account. This name is provided by your Internet Service Provider or your organisation's e-mail administrator, and it usually consists of the text that appears before the at sign (@) in your return e-mail address. In the example *serg@ritlabs.com*, the login name is *serg*.

**Password** - Enter the password you use to access your mail server. You may leave this field blank - in this case you will be prompted to enter the password each time you try to receive mail. The users of Ritlabs AuthenticBat! or Ritlabs SecureBat! can use hardware authentication based on the iKey token.

**Leave copy of messages on the server** - You can use your account from different places. In this case, you may want to leave messages on the server to make it possible to retrieve them from other places where you use your mail account. If you do not want to leave copy of messages on the server, the messages will be deleted from the server mailbox after their successful retrieval.



Once you have completed setting up your account via the *New Account Wizard*, you are now ready to send and receive messages. However, you may need to take an additional step in order to send messages.

If the computer that runs your POP3 or IMAP server (incoming e-mail account) also runs an SMTP server, then no additional setup action is required. You are now ready to send and receive messages in The Bat!.

If you ever wish to change the settings of your e-mail account, you can do so from the Account Properties dialogue. It is available by “Account | Properties” menu command when the account is selected in the folder tree of the main window.

## Registering your copy of The Bat!

The Bat! is a Shareware program. If you intend to use The Bat! after the trial period of 30 days, you must register your copy of The Bat! or stop using it. When you register The Bat!, you are eligible to receive Technical support from The Bat! support team as well as features for registered users can be used from then.

**Note:** If you are part of a site license, you do not need to register. Contact your organisation's e-mail administrator for support.

To get the latest licence prices and register your copy on-line using credit card, please visit The Bat! official WWW home page located at [http://www.ritlabs.com/the\\_bat/](http://www.ritlabs.com/the_bat/) or contact The Bat! support team via e-mail [\*info@thebat.net\*](mailto:info@thebat.net)

## Tip of The Day

Each time you open The Bat! (including the first time after install), the Bat! Tip of the Day dialogue is displayed, showing you the Tip of the Day. You can display next and previous tips by clicking the Next Tip and Previous Tip buttons. To prevent the Tip of the Day dialogue from being displayed on start-up, uncheck the checkbox. You can always display the Tip of the Day from the Help menu. Click the Close button to close the Tip of the Day dialogue.

You can add your own tips or your favourite quotations by editing the file named **“TheBat.tip”** located in the directory where The Bat! is installed. To change the order of tips, delete the file **“tips.ini”** located in the same directory.

## Technical support

If, after reviewing all of the available materials, you are still in need of assistance, contact your e-mail administrator (your Internet Service Provider or your company's The Bat! support coordinator) or The Bat! Support team. If you are eligible for technical support, select "Feedback | Information request" from the Help menu.

You must register your copy of The Bat! to receive technical support. See [Registering Your Copy of The Bat!](#) for details.

## Quitting The Bat!

To quit The Bat!, select **Exit** from the **Message** menu, or press Alt+X. If you have any mail retrieving or sending sessions, you will hear the Windows exclamation sound which means that you cannot quit The Bat! right now.

The Trash folders of accounts are emptied if the **Empty Trash on exit** option is on in the Account Properties. Also, all folders will process their Exit actions (purge and/or compress) which can be set in Folder properties dialogue.

## Uninstalling The Bat!

You can uninstall The Bat! by using the tools provided with your Windows 95 or Windows NT 4.0 (or later) operating system. Open the Control Panel, double-click on **Add/Remove Programs**, select The Bat!, and click **Remove**.

## Creating an Outgoing Message

An outgoing message is a message you send to someone else. The simplest way to create an outgoing message is to select **New** from the **Message** menu or to press Ctrl+N. A new message window is displayed, called the message editor window.

You can assign a priority to outgoing messages and request a Confirm Receipt and Reading Confirmation.

## Using the Message Editor

The message editor window is invoked whenever you edit/reply/forward a new e-mail message. Message editor window consists of the menu bar, the Toolbar, the message header, the Status bar, the attachment window and the message body (text editor with Spell Checking)

### Menu Bar

The menu bar provides common commands that can be used while you edit messages including:

- saving messages in various ways
- attaching files
- standard editing functions such as Copy/Paste/Cut operations as well as additional Pasting/Copying text from/to disk files, pasting as Quotation;
- special operations like block mode switching, inserting current date/time;
- spell checking functions;
- OpenPGP functions and options;
- changing view options (switching on/off header fields, original message text when replying, toolbar and header fields visibility);
- changing message preferences such as priority, character encoding, confirmation receipt requests and the active account from which the message should be sent.

### Toolbar

The toolbar consists of a series of buttons that are displayed just under the title bar. It allows you to perform the operations, which are used often by one mouse click.

### Message header

Outgoing mail headers consist of six fields: *From*, *Reply-To*, *To*, *CC*, *BCC*, *Subject* and *Follow up*. All the fields can be directly edited. To move the cursor from field to field, press either the Tab key or use Up/Down arrows or click in the desired field with the mouse. Most of header fields have history drop-down list, which is used for auto-complete and also can be activated by Alt+Down arrow keystroke.

*Addressee fields* (To, CC and BCC) have quick buttons to invoke address selection dialogue. You can also use **Shift+Enter** keystroke for this. The other feature is quick address pop-up menu which can be invoked either by pressing **Alt+Enter** or by right mouse click - the menu contains the addresses from the address book with “**Add to pop-up menu**” option set. When you type an address, you can press **Ctrl+Plus** to let The Bat! find an addressee with the name or address beginning with whatever you have typed, pressing Ctrl+Plus again takes you to another addressee with matching name or address.

*From* and *Reply-To* fields contain sender's identity and the address to which replies must be sent. These fields can be edited either directly or by choosing alternate addresses from their drop-down lists or by choosing Active account from the Options menu.

The *Follow up* contains the identifier of the message to which a reply is created (empty for a new message) - this field should not usually be edited.

The *Subject* is some brief text indicating the contents of the message. This field can be left blank, although it is considered a point of e-mail etiquette to include a Subject with each message.

### Status bar



The status bar is the line below the message body. It shows current status of the editor and message preferences (left to right): Caret position, Modified flag, current Block mode, current Input mode, the message priority and receipt requests flags, the last is the message's character set). If you click with right mouse button on any section except two first ones, you'll get a pop-up menu to change an option, which attached to a particular status panel.

*See also:* Text Editor

Autosave function automatically saves your message while you editing it within a certain period of time. To enable this facility, use “Options | Editor preferences” dialogue.

## **Saving a Message for Later Changes**

Sometimes it is convenient to save an outgoing message either as a safeguard when typing long messages, or so you can return to it later to make changes. You can use either “Save” or “Save as draft” or “Autosave”

To save the current message, select Save from the File menu. Saved messages are put in the Outbox and the saved message is shown with a Road works sign in the Status column. The Road works sign indicates that the message is not yet completed and thus cannot be sent from the outgoing queue.

You can continue making changes to the message or close it. If you try to close an outgoing message window without saving that version of the message, an alert is displayed asking if the message should be saved or the changes discarded. If you select Discard and the message has never been saved, the message is deleted.

## **Sending a Message Immediately**

If you want to send your messages immediately instead of putting them in a queue to send later, use F2 key while you are in the message editor (be sure the Immediate delivery option is chosen in the Account Properties, use Shift+F2 otherwise).

To send the current message, click on the **Send** button or select **Send Now** from the Message menu. A progress window is displayed to show the progress of the transmission.

*See also:* [Queueing a Message to Send Later](#).

## Queuing a Message to Send Later

If you want to put your messages in the outgoing queue (in the Outbox) to send all together at a later time use Shift+F2 key (be sure the Immediate delivery option is chosen in the Account properties, use F2 otherwise).

To put the current message in the queue, click on the **Queue** button or select **Queue in Outbox** from the Message menu. The message window is closed, the message is saved in the Outbox without Road works sign which means that the message is ready to be sent.

To send all of your queued messages, select Send Queued Mail from the Account menu of the main window. A progress window is displayed momentarily at the top of the screen indicating the progress of the transmission.

*See also:* [Sending a Message Immediately](#).

## Protecting your account's mail with a password

With The Bat!, you can “lock” your account with a password so nobody can read your mail without your permission. This can be often used when your computer is located in office where anybody can access your computer while you are away. Do not confuse the account access password with your mail server password - these are two different things so make sure that you use two different passwords for your privacy sake.

In multi-user environment, the account access password is also used for logging on.

To protect your account with a password, use Set Access Password command of Account menu or press Ctrl+F12. You will be prompted about the password you want to use to access your account. If you have a password set already for your account, you can enter empty string as your password - the account will be unprotected after doing this

## Secure messaging with OpenPGP

The Bat! uses OpenPGP mechanisms to protect your messages from unauthorised reading and/or modification.

To use **PGP v2.6.3** with The Bat!, just install PGP completely, and make sure that PGP executables stay in your system's PATH Environment variable and you have PGPPATH is set accordingly to PGP documentation. Please read the documentation on PGP carefully before using it. Outside U.S., you may download PGP v2.6.3 International. U.S. version of PGP 2.6.3 is no longer available.

To use **PGP v5.5.3 or higher** with The Bat!, just install PGP completely and make sure that all PGP DLL's can be located using your system's PATH Environment variable. United States and Canada residents may download PGP 5.5.5 (U.S. Export Restricted) from its official Web site. Outside the U.S., you may download PGP 5.5.3 International and PGP 5.5.5 User's Manual from the International PGP WWW site - **<http://www.pgpi.com>**.

After you have your version of OpenPGP installed accordingly the above instructions, you can start your secure messaging and OpenPGP functions in the main window's Tools menu and the message editor's OpenPGP menu are available to use.

See also:     [Encrypting your message with OpenPGP](#)  
                  [Decrypting OpenPGP signed messages](#)  
                  [Signing your message with PGP](#)  
                  [Verifying OpenPGP signed messages](#)  
                  [Adding OpenPGP public keys from a message](#)

## Signing your message with OpenPGP

To ensure your recipients that messages they receive are from your e-mail address, it is recommended to encrypt your messages with PGP. OpenPGP signed messages contain their original text plus your key information so if somebody tries to modify a message signed by you, the message won't be verified successfully by its end-recipients. OpenPGP signing can be used together with encryption.

With The Bat!, you can sign your messages from the message editor either manually using “OpenPGP | Sign block” (“Sign entire text”, “Sign and encrypt entire text”) command or automatically using the on/off option “Sign when completed” of OpenPGP menu.

Before sending signed messages, make sure that your recipient have your public OpenPGP key, otherwise they won't be able to verify your signature validity. Please refer to the manual of particular OpenPGP implementation about the distribution of your public key.

See also:      [Encrypting your message with OpenPGP](#)  
                 [Decrypting OpenPGP signed messages](#)  
                 [Verifying OpenPGP signed messages](#)  
                 [Adding OpenPGP public keys from a message](#)



## Encrypting your message with OpenPGP

To send message securely over the Internet, it is recommended to encrypt your messages with OpenPGP. A OpenPGP encrypted message cannot be read by anybody but the persons whose public OpenPGP keys are used to encrypt the message. OpenPGP Encryption can be used together with signing.

With The Bat!, you can encrypt your messages from the message editor either manually using “OpenPGP | Encrypt block” (“Encrypt entire text”, “Sign and encrypt entire text”) command or automatically using the on/off option “Sign when completed” of OpenPGP menu.

To encrypt your message to a particular recipient, you must have the recipient’s public key. Please refer to the manual of particular OpenPGP implementation to learn how you can obtain public keys.

See also:     [Decrypting OpenPGP signed messages](#)  
                  [Signing your message with OpenPGP](#)  
                  [Verifying OpenPGP signed messages](#)  
                  [Adding OpenPGP public keys from a message](#)

## Verifying OpenPGP signed messages

Whenever you receive a [OpenPGP](#) signed message, you can check whether the signature is valid so the information in the message has been sent by an authorised person and is not changed before it has arrived your mailbox. To do this, you should have OpenPGP implementation properly installed and the sender's public key in your public key database.

To verify a signed e-mail message from The Bat!, use “Check OpenPGP signature” command from the Tools menu. If the message is encrypted, it will be automatically decrypted (OpenPGP implementation may ask you for your passphrase).

See also:     [Encrypting your message with OpenPGP](#)  
              [Decrypting OpenPGP signed messages](#)  
              [Signing your message with OpenPGP](#)  
              [Adding OpenPGP public keys from a message](#)

## Decrypting OpenPGP messages

Whenever you receive a PGP encrypted message, you can decrypt it you have OpenPGP implementation properly installed and an appropriate private key in your key database. Please refer to the manual of OpenPGP implementation to get more information about OpenPGP key pairs.

To decrypt an encrypted e-mail message from The Bat!, use “Check OpenPGP signature” command from the Tools menu. If the message is signed, OpenPGP automatically checks its signature validity. Note that the message is NOT stored so you need to decrypt it this way each time you read the message.

Another way is decryption as a message into your message base using “Decrypt OpenPGP...” command of the Tools menu. Note that in this case your privacy might appear as defenceless because the message is stored in your message base in clear text so anybody else can read it.

See also:     [Encrypting your message with OpenPGP](#)  
                  [Signing your message with OpenPGP](#)  
                  [Verifying OpenPGP signed messages](#)  
                  [Adding OpenPGP public keys from a message](#)

## Adding OpenPGP public keys from a message

OpenPGP public keys can be distributed in plain ASCII text and therefore it is possible to embed public key blocks in an e-mail message. Below is an example of a message with public keys embedded.

Hello John,

This is my public key:

```
-----BEGIN PGP PUBLIC KEY BLOCK-----  
Version: 2.6.3i
```

```
mQBtAzPHTM0AAAEDAKCJ5/hM+qpXHaa4wFumqU2DEV4KfkKe5hkkfS6knTtn0rnD  
Zm/90txo+29gOFARBL8ynIpGA7fUgmNG13mprN9q/9xrtH4gg6jV/cYJ3ZtnKN9B  
sfw7QLKsW3r6eMYGsQAFebQiU3RlZmFuIFRhbnVya292IDxzdGVmQHJpdGxhYnMu  
Y29tPokAdQMFEDPHTM2sW3r6eMYGsQEBft0DAIi04Yya7Q5ZZAtz2K4bKHitMBFV  
j1g0Vmr87Cr5qZYgGBzbNasyFwUuFqwqEuiwnLVNSpPiK8QsJTff6Ky9hqMUiZaL  
ENozJ5f9GKw0pmMio7rOlKRE8Szbl6RkpeCUQw==  
=u13G
```

```
-----END PGP PUBLIC KEY BLOCK-----
```

All the best,  
Stefan

To import OpenPGP public keys from an e-mail message with The Bat! use “Import OpenPGP key” command from Tools menu.

See also:     [Encrypting your message with OpenPGP](#)  
              [Decrypting OpenPGP signed messages](#)  
              [Signing your message with OpenPGP](#)  
              [Verifying OpenPGP signed messages](#)

## Select OpenPGP Implementation

**Internal (RFC-1991)** - Internal OpenPGP implementation, supporting RSA keys up to 4096 bits with 128-bitIDEA cipher and MD-5 hash, needs no external executables or plug-ins. [Submission Forms](#) work with this mode only.

**PGP command line (2.6.3, 6.0.x+)** – This mode allows using any command line Win32 PGP executable: PGP 2.6.3, 6.0.x, 6.0.x, 6.5.x, 7.0.x, etc, and compatible. Just make sure that PGP.EXE stays in your system's PATH Environment variable. To use PGP 2.6.3 in this mode, make sure that you have PGPPATH set accordingly to PGP documentation. For more detailed help, consult the PGP User's Guide.

**PGP 5.5.x plug-in** - To use this mode, install PGP 5.5.x for Win32. Make sure that you've rebooted your computer after PGP installation. The plug-in (batpgp55.dll) may not be a part of The Bat! installation and may be installed separately.

**PGP 6.0.x / 6.5.x plug-in** - To use this mode, install PGP 6.0.2 or 6.5.x. for Win32. Make sure that you've rebooted your computer after PGP installation. The plug-in (batpgp60.dll) may not be a part of The Bat! installation and may be installed separately.

**GNU Privacy Guard (GPG)** – To use this mode, install the GNU Privacy Guard command line Win32 executable. Make sure that GPG.EXE stays in your system's PATH Environment variable. Refer to the GPG Manual.

## Submission Forms

Submission forms are a tool for creating queries that should be automatically processed. A query is actually a message with strictly defined fields and string variables.

The queries come from a user equipped with The Bat! software with form templates (files with a TBC-extension). Such a user is called the client. Before submitting a query, The Bat! asks the user to fill out a form, that looks very much like forms that you have probably filled out on the Web. The program that processes user requests is called a server. The Bat! transforms data filled in by the user to an e-mail message, encrypts and signs the message digitally. This message is sent to the server via e-mail. The address of the server is defined in a form template. The system of interacting of client and server by means of queries and replies is called The Bat! Client-Server.

This system is intended for remote access to data and managing data in off-line mode securely. The purpose of this system is to control bank accounts, make ticket and hotel reservations, etc. In order to provide secure transactions, encryption and digital signing is used (RSA/IDEA/MD5 algorithms), defined in RFC-1991. Server software is specific and depends on its tasks: Internet-shops, Banks or Ticket Reservation Agencies may use different types of server software, but the client, using The Bat!, can seamlessly use all of their services.

In order to provide a proper transaction, format of query fields must be equal on both client and server parts. As the server usually works with more than one client, the form templates (TBC-files) are usually being created on the server, signed and sent to clients before starting any transactions between the client and the server. While making transactions, the server can start new kinds of services for its users; in this case, the server would send any new versions of form templates in order for the client to be able to use those new services. Upon reception of a template file and verification of the digital signature, the client adds it to the list of template forms.

The presence of a digital signature is mandatory. It does not only ensure that the formats are equal on both client and server, but protects the system from intrusion of unauthorised templates. This unauthorised intrusion can come as a templates renewal and can cause all or a part of clients to have unauthorised templates instead of authorised ones. This can cause “freezing” of networks or information drain. The changing of a server’s e-mail address in a form template can cause such a drain.

See also: [The Format of a Submission Form Template](#).

## The Format of a Submission Form Template

Form templates are stored in a text file with a TBC-extension. The file content is signed by a digital signature. Form templates (TBC-files) are usually being created on the server, signed and sent to clients before starting any transactions. Upon reception of a template file and verifying the digital signature, the client adds it to the list of template forms (Options/Submission Forms). See [introduction](#) to learn more about submission forms.

A template describes the properties of a screen form window (which is filled in by the client), e-mail message attributes (in which the query will be sent) and variables of request. Each string of a form template can contain one keyword with one or more parameters. Lines that start with “;” character are treated as comments.

### Window and Message Properties

Keyword MSG is used to define the properties of a screen form window and message attributes, other keywords are used to fill in the form. MSG keyword can have the following parameters: FORMTITLE (title of screen form window, string value), FORMWIDTH (width of the window, numeric value), FORMHEIGHT (height of the window, numeric value), AREAWIDTH (width of area inside a window, this area will be scrollable if it exceeds the window, numeric value), AREAHEIGHT (height of the area inside a window, numeric value), TO (e-mail address of destination server, string value), SUBJ (Subject field of the message, string value), PRIORITY (priority type of the message, string value), ENCRYPT (whether the message should be encrypted, ON/OFF), SIGN (whether the message should be signed, ON/OFF), BATCH (whether the same similar query needed to be submitted after submitting the query), QUEUEOUT (do not send the message immediately; queue in outbox), TPL (query content is a plain human-readable text), NFO (info put as a plaintext before encrypted data), FILEDLGTITLE (title for a file open dialogue window for VFILERE), FILEDEFNAME (used for VFILERE to specify the file name that appears in the file open window when the window opens), FILEFILTER (used for VFILERE to determine the [file masks](#) available in file open window), FILEDEFEXT (used for VFILERE, if you want a file extension automatically appended to the filename typed in a file open window).

### Query Content

Query content is the data that a user has entered while filling in the screen form. When the user clicks OK button in a form, The Bat! creates an e-mail message with the query content. If stated in a form template, the message content will be signed and/or encrypted. Then the message will be sent via e-mail to the server. E-mail address of the server is also stated in a form template.

If TPL parameter of MSG keyword is not specified, the message body looks like an unlimited-length line and contains the variables, which are separated by the ampersand (“&”) character. The names of the variables are defined in a form template. The values of the variables are taken from the fields of a screen form, which the user has filled in before clicking OK. In the query string which is being transmitted to server in message body, the name and the value of a variable are separated by an equal sign (“=” character). E.g., the query contains two variables, named **servicetype** and **accountnumber**, the first variable has “getaccountlog” value and the second variable has “849780094” value; in that case the message body will look like this:

**servicetype=getaccountlog&accountnumber=849780094**

This method of generating of a query string conforms to RFC1866 (paragraph 8.2.1, **application/x-www-form-urlencoded** content-type).

Names and values of variables can contain any characters, but some characters should be escaped according to the following rules: the space character should be replaced by a plus sign character. Non-alphanumeric characters are replaced by “%HH”, a percent sign and two hexadecimal digits representing the ASCII code of the character. Line breaks, as in multi-line text field values, are represented as CR LF pairs, i.e. “%0D%0A”.

If TPL parameter of MSG keyword is specified, the query will be sent in a human-readable text according to a given template. You can specify multiline template, each additional TPL parameter adds a line to a template. Names of variables must be enclosed in \$-characters, and will be replaced to respective values.

## Overview of Form Elements

A screen form can contain such elements as input lines, lists, check-boxes, labels and buttons. All these elements are defined in a form template by means of keywords: EDIT (single-line input field), MEMO (multi-line input field), CHECKBOX, LISTBOX (standard list), COMBOBOX (drop-down list) – all of these elements define variables, one variable per element. Besides that, there is a special invisible element, HIDDEN – the user cannot affect the value set by this element, and visible LABEL element that doesn't define a variable and is used as a label for any other visible element. Two elements are used to control the screen form: SUBMIT (button, that causes the query to be sent to server) and CANCEL (button, that cancels the form).

## Overview of Element Parameters

Screen form elements can have the following parameters: **X**, **Y** (coordinates of topmost left corner within a window), **WIDTH**, **HEIGHT** (width and height of an element, most of the elements are defaulted or calculated from font height, so these parameters are optional), **NAME**, **VALUE**, **DEFTPL**, **VBODYRE** (defines name and value of variable, applies to all elements except LABEL, CHECKLISTBOX, SUBMIT and CANCEL). The EDIT element can have a **MASK** parameter that defines formatted input and **PASSWORD** parameter that allows to display an asterisk character in place of any entered text; LISTBOX, COMBOBOX and CHECKLISTBOX elements can have an **ITEM** parameter that adds an item to the list; LABEL, CHECKBOX, SUBMIT and CANCEL elements can have a **CAPTION** parameter.

Besides that, the LABEL element can have some extra parameters: **FOCUSNAME**, which links the label to another element - if the value of the CAPTION argument of a label includes an accelerator key ("&" character) and **SHOWACCEL** is ON, the element named as the value of the FOCUSNAME parameter becomes focused when the user presses the accelerator key. LABEL keyword can also have a **JUSTIFY** parameter, which specifies how text is aligned within the label. The value of JUSTIFY can be on of: LEFT (align text to the left side of the label), CENTER (center text horizontally in the label), RIGHT (align text to the right side of the label), **WORDWRAP** (allows the label to display multiple line of text)

## List of Form Elements

<b>EDIT</b>	<b>Single-line input field.</b> Use EDIT element to put a standard Windows edit box control on the screen form. Edit boxes are used to retrieve information from client, because the client can type data into an edit box. The client can be forced to enter only valid characters using a MASK parameter. MASK can be also used to format the display of data. <i>Mandatory parameters: X, Y, NAME; optional parameters: WIDTH, HEIGHT, VALUE, DEFTPL, VBODYRE, MASK, PASSWORD.</i>
<b>MEMO</b>	<b>Muilti-line input field</b> , which displays text to the client and permits the client to enter text into the application much like the EDIT component. The difference is that the MEMO element permits multiple lines to be entered or displayed, but doesn't permit masked input, unlike EDIT. <i>Mandatory parameters: X, Y, NAME; optional parameters: WIDTH, HEIGHT, VALUE, DEFTPL, VBODYRE.</i>
<b>CHECKBOX</b>	<b>Switch.</b> This element is used to present On/Off (Yes/No, True/False) options to the client, particularly where more than one choice at a time is available from a group of choices. The string value of the checkbox can be either "on" or "off". When the form is submitted with a value of "on", it will be sent as "name=on". Mandatory parameters: X, Y, NAME; optional parameters: WIDTH, HEIGHT, VALUE.
<b>LISTBOX</b>	<b>Standard list.</b> This element defines a Windows list box. A list box displays a list from which users can select one item. <i>Mandatory parameters: X, Y, NAME; optional parameters: ITEM, WIDTH, HEIGHT, VALUE, COLUMNS.</i>



<b>COMBOBOX</b>	<b>Drop-down list.</b> This element defines a list that has much in common with LISTBOX, but it drops down and the only one item can be selected. <i>Mandatory parameters: X, Y, NAME; optional parameters: ITEM, WIDTH, HEIGHT, VALUE.</i>
<b>CHECKLISTBOX</b>	<b>Multiple-checkboxes list.</b> This element is similar to a list box, except that each item has a check box next to it. Users can check or uncheck items in the list. <i>Mandatory parameters: X, Y; optional parameters: ITEM, WIDTH, HEIGHT, COLUMNS.</i>
<b>HIDDEN</b>	<b>Hidden element.</b> Allows embedded information within the form, which cannot be changed by the client. NAME and VALUE of this element will be sent to the server without modifications. <i>Mandatory parameters: NAME, VALUE.</i>
<b>LABEL</b>	<b>Label.</b> It displays text on a window. Usually this text labels some other element. <i>Mandatory parameters: X, Y, CAPTION, optional parameters: WIDTH, HEIGHT, FOCUSNAME, JUSTIFY, SHOWACCEL, WORDWRAP.</i>
<b>SUBMIT</b>	<b>Submit button.</b> When pressed, sends the contents of the form to the server. <i>Mandatory parameters: X, Y, optional parameters: WIDTH, HEIGHT, CAPTION.</i>
<b>CANCEL</b>	<b>Cancel button.</b> When pressed, closes the form and cancels client input. <i>Mandatory parameters: X, Y, optional parameters: WIDTH, HEIGHT, CAPTION.</i>

### List of Element Parameters

<b>X</b>	Horizontal coordinate of left edge of an element within a form. This is a mandatory parameter for all visible elements: EDIT, MEMO, CHECKBOX, LISTBOX, COMBOBOX, LABEL, SUBMIT, CANCEL, e.g. all except HIDDEN. Numeric value.
<b>Y</b>	Vertical coordinate of top edge of an element within a form. Coordinates are incremented from top to bottom. This is a mandatory parameter for all visible elements: EDIT, MEMO, CHECKBOX, LISTBOX, COMBOBOX, LABEL, SUBMIT, CANCEL, e.g. all except HIDDEN. Numeric value.
<b>WIDTH</b>	Width of an element. The width of all visible elements is set by default. This is an optional parameter for all visible elements: EDIT, MEMO, CHECKBOX, LISTBOX, COMBOBOX, LABEL, SUBMIT, CANCEL, e.g. all except HIDDEN. Numeric value.
<b>HEIGHT</b>	Height of an element. The height of all visible elements is set by default or by font height. This is an optional parameter for all visible elements: EDIT, MEMO, CHECKBOX, LISTBOX, COMBOBOX, LABEL, SUBMIT, CANCEL, e.g. all except HIDDEN. Numeric value.
<b>NAME</b>	Variable name. This is a mandatory parameter for EDIT, MEMO, CHECKBOX, LISTBOX, COMBOBOX. String value.
<b>VALUE</b>	Default value. This is a mandatory parameter for HIDDEN and optional for EDIT, MEMO and CHECKBOX. String value.
<b>CAPTION</b>	Caption. This is a mandatory parameter for LABEL and CHECKBOX and optional for SUBMIT and CANCEL. String value.
<b>FOCUSNAME</b>	Name of element to be focused by a label. It defines the element which links the label to another element - if the value of CAPTION argument of a label includes an accelerator key ("&" character), the element named as the value of this parameter becomes focused when the user presses the accelerator key. This is an optional parameter for LABEL. String value.
<b>JUSTIFY</b>	Justifies label text. This parameter specifies how text is aligned within the label. This is an optional parameter for LABEL. The value of JUSTIFY can be one of: LEFT (align text to the left side of the label), CENTER (center text horizontally in the label), RIGHT (align text to the right side of the label).
<b>SHOWACCEL</b>	Control accelerator character. This parameter determines how an ampersand in the caption of a label appears. Boolean value. If value is ON, an ampersand appears as an underline under the character to its right in the caption indicating the underlined character is an accelerator character. If value is OFF, the ampersand character appears as an ampersand.
<b>ITEM</b>	Defines an item of a list element: LISTBOX, COMBOBOX or CHECKLISTBOX. Have

	two coma-separated string values for LISTBOX, COMBOBOX and three for CHECKLISTBOX. First value defines item name, second - item caption and third argument (applies only to CHECKLISTBOX) defines whether an item is checked (value is empty) or unchecked (value is not empty).
<b>WORDWRAP</b>	Multiple lines. Boolean value. Set WordWrap to ON to allow the label to display multiple line of text. When WordWrap is ON, text that is too wide for the label control wraps at the right margin and continues in additional lines. Set WordWrap to OFF to limit the label to a single line. When WordWrap is OFF, text that is too wide for the label appears truncated.
<b>PASSWORD</b>	Password input. If value is ON, EDIT control displays a special character (asterisk) in place of any entered text.
<b>COLUMNS</b>	Number of columns in a list element. This is an optional parameter for LISTBOX and CHECKLISTBOX elements. Numeric value. Use COLUMNS to specify the number of columns, in a multi-column list box, that are visible without having to use the horizontal scrollbar. Multi-column list boxes have a horizontal scrollbar that allows users to view multiple columns as they wrap. The default value for COLUMNS is 0, meaning that the list box is not multi-column. That is, users can scroll only vertically and the list of items will not wrap. For COLUMNS values greater than 0, multiple columns accommodate the items as they wrap beyond the bottom of the list box. Parameter COLUMNS specifies the number of columns that are visible without having to horizontally scroll the list box. The width of each column depends upon both the WIDTH parameter and the number of columns.
<b>DEFTPL</b>	Default value template. VALUE parameter of the element will be set using <u>macros</u> . Exceptions from standard macros is that # character is used here as a prefix instead of %. E.g. DEFTPL="#FOLDERFROMNAME" as an argument of EDIT element sets VALUE of the element to "FROM" name from the current folder's Identity properties. To be used in EDIT, MEMO and HIDDEN elements.
<b>VBODYRE</b>	Default value: regular expression on message body. Similar to DEFTPL except value is not a template with macros but a regular expression. The matching is done upon text of a currently selected message, e.g. VBODYRE="^UserFullName: (.+)\$".
<b>VFILERE</b>	Default value: regular expression on a particular text file. Similar to VBODYRE except value is extracted from a text file, not message body. A user will be given a standard windows file open dialogue to select a file.

### **MASK Parameter: Character-By-Character Validation**

Use MASK parameter to restrict the characters a user can enter into EDIT element to valid characters and formats. If the user attempts to enter an invalid character, the edit control does not accept the character. Validation is performed on a character-by-character basis. Be sure contents of MASK parameter doesn't start with ampersand (&) character when using character-by-character validation. If contents of MASK parameters starts with ampersand character, it is treated as advanced mask.

A mask consists of three fields with semicolons separating the fields. The first part of the mask is the mask itself. The second part is the character that determines whether the literal characters of a mask are saved as part of the data. The third part of the mask is the character used to represent unentered characters in the mask.

These are the special characters used in the first field of the mask:

- ! If a ! character appears in the mask, optional characters are represented as leading blanks. If a ! character is not present, optional characters are represented as trailing blanks.
- > If a > character appears in the mask, all characters that follow are in uppercase until the end of the mask or until a < character is encountered.
- < If a < character appears in the mask, all characters that follow are in lowercase until the end of the mask or until a > character is encountered.
- <> If these two characters appear together in a mask, no case checking is done and the data is formatted with the case the user uses to enter the data.

<b>\</b>	The character that follows a \ character is a literal character. Use this character to use any of the mask special characters as a literal in the data.
<b>L</b>	The L character requires an alphabetic character only in this position. For the US, this is A-Z, a-z.
<b>l</b>	The l character permits only an alphabetic character in this position, but doesn't require it.
<b>A</b>	The A character requires an alphanumeric character only in this position. For the US, this is A-Z, a-z, 0-9.
<b>a</b>	The a character permits an alphanumeric character in this position, but doesn't require it.
<b>C</b>	The C character requires an arbitrary character in this position.
<b>c</b>	The c character permits an arbitrary character in this position, but doesn't require it.
<b>0</b>	The 0 character requires a numeric character only in this position.
<b>9</b>	The 9 character permits a numeric character in this position, but doesn't require it.
<b>#</b>	The # character permits a numeric character or a plus or minus sign in this position, but doesn't require it.
<b>:</b>	The : character is used to separate hours, minutes, and seconds in times. If the character that separates hours, minutes, and seconds is different in the regional settings of the Control Panel utility on your computer system, that character is used instead.
<b>/</b>	The / character is used to separate months, days, and years in dates. If the character that separates months, days, and years is different in the regional settings of the Control Panel utility on your computer system, that character is used instead.
<b>;</b>	The ; character is used to separate the three fields of the mask.
<b>_</b>	The _ character automatically inserts spaces into the text. When the user enters characters in the field, the cursor skips the _ character.

Any character that does not appear in the preceding table can appear in the first part of the mask as a literal character. Literal characters must be matched exactly in the edit control. They are inserted automatically, and the cursor skips over them during editing. The special mask characters can also appear as literal characters if preceded by a backslash character (\).

The second field of the mask is a single character that indicates whether literal characters from the mask should be included as part of the text.

### **MASK Parameter: Advanced Use**

If contents of MASK parameter starts with ampersand character, it is treated as advanced mask, so no character-by-character validation is done, and the contents are parsed different way: characters following ampersand up to equal sign (or end of parameter) are treated as a keyword. According to a keyword, contents of a EDIT element is checked upon one of the following conditions

<b>Keyword</b>	<b>Condition</b>
<b>URL</b>	Is a valid URL
<b>EMAIL</b>	Is a valid e-mail
<b>DATA</b>	Data is the same as of given control. Useful for validating whether a password is properly confirmed. E.g. MASK="&DATA=UserPasswordAgain=Password is not properly confirmed" compares the string entered by user in a EDIT named UserPasswordAgain and, if the two strings are not bitwise equal, displays Password is not properly confirmed error box.
<b>REGEX</b>	Matches given regular expression. E.g. MASK="&REGEX=^[a-fA-F0-9]{8}\$" matches a string of exactly eight alphanumeric characters.
<b>MANDATORY</b>	Value cannot be empty. If a user leaves empty value, an error box is displayed.

### **Example**

Below is content of a sample TBC-file that can be used to receive currency trend for a stated period. The screen form contains a drop-down list (COMBOBOX) for selecting currency type and fields for date input (EDIT).

```
-----BEGIN PGP SIGNED MESSAGE-----

MSG FORMTITLE="Currency Rate Request"
MSG TO="Client-Bank <client-bank@bank.com>"
MSG SUBJ="Client-Bank"
```

```

MSG PRIORITY="High"
MSG FORMWIDTH=260
MSG FORMHEIGHT=190
MSG ENCRYPT=ON
MSG SIGN=ON

HIDDEN NAME="service" VALUE="getcurrencytrend"

LABEL CAPTION="&Currency" X=16 Y=8 FOCUSNAME="iso"

COMBOBOX NAME="iso" X=16 Y=24 WIDTH=120

COMBOBOX NAME="iso" ITEM="MDL","Moldavian Leu"
COMBOBOX NAME="iso" ITEM="USD","US Dollar"
COMBOBOX NAME="iso" ITEM="RUR","Russian Rouble"
COMBOBOX NAME="iso" ITEM="UAH","Ukrainean Hr."
COMBOBOX NAME="iso" ITEM="FIM","Finnish Mark"
COMBOBOX NAME="iso" ITEM="DEM","Deutsche Mark"
COMBOBOX NAME="iso" ITEM="GBP","GB Pound"

LABEL CAPTION="&Starting Date" X=154 Y=8 FOCUSNAME="date1"
EDIT NAME="date1" X=154 Y=24

EDIT NAME="date1" MASK="00/00/0000" VALUE="01011996"

LABEL CAPTION="&Ending Date" X=154 Y=52 FOCUSNAME="date2"
EDIT NAME="date2" X=154 Y=68
EDIT NAME="date2" MASK="00/00/0000" VALUE="01011999"

SUBMIT CAPTION="OK" X=60 Y=112
CANCEL CAPTION="Cancel" X=154 Y=112

-----BEGIN PGP SIGNATURE-----
iQDVAwUANfjoiRQeqwtQAYtzAQHkIQX+JhO6lHCBfVw5VF3KqQhpApmOPhk2gEKz
C7qkNwpX7lexjbZvTVNGNfxTk7cTx5AxSFiUV32JH9WpM9fuJ5/bXIfWojDnaYbi
OM/BqS2KykFyoiuy+KULji10oRTt1jdCOYqyv+OMgo2RuMwI0QhUozpHlVvKEHkk
r3acZ3NVIjBZHdW0I4YCjA3FKxBQmoFsQg9/ndjeE2lHZTCTcLK2Kue036PeSCJq1
JWwVgWwT0JfhfFMXjj5tZm7pspNaGLst
=FBRX
-----END PGP SIGNATURE-----

```

When the client clicks the OK button, The Bat! generates the following request line:

**service=getcurrencytrend&iso=USD&date1=01011996&date2=01011999**

As values of ENCRYPT and SIGN parameters of MSG keyword are set to ON, this string will be signed and encrypted before being sent to the server via e-mail (to client-bank@bank.com). The e-mail message body will look like this:

```

-----BEGIN PGP MESSAGE-----
hMwDFB6rC1ABi3MBBgBuPwDXPA/obgJJ/fcjdSqO7oh820EHlKpA5mzbYSwg/qWg
wYvNO/iJq4mofSfUnwozK3S5dD4zSeNcEqzQzw3ZZpE5D8aLOlNRAKlRh2ZIfN4W
6pmLJrhcX/GvsH667YUfL97r8LCWbaBtYI8D85ZluPSNGzt8AIgjIEZLKaAcq6yD
eYpO5GpxvvDvT8Vb80QnUZW5kTX9MGFG+gtUGqcb3f3dibywWZfjndhkjj4Br4I
QLQ5v0mjx76lqxhbi5qmaAABM3PfOx1lpLE7VkhCTuQSPsm09a69rjQqgKimOPxj
hcH8wKXAi2FYDQ8wy8isZRZ9IeDHJtqOsdR0vU1tT0Mj0jGvWi1Jb/jFOZ0jW/2J
klufMo5Dy/gaGQjjCIzVrT2dWy7CQ67vnegTi2zsvMAPM6/Bz7qTA6lsMMY88E9p
4qDlcN6OZi1Y8DGAExcIDpoZwPChgIQfA55VQyNfN5YKzfBxCmtqSWbbLFh7WnkR
8FBLi9E8rPGg2jmfAnlCcZynlfXinHtuAKmxmt3m3bOyNv5SGmMz+9WWWhIBH8Q
XIBXrfH2yDDldKwkyj0tsjeXs6LFS7D/kMCdBNM1FC1Ww1EftTBbdwL0ykDX9Z6f
NS6fQxYJxGrfAMP7hSaGydrenSKEviAy5o6HyjBxoCqQLGKdsWQ=
=wflO
-----END PGP MESSAGE-----

```

Upon reception of the message, the server decrypts it, verifies the digital signature, extracts the request line, parses it and sends the currency trend back to the user.

If TBC-file had following strings...

```
MSG TPL=" Service : $service$"  
MSG TPL="Currency : $iso$"  
MSG TPL="      Start : $date1$"  
MSG TPL="      End : $date2$"
```

...query had been sent in a human-readable format, and decrypted message body had been looked like:

```
Service : getcurrencytrend  
Currency : MDL  
Start : 01.01.1996  
End : 01.01.1999
```

If TBC-file had the following strings...

```
MSG NFO="This message contains your currency rates request"  
MSG NFO="that will be sent to the bank. It is PGP-encrypted"  
MSG NFO="to protect sensitive information. When you are online"  
MSG NFO="press Shift+F2 (Send Queued Mail item of Account menu) "  
MSG NFO="to send the message."
```

... the message sent to the server had been looked like:

```
This message contains your currency rates request"  
that will be sent to the bank. It is PGP-encrypted"  
to protect sensitive information. When you are online"  
press Shift+F2 (Send Queued Mail item of Account menu) "  
to send the message."
```

```
-----BEGIN PGP MESSAGE-----  
hMwDFB6rC1ABi3MBBgBuPwDXPA/obgJJ/fcjDSqO7oh820EHlKpA5mzbYSwg/qWg  
wYvNO/iJq4mofSfUnwozK3S5dD4zSeNcEqzQzw3ZZpE5D8aL0lNRAKlRh2ZIfN4W  
6pmLJrhcX/GvsH667YUfL97r8LCWbaBtYI8D85ZluPSNGzt8AIgjIEZLKaAcq6yD  
eYpO5GpxvvDvT8Vb80QnUZW5kTX9MGFG+gtUGqcb3f3dibyewWZfjndhkjj4Br4I  
QLQ5v0mjx76lqxhbi5qmAAABM3PfOx1lpLE7VkhCTuQSPsm09a69rjQqgKimOPxj  
hcH8wKXAi2FYDQ8wy8iszRZ9IeDHJtqOsdR0vUltT0Mj0jGvWi1Jb/jFOZ0jW/2J  
kluFMo5Dy/gaGQjjCIZvRT2dWy7CQ67vnegTi2zsvMAPM6/Bz7qTA6lsMMY88E9p  
4qDlcN6OZiLY8DgaExcIDpoZwPChgIQfA55VQyNfN5YKzfBxCmtqSWbbLFh7WnkR  
8FBLi9E8rPGg2jmfAnlCczYnlfXinHtuAKmxmt3m3bOyNv5SGmMz+9WWhWIBH8Q  
XIBXrfH2yDDldKwkyj0tsjeXs6LFS7D/kMCdBNM1FC1Ww1EftTBbdwL0ykDX9Z6f  
NS6fQxYJxGrfAmp7hSaGydrenSKEviAy5o6HyjBxoCqQLGKdsWQ=  
=wf1O  
-----END PGP MESSAGE-----
```

## File Filter Format for FILEFILTER

This chapter describes format of file filter in FILEFILTER keyword.

A file mask or file filter is a file name that usually includes wildcard characters (\*.TXT, for example). Only files that match the selected file filter are displayed in the dialog box's list box, and the selected file filter appears in the File Name edit box. To specify a file filter, assign a filter string as the value of Filter. To create the string, follow these steps:

- 1 Type some meaningful text that indicates the type of file.
- 2 Type a | character (this is the "pipe" or "or" character).
- 3 Type the file filter.

Don't put in any spaces around the | character in the string.  
Here's an example:

```
Text files|*.TXT
```

If you entered the preceding example as the filter, the string "Text files" appears in the List Files of Type drop-down list box when the dialog window appears, the file filter appears in the File Name edit box, and only .TXT files appear in the list box. You can specify multiple file filters so that a list of filters appears in the List Files of Type drop-down list box or in the filter combo box. This allows the user to select from a number of file filters and determine which files are displayed in the list box.

To specify multiple file filters,

- 1 Create a file filter string as previously shown.
- 2 Type another file filter in the same way, but separate the second file filter from the first with the | character.
- 3 Continue adding as many file filters as you like, separating them with the | character. The string can be up to 255 characters.

Here's an example of three file filters specified as the value of the Filter property:

```
Text files (*.TXT)|*.TXT|Report files (*.RPT)|*.RPT|Output files (*.OUT)|  
*.OUT
```

Now when the dialog box appears, the user can choose from three file filters that appear in the List Files of Type drop-down list box.

Note that the previous example includes the file filters in parentheses in the text parts. This isn't required, but it's a common convention that helps users understand what to expect when they select a file filter.

You can string multiple wildcard file filters together if you separate them with semicolons:

```
All files|*.TXT;*.RPT;*.OUT
```

# Using The Bat! as a Mailing List Server by Leif Gregory

## What is a mailing list?

There are basically two types of mailing lists. The first is a distribution type, and the second is a discussion type. The distribution type is basically a list where you are providing information in one direction. Typical lists of this type are announcements, newsletters and well, SPAM. These lists are very simple to set up, as about the only thing you have to do is to create an address group with all of the recipients in it. A discussion list is quite a bit more complex to set up. Basically, people send posts (e-mail) to a single e-mail address. From there the post is processed and then sent to everyone in an address book group. The complexity lies in setting up the Reply To:, From:, POP3 accounts etc.

## General setup for discussion mailing lists

The easiest way to run a list is to have a separate POP3 account, and as a major plus a forwarding e-mail address (This is a really nice option to have. If you have a forwarding address set up, the e-mail address that subscribers will Un/Subscribe and send posts to will never change. So if you change POP3 servers or give the list over to someone else because you've decided you don't want to run it anymore, you will not impact your subscribers.) To find one of these types of services, just type either "free +email" or "free +POP3" in your favorite search engine. You can set up a discussion list using your existing (main) POP3 account, but it takes quite a bit more work, which I will not cover in this how-to. It will however, generally follow the same guidelines as a stand alone POP3, but your filtering will be a little more complex.

I am going to define some variables so that the below how-to will be much clearer. Please substitute your actual information for that which I have provided.

IDL - The name of our list. It stands for Internet Discussion List

ldgregory@biogate.com - The list moderator's address (Your normal e-mail address)

list@idl.net - The e-mail address that subscribers send their posts and Un/Subscribe requests to.

www.idl.com - The IDL Web site.

IDL <list> - The address book group for normal IDL discussion list subscribers.

IDL Digest <list> - The address book group for digest version IDL list members.

## Procedure for setting up the list using a separate POP3 account

First download from [http://www.rtlabs.com/the\\_bat/idl.zip](http://www.rtlabs.com/the_bat/idl.zip) the IDL mailing list account that I have created. You will be "importing" this to The Bat! Imported is in quotes, because there is no actual provision under The Bat! to import an account. I'll explain how to get this account installed into The Bat! in just a minute. I have done about 90% of the work for you, all you have to do is replace the "IDL" information to suit your discussion list, which we will go through step by step.

How to "import" the IDL account into The Bat!

1. Go to where IDL.ZIP was saved when you downloaded it.
2. Double click the IDL.ZIP file.
3. Click the "Extract" button.
4. In the dialog box that comes up do these two things: The "Extract To" field should be the directory that your

accounts are stored in under The Bat! In my case it is: C:\Program Files\The Bat!\MAIL. The second thing is to check the "Use Folder Names" checkbox if it isn't already checked.

5. If The Bat! isn't already running, start it.
6. Click "Account", select "New".
7. In the "Account Name" field, enter "IDL".
8. Click the "Next" button.
9. Put your name in the "Your Full Name" field.
10. In the "E-mail Address" field put the forwarding e-mail address you set up for your list. If you didn't set up a forwarding e-mail address (You really should use one of these), then put the POP3 e-mail address that you will be using for the list.
11. The "Organization" field can be left blank or put in the name of your discussion list.
12. Click the "Next" button.
13. In the "SMTP Server" field, put whatever SMTP server you will be using. I have mine set up for my regular ISP SMTP server address.
14. In the "POP3 server" field, put the same address as you used in step 10.
15. Click the "Next" button.
16. In the "Username" and "Password" fields, put whatever your username/login is for the POP3 account you are using in step 14.
17. The two checkboxes are up to you, but I have both unchecked. For the normal operation of the discussion list, you don't need to keep copies of messages on the server.
18. Click the "Next" button.
19. Check the "Do you want to check the account settings now" radio button and click the "Finish" button.

Ok, now that you have "imported" the IDL account, we need to finish setting it up for your discussion list.

How to set up the account for your discussion list.

1. Under the "General" sheet, verify that all the information is correct.
2. Under the "Transport" sheet, verify that all the information is correct.
3. For the first few weeks of your list's "life", you should check the "Deferred" option in the "Delivery" options on the "Transport" sheet. The reason for this, is that you want to make sure that everything is correct in the outgoing messages to be sent to your subscribers before they get sent.
4. Under the "Mail Management" sheet, you shouldn't have to change anything.
5. Under the "Options" sheet, set whatever options you prefer to use. You may want to leave the "Empty trash folder on exit" unchecked for the first few weeks of list "life" also. Sometimes you make a mistake in mail handling, and may accidentally delete a message.



6. Under the "Files and Directories" sheet, you may have to modify the "Home Directory" depending on whether you specified a different "Extract To" path in step 4 of the "How to "import" the IDL account into The Bat!".

7. Click the "+" in front of the "Templates" sheet.

8. Select "New Message". We are now going to set this template up. Hopefully you have given some thought as to what you would like your messages to look like when you post to your own discussion list. The merit to having the below boilerplate:

```
--  
*****
```

Report problems to: <mailto:ldgregory@biogate.com>

Check the IDL FAQ: <http://www.idl.com/FAQ>

To unsubscribe, click here:

<mailto:list@idl.net?subject=Unsubscribe\_IDL>

```
*****
```

Is that your subscribers will be reminded each time you post a message to your own discussion list of how to report problems, unsubscribe, and where your discussion list web site or FAQ is. You can make this contain as much or as little information as you want, but I would suggest keeping it short.

Take very careful notice of the two dashes right above the boilerplate. Please note that it is '-- ' (dash dash space). The trailing space is extremely important. What this is for, is that TB cuts off any text following the '-- ' when you reply to a message. How this works in your favor is like this.

You send a message to the list, at the bottom of this message is your boilerplate. Now if you don't have the '-- ', then when someone replies to your message, the boilerplate is quoted. This basically helps you to keep the posts looking nice and clean.

This next block is your macros. Let me show them to you and then I'll explain what they are doing.

```
%TO="IDL Members <list@idl.net>"
```

```
%BCC="IDL <list>"
```

```
%RETURNPATH=""
```

```
%RETURNPATH="ldgregory@biogate.com"
```

The Bat! requires that you have a valid e-mail address in the TO field. Putting the discussion lists e-mail address in here serves two purposes. The first is that it looks better than putting in your e-mail address. The second and most important, is that when the message is sent, you will receive it again.

This sounds stupid, but it is necessary if you wish to verify that your message went out without actually subscribing yourself to your own list, and it allows you to keep the message in its final form just like all the other posts that come from your subscribers. This is useful if you want to create a digest version (which I'll get to later in this article), or if you want to place a searchable message archive on a Web site.

The BCC field must contain your address book group that has all of your subscribers in it.. This is very important. If you put the address book group with all of your subscribers in the TO field, then everyone who is subscribed to your discussion list will see everyone else's e-mail address who is also subscribed to the discussion list. This is considered very bad netiquette.

The RETURNPATH is very important in that it provides an e-mail address for bounced messages to come back to instead of your subscribers. Trust me, you have to have it in there, I learned the hard way with my list. Why are there two of them? Well, there's an undocumented feature in The Bat! that allows you to run a macro with a NULL input. What this does is to clear the field so that the second occurrence of the macro can fill it without worrying about what was stored in it previously. This will become much more easily understood when we get to the filtering section

(filter four).

10. Select the "Reply" template. Again, you'll want include the boilerplate to remind the subscribers when you reply to a subscribers post. Here are the macros:

```
%TO""  
%TO="%OFROMNAME <list@idl.net>"  
%BCC="IDL <list>"  
%RETURNPATH=""  
%RETURNPATH="ldgregory@biogate.com"
```

Here's where the NULL value macro becomes a little easier to understand. Let's say "Billy Bob" <bob@somewhere.com> sends a post. If I want to reply to his message before it gets sent to the list, and send my reply to the list, then I need to change his TO info so that it matches with what we are going to do in the filtering section to all of the messages posted. Namely change the TO, and REPLY-TO information. Again, just hang on a little while and I'll explain this fully in the filters section.

11. This pretty much concludes the account setup. You shouldn't have to use the forward template, and you can put whatever cookies in the cookies template that you want to use.

12. One last thing about the account settings. There are five folders (other than the default ones) included the IDL account. Here's what they are for.

\*\*\*\*\* Note: If you don't see the below folders, select the IDL account, then hold down the CTRL-ALT-SHFT-L keys simultaneously. This will force TB to search for any missing folders in that account.

SPAM - This is the folder I move SPAM to. I'm pretty aggressive in tracking down spammers. If you are interested, I wrote an article for my weekly e-zine entitled "To Catch A Spammer".

Moderated - I've already explained this one, but it is where the messages in their final form (as they were distributed to your subscribers) will reside after the "Move moderated incoming messages" filter processes it.

Failures - This folder will be used to catch bounce messages. Believe me, you will get them. Remember the RETURN-PATH macro? This is what its main purpose is for. Please ensure that the e-mail address you use in RETURN-PATH is different than the one your subscribers use to post and Un/Subscribe to. This will keep the bounce messages from possibly being missed by the filters in your IDL account and getting sent out to your subscribers.

IDL Subscribers - This is the folder that I move the Subscribe requests to, once I have verified that they have been added to the address book. This provides three functions. One is that at a glance, I can tell how many people are subscribed to the discussion list. Two, I have a record of subscribes. This way if someone later on says that I am spamming them, I can prove that they (or at least someone pretending to be them) did indeed subscribe, and three, if someone did not use the correct string to subscribe i.e. they sent a message with the words "Subscribe IDL " in the body of the message as opposed to the subject, you can just copy their e-mail address to the clipboard, right click on the "IDL Subscribers" folder, select "Create a new message" which will create a new message with the "Subscription receipt" in it. All you have to do is paste the subscribers e-mail address into the TO field and send it.

\*\*\* NOTE: I have used text files for inclusion into the templates for both "IDL Subscribes" (both filter and folder), and "IDL Unsubscribes" (both filter and folder) that contain the actual text or the Un/Subscribe. This way, if you make changes, you can do it in one place and it updates both the filter and folder versions. If you specified a different "Extract To" path in step 4 of the "How to "import" the IDL account into The Bat!", then you will need to change the directory path in all the templates (both folder and filter) for IDL Un/Subscribes.

IDL Unsubscribers - Serves the same function as the "IDL Subscribers", but is for unsubscribers. If a subscriber messes up the unsubscription process i.e. puts the words "Unsubscribe IDL" in the body as opposed to the subject, then you can cut and paste their address into the "Create a new message" option when you right click the

"Unsubscribe IDL" folder.

## The Filtering Sub-system

This is the area that will make or break your mailing list. You will need at least four of the five filters that I will be explaining for the basic operation.

1. While the IDL account is highlighted, click "Account" and select "Sorting office/filters"
2. Click the "+" in front of "Incoming messages"

### Filter One - Housekeeping

This filter is a housekeeping filter. Because we are changing the TO header in the "Post" filter (See filter four) to "<list@idl.net>" this message is going to come back to you through the normal post e-mail address. You've already processed this message, so you don't want to send it back out to the subscribers again. This is one other reason we are using the RETURN-PATH macro. It is our filter string, whose job it is to move already processed (sent to the list subscribers) messages to the "Moderated" folder (which by the way is where you should make your replies to messages that subscribers have posted.) If you specified a different "Extract To" path in step 4 of the "How to import" the IDL account into The Bat!, then you will need to change the directory path in this filter to reflect the actual location of the "Moderated" folder.

What we are actually doing with this filter is searching for the string [RETURN-PATH: ldgregory@biogate.com]. This is something we have modified in filter four to solve two problems: The first is bounce messages get sent to the designated address, and two we can use it as our search string, because it will only appear in messages we have already processed.

Of course you could just send these messages to the trashcan, but if you are going to participate in your own discussion list, it is easier to reply to this message than cutting and pasting all their info during the first time it comes through. Also, if you decide later on to build a searchable message-base on a Web site, you will have the message in its final, post filter-processed form.

### Filter Two - Subscribes

The Second filter is the subscription filter. You need to determine what keywords you want to use for your filter string. The easiest, and most logical keywords would be something like "Subscribe" and "IDL". Because not all e-mail clients support the "%20" symbol when resolving a mailto type URL, and also because some do not support actual space characters, I like to use the underscore character between the keywords i.e. "Subscribe\_IDL". This is a good way to ensure that whatever e-mail client your subscribers may be using, your URL will work. The added benefit of using the underscore character is that it is almost totally invisible in a URL, because they are almost always underlined. An example of this URL that you would put on a Web page or in the sig portion of your e-mail messages would look something like this:

To subscribe to the IDL mailing list, click here:  
<mailto:list@idl.com?subject=Subscribe\_IDL>

If you click on this link, The Bat! will create a new message with the subject line filled in with "Subscribe\_IDL", now all the prospective subscriber has to do is click send.

After you have highlighted the filter named "Subscribe\_IDL" do this:

1. Change the filter name in the "Name" field to whatever you wish to call it.
2. Change the filter string in the field "Strings" to whatever you will use as your subscribe filter string.
3. Click the "Alternatives" tab.

What we are going to do here is to try and idiot proof your list a little. Believe me, no matter how hard you try, you will always have people trying to subscribe/unsubscribe from your list using everything but the correct keywords.

The alternatives tab is for "ORing" your filter strings. For more information on this, check the TB help file under "Mail Filtering (Sorting Office)" "Signal Strings".

On this sheet, you will want to change all the different variations of the subscribe signal string to match your list needs.

4. Click the "Actions" tab.
5. Check the "Send Auto-Reply" checkbox.
6. Click the "Template" button.

Here's where you will tell the new subscriber about the list, what to expect, what e-mail address to use to post messages, some rules (concerning flaming, beating a dead horse etc), where to report problems etc. Take some time to think out what you want to put in here. You can use or modify the sample template to fit your needs.

If you read the entire sample, you would have noticed at the bottom there was a macro. It was like this:

```
%SUBJECT="IDL Discussion List Subscription Receipt."
```

Because this is a new message created by the filter, you should specify a subject to be placed in the SUBJECT field. The TO field will have whatever e-mail address is specified in the subscribers REPLY-TO information. Here's something important to remember, this message will contain in the FROM and REPLY-TO fields whatever information you have specified in the account that calls this filter (In this case it's the IDL account settings). This is why you should set up a separate account in The Bat! when running a discussion list. You can specify that your FROM information say something like "IDL Discussion List Moderator <list@idl.net>" and your REPLY-TO info "Leif Gregory <list@idl.net>"

7. Also on the "Actions" sheet, you'll see that I have checked the "Add the sender's address to the address book" option. This option allows new subscribers to be automatically added to your address book group for the list. Make sure you change the name of the address book group listed to the one you are using for the list.

\*\*\*\*\* Note: If you check under the unsubscribe filter (filter three), you'll notice that the option to automatically remove an unsubscriber's address is not enabled. Unfortunately, TB does not currently check to see if the address really exists in the group before deleting it. What this means is if someone tries to unsubscribe from your list, but tries to do it under the wrong e-mail address, then TB will happily comply, but in reality does nothing other than to send them the unsubscription receipt saying they were successfully unsubscribe. Of course this person will probably send you some nasty e-mail when they continue receiving posts to the list. For now, it is better to do it manually.

The way it is set up now, the unsubscribe receipt will still be automatically generated, but you can intervene before it is sent out if the address proved to be incorrect when you manually try to delete them from the address book group.

### **Filter Three - Unsubscribes**

The third filter is your unsubscribe filter. Eventually, someone will want to unsubscribe, so set up this filter similar to the subscribe filter. Of course, you will want to make changes so that it reflects an unsubscribe command. I've included a sample template for you to modify or use as your see fit.

### **Filter Four - Posts**

The fourth filter is the subscriber post filter. This is the real meat of your filtering system, because it allows subscribers to post messages to the rest of the subscribers. This filter will be set up similar to the Un/Subscribe filters, with the following differences:

Under the "Rule" tab, your first filter string should be: [list@idl.net], select the "Recipient" location, and presence is "Yes". This is a SPAM measure. Most times when you receive SPAM, the recipient list has been suppressed, meaning your e-mail address is not displayed along with all the other unlucky recipients of the SPAM. So by specifying "list@idl.net" you will be eliminating a huge majority of SPAM from making it to your subscribers.

The second filter string should be: [Subscribe\_IDL], select the "Subject" location, and presence is "No". This will ensure that Subscribe requests don't get sent to the rest of your subscribers.

The third filter string should be: [Unsubscribe\_IDL], select the "Subject" location, and presence is "No". This will ensure that unsubscribe requests also don't get sent.

Under the "Actions" tab scroll down till you see "Create a message for", put your name and e-mail address here like this: "Leif Gregory"<list@idl.net> Now click the template button. Here is what the stuff in the template means:

The %TEXT is going to cut and paste the posters text into this message. You don't want to use the %QUOTE macro, because that's not what you want.

The three blank macros "TO, REPLYTO, and FROM" are an undocumented feature in The Bat! What happens when you don't specify any text between the quotes is that it blanks out the field (A NULL value.) Remember when you put your name and e-mail address in the field under the action "Create a message for"? Well, if you don't blank out this field, that info will appear here in addition to the info placed there by the second occurrence of "TO" in the above macro. This is a bad thing. The REPLYTO and FROM fields will contain the information you specified in the IDL account settings, so we want to blank these out.

Ok, now on to the second occurrences of TO, REPLYTO and FROM. When a subscriber posts a message, and the other subscribers receive it, you want the TO, FROM and REPLYTO information to reflect the person who posted the message. i.e. say Billy Bob <bboob@user.com> posts a message to the list. When a subscriber receives the message, you want the TO field to say "IDL Discussion List <list@idl.net>" (unless they were replying to someone's message, then you would want it to have the repliees address here.), you want the REPLYTO field to say "Billy Bob <list@idl.net>" (This is extremely important! The REPLYTO header is the address that is used when someone replies to a message. We want the replies to come back to the list, so that they will get sent to all the other subscribers too. If the replyer wants to send a private message to only Billy Bob, then they should cut and paste the address in the FROM field), which brings up what you want displayed in the FROM field. You want it to say "Billy Bob <bboob@user.com>" (You want the subscribers real e-mail address here.) With that out of the way, let's dissect the macros.

%TO="%OTONAME <list@idl.net>" Remember when I said that you wanted the TO field to display "IDL Discussion List" unless the post is a reply? Here's what happens in this macro. %OTONAME takes all the text except the actual e-mail address i.e. "IDL Discussion List", or in the case of a reply "Billy Bob" and puts it into the TO field. We are then specifying that the e-mail address be <list@idl.net>. So the new TO address will be "IDL Discussion List <list@idl.net>" or in the case of a reply "Billy Bob <list@idl.net>" .

%ReplyTo=%OFROMNAME <list@idl.net> When a subscriber replies to a message, this is the actual e-mail address it will use. We are taking the original posters name and changing the e-mail address so that the reply gets sent to the list instead of the original poster personally.

%From="%OFROMNAME <%OFROMADDR>" Here we are using all of the original posters real information. Their real name and real e-mail address.

%BCC="IDL <list>" This is your address book group entry. Of course we do not want to show everyone on the list who else is also subscribed to the list, so we put it in the Blind Carbon Copy (BCC) field. This also help to prevent spamming of your subscribers.

### **Filter Five - Move sent messages to trash**

This is the filter can do without if you don't want it. However, I would highly recommend it, otherwise the sent posts will accumulate in your "sent" mail folder, and you would have to move them to the trash manually.

## Setting up the digest version

Right now, this is one area when The Bat! is kind of weak. I have asked RIT Labs to try and work on this area just a little more. I would have included this section in the filters section, but I didn't feel that it has the proper capability to really be used as a digest generator. There is an option on the "Actions" tab for "Export message to file". The problem is that you can either include all of the kludges (really messy looking), or none of the kludges (difficult to reply to people or to see who wrote the message and it's subject.) I asked RIT Labs if they could allow you to select which kludges to include in the exported message. I think that these kludges are necessary (DATE, FROM, REPLYTO, TO and SUBJECT,). If you wish to experiment around with the digest generator anyway, do this:

In the IDL filters, click the "+" in front of "Incoming messages".

Select "Move moderated incoming messages".

Select the "Actions" tab.

Scroll down till you see "Export message to file".

Put a checkmark in the box.

Click the "Browse" button to the right of the pathname field.

I have already selected the "TBDigest.TXT" file. I'd suggest keeping a copy of all the digests you create for future use (on a web site etc).

If you specified a different "Extract To" path in step 4 of the "How to "import" the IDL account into The Bat!", then you will need to change the directory path to reflect your directories.

Put a checkmark in the "Append to existing file" radio button.

Exit out of the filters dialog window.

To send a digest to someone (You'll have to do this manually, as there is no way currently to define to TB when to send these digests out.) right click on the "Digest" folder, select "Create a new message". You should be able to just click send from here.

Remember to empty out the "TBDigest.TXT" after sending it, because then you'll just append more messages to the end.

Also remember to set up an address book group called "IDL Digest" (Or change it to whatever.)

## Closing remarks and how to contact the author

Ok, we're done. You have all of the stuff necessary to start your own discussion list. If you have any additional questions or find any faults in this how-to, please e-mail me at:

"Leif Gregory <ldgregory@pcwize.com>"

I will be glad to answer any questions you may have.

## Troubleshooting

No matter how hard you try, there is always something that didn't get covered. Here are the answers to some common problems people using this how-to have come up against.

**The RETURN-PATH doesn't seem to be working / posts that have already been processed and sent out to the list are being treated like new posts when they come back through the system.**

Most likely the SMTP or POP server you are using is removing the RETURN-PATH statement from the header. Some are set up this way, and you might be able to get the administrator of that server to change that. If not, then you will most likely have to find a different SMTP or POP server.

To determine which is actually the culprit, create a new message from the IDL list, remove the addresses from the TO and BCC fields (you might have to enable the viewing of these fields first by clicking "View" in the message editor window and putting a check mark next to them.) Once you have cleared out these addresses, put my address <ldgregory@pcwize.com> in the TO field and send it to me. Of course, please type something in telling me why you are sending the message to me. I'll check the headers, and if it is missing, then the culprit is your SMTP server. I will then send you a reply that has a RETURN-PATH statement in it as well, you check the headers to see if it is there. If not, then your POP server is deleting them as well.

*I've got a problem with a person who used to be subscribed, but I unsubscribed them because they either didn't follow the list rules or were causing problems. Even though they are unsubscribed, they can still post to the list.*

Unfortunately, there is no way to have TB check to see if a post is coming from a person in the subscriber address book group. The only thing I can tell you is to put a filter in the incoming filter section to move their messages to the trash. Make sure you put that filter at the top of the list using the "move up" button. TB checks messages against filters in the order they are listed. Therefore, if you don't put it above at least filter four, then their messages will still get processed

Copyright © 1998-1999 Leif Gregory. All rights reserved.

## About The Bat! Command line parameters

After many discussions and tests, we have decided that the best way of interacting with other programs with The Bat! is the old good Command Line. The command line is a text string that is passed to the system whenever any program is executed; it contains the path to the program with the following set of parameters. In Windows, you can start a program via Start Menu “Run...” command or by clicking on a Shortcut on the Desktop or in Start menu. If you start a program by clicking a Shortcut, you can edit the command line for the program in Shortcut Properties’ “Target” input box.

For The Bat!, the Command Line in a Shortcut usually looks like this:

```
“C:\Program Files\The Bat!\TheBat.EXE”
```

You can add some start-up parameters to it to define a set of actions performed whenever you open the Shortcut. For example, if the Shortcut is located in the Windows Startup folder, you may want to switch The Bat! startup Logo off by adding /NOLOGO parameter, so the “Target” property of the shortcut becomes something like this:

```
“C:\Program Files\The Bat!\TheBat.EXE” /NOLOGO
```

We have implemented a set of Command Line parameters (we call them commands) helping to perform some actions within The Bat! without doing this manually, so it is possible to use The Bat! from batch files or other programs as an e-mail transport without knowing details of Internet mailing and for performing automatically scheduled tasks (for example, checking for new mail at particular time and then exiting so The Bat! would not use your system’s resources for a long time).

Note that you can run only one copy of The Bat! at your PC at a time, but if you try to start another copy of The Bat!, all Command Line parameters will be seamlessly passed to the running copy of the program and executed.

For those who are familiar with programming we give a hint: The Bat! checks the file called TheBat.IPC which is located in the same directory where The Bat! executable (TheBat.exe) file is located. In this text file, each line represents one command that could be executed by the program, so you can write commands to this file directly and control The Bat! on your PC even over a local network. Note that TheBat.IPC is deleted as soon as all commands from it are executed, so if this file do not exist, you must create it.

To ensure that a copy of The Bat! is running on a computer, a program which wants to interact with The Bat! must check whether the mutex called “The Bat!” is owned by a process and if it is not, The Bat! must be started.



## Checking for new mail - /CHECK and /CHECKALL commands

/CHECK can be used as a command line parameter for The Bat! whenever you want to check mail for one or several accounts.

/CHECKALL command is used for checking mail for all accounts and exiting from The Bat! if there were no new messages received. It is a shortcut for the combination of /CHECK\* /SMARTEXTIT

The syntax of /CHECK command is:

**/CHECK<account mask1>[;account mask2[;account mask3[...]]]**

Account mask is used for identification of a single or multiple accounts for performing the operation. It can be:

- the full name of a single account. Example: /CHECK"My account 1";"My account 2". This instructs The Bat! to check new mail for two accounts with names "My account 1" and "My account 2" (note that quotation marks are not included into accounts' names – they are used only because the names contain space characters)
- the beginning of a name ending with asterisk. Example: to check new mail for all accounts with names starting with word "My", use /CHECKmy\*
- asterisk with following ending of a name. Example: /CHECK\*1 will check new mail for all accounts with names ending with "1".
- a name fragment included in asterisks. Example: /CHECK\*account\*
- an asterisk which mean that all accounts will be checked

### **Notes:**

1. To separate account masks, use semicolon (";" character). Do not use spaces between account masks when use /CHECK command as a command line parameter because a space-separated mask will be treated as the next command line parameter and will not be processed.
1. If a mask contains space characters, put it into quotation marks as it shown in the first example. If a mask contains quotation marks, you should use upper commas (" " " character).

## Sending queued mail - /SEND and /SENDALL commands

/SEND can be used as a command line parameter for The Bat! whenever you want to send all queued mail from one or several accounts.

/SENDALL command is used for sending queued mail from all accounts and exiting from The Bat! if there were no new messages received. It is a shortcut for the combination of /SEND\* /SMARTEXTIT

The syntax of /SEND command is:

**/SEND<account mask1>[;account mask2[;account mask3[...]]]**

Account mask is used for identification of a single or multiple accounts for performing the operation. It can be:

- the full name of a single account. Example: /SEND"My account 1";"My account 2". This instructs The Bat! to send queued mail from two accounts with names "My account 1" and "My account 2" (note that quotation marks are not included into accounts' names – they are used only because the names contain space characters)
- the beginning of a name ending with asterisk. Example: to send queued mail from all accounts with names starting with word "My", use /SENDmy\*
- asterisk with following ending of a name. Example: /SEND\*1 will send queued mail from all accounts with names ending with "1".
- a name fragment included in asterisks. Example: /SEND\*account\*
- an asterisk which mean that queued mail will be sent from all accounts

### **Notes:**

1. To separate account masks, use semicolon (";" character). Do not use spaces between account masks when use /SEND command as a command line parameter because a space-separated mask will be treated as the next command line parameter and will not be processed.
1. If a mask contains space characters, put it into quotation marks as it shown in the first example. If a mask contains quotation marks, you should use upper commas ("'" character).

## Import messages - /IMPORT command

/IMPORT command allows batch importing of e-mail messages into a specified folder from multiple RFC-822 message files or from UNIX mailbox files. The syntax of /IMPORT command is:

**/IMPORT**[parameter1[;parameter2[;parameter3[...]]]

Possible parameters are (parameter may be identified by two or more names):

- |  |  |
|--|--|
| <b>USER</b> = <i>value</i> or <b>U</b> = <i>value</i>  | - the <i>value</i> is the name of the destination account. If no <b>FOLDER</b> parameter is specified, the destination folder will be Inbox of the given account.  |
| <b>PASSWORD</b> = <i>value</i> or <b>P</b> = <i>value</i>  | - the <i>value</i> is the password which will unlock the account if it is needed.  |
| <b>FOLDER</b> = <i>value</i> or <b>F</b> = <i>value</i>  | - the <i>value</i> is the destination folder's pathname. If the pathname does not include account name, The Bat! will search all accounts for a folder with such name; the first folder found will be used as the destination folder. If the specified folder is not found, Inbox folder of the destination account is used. |
| <b>UNIX</b> or <b>X</b>  | - this parameter tells The Bat! that input files are in UNIX mailbox format. By default, input files are treated as separate RFC-822 messages.   |
| <b>READ</b> or <b>R</b>  | - when this parameter is used, all imported messages will be marked as read. By default, all imported messages are marked unread.  |
| <b>FILE</b> = <i>value</i> or <b>IN</b> = <i>value</i> or<br><b>INFILE</b> = <i>value</i> or <b>I</b> = <i>value</i> | - the <i>value</i> is a file mask with a pathname of input files. /IMPORT command can have unlimited number of <b>FILE</b> parameters.   |
| <b>DELETE</b> or <b>DEL</b> or <b>W</b>  | - if this parameter is specified, all processed files will be deleted after successful send of import operation.   |

### Examples of /IMPORT command:

/IMPORTU="My account 1";FOLDER="Friends and relatives\Sam";IN=C:\InFiles\Sam\\*.MSG

/IMPORTF="\\My account 1\Business\Unsorted";UNIX;FILE=C:\InFiles\Unsorted\\*.mbx;READ

### Notes:

1. To separate parameters, use semicolon (“;” character). Do not use spaces between parameters when use /IMPORT command as a command line parameter because a space-separated mask will be treated as the next command line parameter and will not be processed.
1. If a parameter value contains space characters, put it into quotation marks. If a value contains quotation marks, you should use upper commas (“ ” character).

## Export messages or addresses - /EXPORT command

/EXPORT command allows batch exporting of e-mail messages from a specified folder to multiple RFC-822 message files or to UNIX mailbox files. This command also allows exporting of address book entries if the **LDIF** parameter is specified. The syntax of /EXPORT command is:

**/EXPORT[parameter1[:parameter2[:parameter3[...]]]**

Possible parameters are (parameter may be identified by two or more names):

- |   |   |
|---|---|
| <b>USER</b> = <i>value</i> or <b>U</b> = <i>value</i>   | - the <i>value</i> is the name of the source account. If no <b>FOLDER</b> parameter is specified, the source folder will be Inbox of the given account.   |
| <b>PASSWORD</b> = <i>value</i> or <b>P</b> = <i>value</i>   | - the <i>value</i> is the password which will unlock the account if it is needed.   |
| <b>FOLDER</b> = <i>value</i> or <b>F</b> = <i>value</i>   | - the <i>value</i> is the source folder's pathname. If the pathname does not include account name, The Bat! will search all accounts for a folder with such name; the first folder found will be used as the source folder. If the specified folder is not found, Inbox folder of the source account is used.   |
| <b>DIR</b> = <i>value</i> or <b>D</b> = <i>value</i> or<br><b>OUT</b> = <i>value</i> or <b>O</b> = <i>value</i> | - the <i>value</i> is the pathname of the output directory (for RFC-822 messages) or the output file (for UNIX mailbox). If RFC-822 format is chosen, the exported messages are stored in the output directory in files with names xxxxxxxx.MSG (each "x" character corresponds to a digit from 0 to 9). When export starts, The Bat! calculates the starting xxxxxxxx number by searching the output directory for the files with names in the same format and, in case when such files are found, the output file names will represent greater numbers than the greater found. For example, if file 00001234.MSG was in the output directory, the first exported message will be named as 00001235.MSG. |
| <b>UNIX</b> or <b>X</b>   | - this parameter tells The Bat! that output file must be in UNIX mailbox format. In this case, <b>DIR</b> parameter specifies the name of the output file. By default, output files are in RFC-822 message format.  |
| <b>READ</b> or <b>R</b>   | - when this parameter is used, only read messages are to be exported. By default, all read and unread messages are to be exported.  |
| <b>UNREAD</b> or <b>N</b>   | - when this parameter is used, only unread messages are to be exported.   |
| <b>REPLIED</b> or <b>RE</b>   | - when this parameter is used, only replied messages are to be exported. By default, all replied and non-replied messages are to be exported.   |
| <b>UNREPLIED</b> or <b>NR</b>   | - when this parameter is used, only non-replied messages are to be exported.  |
| <b>PARKED</b> or <b>PR</b>  | - when this parameter is used, only parked messages are to be exported. By default, all parked and non-parked messages are to be exported.  |
| <b>UNPARKED</b> or <b>UP</b>  | - when this parameter is used, only non-parked messages are to be exported.   |
| <b>MAXAGE</b> = <i>value</i> or <b>AGE</b> = <i>value</i> or<br><b>A</b> = <i>value</i>                         | - the <i>value</i> specifies the maximum age (in days) of exported messages. If age of a message exceeds the specified maximum age, the message will not be exported. By default the maximum age is not limited.  |

<b>START</b> = <i>value</i> or <b>S</b> = <i>value</i>	- the <i>value</i> specifies the number of the starting message in the source folder. All messages located before the starting messages will not be exported. If a negative value is given, the starting message number is calculated by subtracting the positive value from number of messages in folder, for example, if -5 is given, this means that export will start from fifth message from the end of message base.
<b>END</b> = <i>value</i> or <b>E</b> = <i>value</i>	- the <i>value</i> specifies the number of the ending message in the source folder. All messages located after the ending messages will not be exported. If a negative value is given, the ending message number is calculated by subtracting the positive value from number of messages in folder, for example, if -2 is given, this means that export will end at second message from the end of message base.
<b>OVERRIDE</b> or <b>V</b>	- (only when either <b>UNIX</b> or <b>LDIF</b> parameter is specified) if this parameter is specified, The Bat! will override the output file in case when it exists. By default, The Bat! adds new messages to the end of the mailbox file.
<b>DELETE</b> or <b>DEL</b> or <b>W</b>	- if this parameter is specified, all processed messages will be deleted after successful end of export operation.
<b>LDIF</b>	- if this parameter is specified, export of addresses will be performed. The <b>OUTPUT</b> parameter must specify the name of the output file that will receive exported data in LDIF format.
<b>ADDRESSBOOK</b> = <i>value</i> or <b>AB</b> = <i>value</i> or <b>BOOK</b> = <i>value</i>	- (only with the <b>LDIF</b> parameter) the <i>value</i> specifies name, filename or the full path of the address book which should be exported. If no address book specified, The Bat! will use the default address book.
<b>GROUP</b> = <i>value</i> or <b>G</b> = <i>value</i>	- (only with the <b>LDIF</b> parameter) the <i>value</i> specifies the handle of the address group which should be exported. If no address book explicitly specified, The Bat! will search all address books for the first group with the specified handle.

### Examples of /EXPORT command:

/EXPORTU="My account 1";F="Friends\Sam";DIR=C:\InFiles\Sam\S=-20

/EXPORTF="\\Account1\Business\Unsorted";UNIX;O=C:\InFiles\Unsorted\Mail.mbx;UNREAD

/EXPORTLDIF;AB="My Book";Group="My Group";O="C:\MyGroupFile.LDIF"

/EXPORTLDIF;Group="My Test Group";O="C:\MyTestGroup.LDIF"

### Notes:

1. To separate parameters, use semicolon (“;” character). Do not use spaces between parameters when use /EXPORT command as a command line parameter because a space-separated mask will be treated as the next command line parameter and will not be processed.
1. If a parameter value contains space characters, put it into quotation marks. If a value contains quotation marks, you should use upper commas (“ ” character).

## Focusing a folder - /FOCUS command

/FOCUS command allows to automatically focus the specified folder in the main window. This command is especially useful at the program's startup. The syntax of /FOCUS command is:

**/FOCUS**[parameter1[;parameter2[;parameter3[...]]]

Possible parameters are (parameter may be identified by two or more names):

- USER**=*value* or **U**=*value* - the *value* is the name of the source account. If no **FOLDER** parameter is specified, the target folder will be Inbox of the given account.
- PASSWORD**=*value* or **P**=*value* - the *value* is the password which will unlock the account if it is needed.
- FOLDER**=*value* or **F**=*value* - the *value* is the target folder's pathname. If the pathname does not include account name, The Bat! will search all accounts for a folder with such name; the first folder found will be used as the target folder. If the specified folder is not found, Inbox folder of the target account is used.

### Examples of /FOCUS command:

```
/FOCUSU="My account 1";F="Friends\Sam"  
/FocusF="\\MyAccount\New mail";P=mypass
```

### Notes:

1. To separate parameters, use semicolon (“;” character). Do not use spaces between parameters when use /FOCUS command as a command line parameter because a space-separated mask will be treated as the next command line parameter and will not be processed.
1. If a parameter value contains space characters, put it into quotation marks. If a value contains quotation marks, you should use upper commas (“ ” character).

## Automated message creation - /MAIL command

/MAIL command is used for automated message creation from a template, text file and/or set of attached files for a specific address. This command is extremely useful for applications that require sending e-mail messages without making a lot of additional work connected to implementation of Internet standards. The syntax of /MAIL command is:

**/MAIL**[parameter1[;parameter2[;parameter3[...]]]

Possible parameters are (parameter may be identified by two or more names):

- |   |   |
|---|---|
| <b>USER</b> = <i>value</i> or <b>U</b> = <i>value</i>                                   | - the <i>value</i> is the name of the source account. If no <b>FOLDER</b> parameter is specified, the target folder will be Inbox of the given account.   |
| <b>PASSWORD</b> = <i>value</i> or <b>P</b> = <i>value</i>                               | - the <i>value</i> is the password which will unlock the account if it is needed.   |
| <b>FOLDER</b> = <i>value</i> or <b>F</b> = <i>value</i>                                 | - the <i>value</i> is the target folder's pathname. If the pathname does not include account name, The Bat! will search all accounts for a folder with such name; the first folder found will be used as the target folder. If the specified folder is not found, Inbox folder of the target account is used. |
| <b>TEMPLATE</b> = <i>value</i> or <b>T</b> = <i>value</i>                               | - the <i>value</i> is the pathname of the file which contains the template which must be used for the message creation. By default, it is the template of the target folder or the target account.  |
| <b>TO</b> = <i>value</i>  | - the <i>value</i> specifies the primary addressee of the message. You can add additional addressees by template macros %TO, %CC, %BCC.   |
| <b>SUBJECT</b> = <i>value</i> or <b>S</b> = <i>value</i>                                | - the <i>value</i> specifies the subject of the message. It is also possible to define message subject in the template using %SUBJECT macros.   |
| <b>TEXT</b> = <i>value</i> or <b>CONTENTS</b> = <i>value</i> or <b>C</b> = <i>value</i> | - the <i>value</i> is the pathname of the plain text file which contains the text of the message. It is also possible to include a text file into a message using %PUT macro in the template.   |
| <b>ATTACH</b> = <i>value</i> or <b>FILE</b> = <i>value</i> or <b>A</b> = <i>value</i>   | - the <i>value</i> is the pathname of the file which must be attached to the message. It is also possible to use %ATTACHFILE macros in the template.  |
| <b>SEND</b>   | - if this parameter is used, the created message will be immediately sent out once it is created.   |
| <b>QUEUE</b>  | - if this parameter is used, the created message will be queued in the Outbox once it is created.   |

### Examples of /EXPORT command:

/MAILU=MyAccount;TO=some@address.com;S=Test;TEXT=C:\TESTs\TEST.MSG

/MAILF=\\MyAccount\Test;TO=some@address.com

***Notes:***

1. To separate parameters, use semicolon (“;” character). Do not use spaces between parameters when use /EXPORT command as a command line parameter because a space-separated mask will be treated as the next command line parameter and will not be processed.
1. If a parameter value contains space characters, put it into quotation marks. If a value contains quotation marks, you should use upper commas (“ ’ ” character).



## Interactive message creating – mailto: command

**mailto:** command is used to force The Bat! to open a new editor window with some pre-set parameters. **mailto:** is a standard Internet URL for publishing e-mail addresses and is described in RFC 2368. This command line parameter is usually used by browsers which need to open an e-mail editor when user clicks on a link which contains mailto: URL.

## Execution of multiple commands from a file - /BATCH command

If you need to execute a lot of commands many times, /BATCH command is the saver. This command allows executing multiple commands defined in a text file; each command is placed in one line. The syntax of /BATCH command is:

**/BATCH:<filepath>**

For example, if you have to check your account and send queued mail from it, you can create a batch file **C:\The Bat!\CheckMail.BAT** which will contain three lines:

```
/CHECKMyAccount  
/SENDMyAccount  
/SMARTEXIT
```

To execute this batch from a command line, you can run The Bat! with command line parameter **/BATCH:C:\The Bat!\CheckMail.BAT**

## **Miscellaneous commands - /MINIMIZE, /EXIT, /SMARTEXIT and /NOLOGO**

All of the commands below can be sent to The Bat! at the program's start-up or during its operation.

**/EXIT** command is used to force The Bat! to exit as soon as all mail transfer operations are finished.

**/SMARTEXIT** command is used to force The Bat! to exit as soon as all mail transfer operations are finished and if there were no new messages received.

**/MINIMIZE** command forces The Bat! to minimise.

**/NOLOGO** command can be used only at start-up of The Bat!. When it is used in the program's start-up command line, The Bat! Logo screen will not be displayed while program is loading its data. This feature can be used when The Bat! should run at your system's start-up automatically.

## Quick Templates

Quick Templates are designed for saving hours of typing similar blocks of text in your messages. This feature is extremely useful especially when you do any kind of support via e-mail and have to answer on similar questions. With Quick Templates, it is also possible to attach files, vCard and change other settings of a message without quitting typing a message and using menus. Note that all template macros are working in Quick Templates just like in other templates used within The Bat!

Each Quick Template has a unique handle, which should be easily remembered by you. When you want to insert a Quick Template in your message, just type its handle in the text and press Ctrl+Space – the typed handle will be replaced with the text block accordingly to the used template. If you do not remember the handle, you still can use the “Utilities | Insert Quick Template” menu of the message editor.

Quick Templates are associated with an account, so you can edit them from Account Properties dialogue: at the Templates page, press the Edit Quick Templates” button. It is also possible to edit Quick Templates using the “Options | Quick templates” menu command from the main program window. If you want to share a Quick Template between other accounts, tick the appropriate check box in the template editor.

## Template Macros

A Template Macro is a special command to the template processor that can generate text, set some message parameters etc. The macro when it is inserted into a template always starts with the “%” symbol followed by the name and an optional parameter in double quotes. Here is the complete list of possible names:

-           - the dash macro (%-) used at the end of a template line means that the next line must be added to the end of the current line (this helps to make templates more readable). An empty string replaces the dash macro used in the middle of a line.

<b>TONAME</b>	- insert the <b>full</b> name of the first TO addressee
<b>TOFNAME</b>	- insert the <b>first</b> name of the first TO addressee
<b>TOLNAME</b>	- insert the <b>last</b> name of the first TO addressee
<b>FROMNAME</b>	- insert the <b>full</b> name of the sender
<b>FROMFNAME</b>	- insert the <b>first</b> name of the sender
<b>FROMLNAME</b>	- insert the <b>last</b> name of the sender
<b>REPLYNAME</b>	- insert the <b>full</b> name of the reply recipient
<b>REPLYFNAME</b>	- insert the <b>first</b> name of the reply recipient
<b>REPLYLNAME</b>	- insert the <b>last</b> name of the reply recipient
<b>OTONAME</b> message	- (replies and forward) insert the <b>full</b> name of the first TO addressee of the original message
<b>OTOFNAME</b> message	- (replies and forward) insert the <b>first</b> name of the first TO addressee of the original message
<b>OTOLNAME</b> message	- (replies and forward) insert the <b>last</b> name of the first TO addressee of the original message
<b>OFROMNAME</b>	- (replies and forward) insert the <b>full</b> name of the sender of the original message
<b>OFROMFNAME</b>	- (replies and forward) insert the <b>first</b> name of the sender of the original message
<b>OFROMLNAME</b>	- (replies and forward) insert the <b>last</b> name of the sender of the original message
<b>OREPLYNAME</b>	- (replies and forward) insert the <b>full</b> name of the reply recipient of the original message
<b>OREPLYFNAME</b>	- (replies and forward) insert the <b>first</b> name of the reply recipient of the original message
<b>OREPLYLNAME</b>	- (replies and forward) insert the <b>last</b> name of the reply recipient of the original message
<b>TOADDR</b>	- insert the first TO e-mail address
<b>FROMADDR</b>	- insert the address of the sender
<b>REPLYADDR</b>	- insert the address of the reply recipient

<b>SUBJ</b>	- insert the subject of the message without Re: and Fwd: prefixes
<b>OSUBJ</b> prefixes	- (replies and forward) insert the subject of the original message without Re: and Fwd: prefixes
<b>FULLSUBJ</b>	- insert the full subject of the message
<b>OFULLSUBJ</b>	- (replies and forward) insert the full subject of the original message
<b>OTOADDR</b>	- (replies and forward) insert the first TO address of the original message
<b>OFROMADDR</b> <b>OREPLYADDR</b>	- (replies and forward) insert the address of the sender of the original message - (replies and forward) insert the address of the reply recipient of the original message
<b>ORGANIZATION,</b> <b>ORGANISATION</b>	- insert the organisation name
<b>OORGANIZATION,</b> <b>OORGANISATION</b>	- (replies and forward) insert the organisation name from the original message
<b>MSGID</b>	- insert the current message's Message ID (taken from the message headers, not working for newly created messages)
<b>OMSGID</b>	- insert the original message's Message ID (taken from the message headers)
<b>TOLIST</b>	- insert the full list of recipient in the <b>TO</b> address field
<b>CCLIST</b>	- insert the full list of recipient in the <b>CC</b> address field
<b>BCCLIST</b>	- insert the full list of recipient in the <b>BCC</b> address field
<b>OTOLIST</b> original message	- (replies and forward) insert the full list of recipient in the <b>TO</b> address field of the original message
<b>OCCLIST</b> original message	- (replies and forward) insert the full list of recipient in the <b>CC</b> address field of the original message
<b>OBCCLIST</b> original message	- (replies and forward) insert the full list of recipient in the <b>BCC</b> address field of the original message
<b>DOW</b>	- insert the current day of week
<b>ODOW</b>	- (replies and forward) insert the day of week of the original message
<b>CURSOR</b>	- place the caret the message editor in the position of this macro. This macro does nothing for automatically generated messages.
<b>TEXT</b>	- insert the text of the original message
<b>TEXT="n"</b>	- insert <i>n</i> first lines of of the original message text
<b>TEXT="n L"</b>	- insert <i>n</i> first lines of of the original message text and the lines with URLs.
<b>QUOTES</b>	- insert the quoted original message text

**QUOTES**="string" - insert quoted *string*

**WRAPPED**="text" - insert the *text* wrapped accordingly to the default editor settings. It is possible to use template macros included to this list in the *text*. This macro is very usable when you use a lot of macros to generate some text and want the generated text to be nicely laid out. It is recommended to place the %WRAPPED="..." macro at the beginning of a new line.

**HEADERS** - insert the RFC-822 headers of the original message

**QHEADERS** - insert the quoted RFC-822 headers of the original message

**COOKIE** - insert a random cookie from the current account's cookies

**COOKIE**="filepath" - insert a random cookie from the file defined by the *filepath*. This allows using virtually unlimited amount of cookies.

**CLIPBOARD** - insert the textual contents of the system Clipboard

**QUOTECLIPBOARD** - insert the quoted textual contents of the system Clipboard

**DATE** - insert the current date in the **long** date format (defined by the system's country settings)

**DATESHORT** - insert the current date in the **short** date format (defined by the system's country settings)

**ODATE** - (replies and forward) insert the date of the original message in the **long** date format (defined by the system's country settings)

**ODATESHORT** - (replies and forward) insert the current date of the original message in the **short** date format (defined by the system's country settings)

**TIME** - insert the current time in the **short** time format (defined by the system's country settings)

**TIMELONG** - insert the current time in the **long** time format (defined by the system's country settings)

**OTIME** - insert the time of the original message in the **short** time format (defined by the system's country settings)

**OTIMELONG** - insert the time of the original message in the **long** time format (defined by the system's country settings)

**DATEEN** - insert the current date in "**day-of-week, month day, year**" format. The day-of-week and month are full words in English

**ODATEEN** - insert the date of the original message in "**day-of-week, month day, year**" format. The day-of-week and month are full words in English

**TIMELONGEN** - insert the current time in "**hour:minutes:seconds AM/PM**" format

**OTIMELONGEN** - insert the time of the original message in "**hour:minutes:seconds AM/PM**" format

**REPLYCOUNTER, RECOUNT** - insert the numbered reply counter of the current message. This can be used in %SUBJECT macro: %SUBJECT="%RECOUNT Why don't we change the subject?"

**OREPLYCOUNTER,**  
**ORECOUNT** - insert the numbered reply counter of the original message.

**TO=""** - remove all recipients from the **TO** field

**TO="address(es)"** - add address(es) to the **TO** field

**CC=""** - remove all recipients from the **CC** field

**CC="address(es)"** - add address(es) to the **CC** field

**BCC=""** - remove all recipients from the **BCC** field

**BCC="address(es)"** - add address(es) to the **BCC** field

**FROM="address"** - redefine the default **FROM** field contents with the *address* data

**REPLYTO="address"** - redefine the default **REPLY-TO** field contents with the *address* data

**RETURNPATH="address"** - redefine the default **RETURN-PATH** field contents with the *address* data

**SUBJECT="subject"** - set the new subject to the message

**ORG="organisation"** - redefine the default **ORGANISATION** field contents with the *organisation* data

**CHARSET="Charset ID"** - set the current message character set to the character set that corresponds to the *Charset ID*. All possible character sets with their IDs are defined in the “Options | XLAT Tables” dialogue.

**ACCOUNT="Account Name"** - set the active account to the account with the name equal to *Account Name*.

**READCONFIRM** - set the Reading Confirmation Request flag for the current message.

**NOREADCONFIRM** - disable the Reading Confirmation Request flag for the current message.

**RCPTCONFIRM** - set the Receipt Request flag for the current message.

**NORCPTCONFIRM** - disable the Receipt Request flag for the current message.

**SIGNCOMPLETE** - automatically ask to sign the current message when it is completed

**NOSIGNCOMPLETE** - message editor only – disables automatic signing of a message if it was selected by the %SIGNCOMPLETE macro, using the default settings or manually with the menu option

**ENCRYPTCOMPLETE** - automatically encrypt the current message when it is completed

**NOENCRYPTCOMPLETE** - message editor only – disables automatic encryption of a message if it was selected by the %SIGNCOMPLETE macro, using the default settings or manually with the menu option

**USEPGP** - enable using of OpenPGP for encryption/signing of the current message

**NOUSEPGP** - disable using of OpenPGP for encryption/signing of the current message

**USES Mime** - enable using of S/MIME for encryption/signing of the current message

**NOUSES Mime** - disable using of S/MIME for encryption/signing of the current message



**SINGLERE** - disable reply counting.

**WRAPJUSTIFY** - used for the %WRAPPED="..." macro to keep the wrapped text justified by the left and the right edges.

**NOWRAPJUSTIFY** - used for the %WRAPPED="..." macro to keep the wrapped text justified only by the left edge (no justification performed).

**CLEAR** - when used in a Quick Template, clears all the previously entered/generated text.

**ISSIGNATURE** - when used in a Quick Template, tells the message editor that the Quick Template must be used as a signature. In other words, the text generated by the Quick Template replaces everything after the end-of-message line ("-- ", dash-dash-space); if the end-of-message line is not detected, it is added to the end of the message followed by the generated text.

**SINGLELINE,**  
**ONELINE** - when used in a Quick Template, makes the output text of the template composed in a single line. This can be useful for templates where long regular expressions used for extracting some text for message fields

**BLANK** - this macro is used to create a blank template.

**WINDOWSMAJORVERSION**- insert Windows' major version number

**WINDOWSMINORVERSION** - insert Windows' minor version number

**WINDOWSBUILDNUMBER**- insert Windows' version build number

**WINDOWSCSDVERSION**- insert Windows' additional version information

**WINDOWSPLATFORMNAME** - insert Windows' platform name (e.g. Windows 95, Windows NT)

**THEBATVERSION** - insert The Bat! version number.

**FOLDERFROMNAME** - insert **FROM** name from the current folder's Identity properties

**FOLDERFROMADDR** - insert **FROM** address from the current folder's Identity properties

**FOLDERREPLYNAME** - insert **REPLY-TO** name from the current folder's Identity properties

**FOLDERREPLYADDR** - insert **REPLY-TO** address from the current folder's Identity properties

**ACCOUNTNAME** - insert the name of the active account

**FOLDERNAME** - insert the name of the current folder

**FOLDERORGANISATION,**  
**FOLDERORGANIZATION**- insert the Organisation name from the current folder's Identity properties

**OTEXTSIZE** - insert the byte size of the original message

**PAGENUMBER**- insert the number of the currently printed page

**TOTALPAGES** insert the number of the currently printed page

**ATTACHMENTS** - insert the list of the files attached to the current message

**OATTACHMENTS** - insert the list of the files attached to the original message

**REGEXPTEXT**="*regex*" - insert the result of the search of the Regular Expression *regex* with the text of the original message as the subject string.

**REGEXPQUOTES**="*regex*" - insert the result of the search of the Regular Expression *regex* as quotation.

**SETPATTREGEXP**="*regex*" - set the Regular Expression pattern which will be evaluated with the following **%REGEXPMATCH** macro *text* parameter as the subject string.

**REGEXPMATCH**="*giventext*" - insert the result of the search in *giventext* of the Regular Expression previously set by **SETPATTREGEXP**="*regex*".

**REGEXPBLINDMATCH**="*giventext*" - same as **REGEXPMATCH** except matching text is not inserted but may be extracted by **SUBPATT**="*n*".

**SUBPATT**="*n*" - *n* is a number of captured substring to be inserted. Substrings are set by **REGEXPMATCH** or **REGEXPBLINDMATCH** macros. 0 means text matching entire regex pattern, 1 - matching first subpattern, 2 - matching second subpattern, etc...

**INSERTPGPKEY**="*address*" - insert the OpenPGP key from the public key ring. This feature works with the internal OpenPGP implementation only.

**INCLUDE**="*filepath*" - insert the text generated by the template located in the file determined by the *filepath*.

**PUT**="*filepath*" - insert the text from the file determined by the *filepath*.

**ATTACHVCARD** - attach the personal vCard of the active account.

**ATTACHFILE**="*filepath*" - attach a file determined by the *filepath*.

**QINCLUDE**="*Quick Template Handle*" - insert the text generated by the Quick Template with

**REM**="*comment*" - use this macro to comment your templates

**UPPER**="*text*" - convert the *text* to upper case

**LOWER**="*text*" - convert the *text* to lower case

**CAPITAL**="*text*" - convert the *text* to lower case and then convert the first letter of each word to upper case

**CAPITALFIRST**="*text*" / **UCFIRST**="*text*" - convert the first letter of the *text* to upper case

**LANGUAGE**="*Language ID*" - set the default spell checker language to the language defined by Language ID. Possible Language IDs can be:

AM - American English  
EA - Australian English  
BR - British English  
CT - Catalan  
CZ - Czech  
DA - Danish

FI - Finnish  
 FR - French  
 FC - French Canadian  
 GE - German  
 IT - Italian  
 NO - Norwegian (Bokmal)  
 NN - Norwegian (Nynorsk)  
 NL - Dutch  
 PB - Portuguese (Brazil)  
 PT - Portuguese (Iberian)  
 PL - Polish  
 SP - Spanish  
 SW - Swedish  
 RU - Russian  
 HU - Hungarian

To use Common Speller API dictionaries, the Language ID must be prefixed by "CSAPI". E.g.  
**%LANGUAGE="CSAPI RU"**

**QUOTESTYLE="expression"** - specify the quotation sign prefix used after this macro. *Expression* can be any of the following:

NONE - use an empty prefix (a standard)  
 I - use initials of the sender of the original message  
 IF - use the first initial of the sender of the original message  
 N - use the full name of the sender of the original message  
 F - use the first name of the sender of the original message  
 L - use the last name of the sender of the original message  
 =text - use the "text" as the prefix, e.g. **%QUOTESTYLE="--|"**

**ABnnnPPP** - insert a parameter from the address book. The "**nnn**" part determines the address used for retrieving information from the address book, the "**PPP**" part is the name of the address book entry parameter. Possible values for the "**nnn**" part are:

TO - use the address entry for the TO addressee of the current message  
 OFROM - use the address entry for the FROM address of the original message  
 OREPLY - use the address entry for the REPLY-TO address of the original message  
 OTO - use the address entry for the TO addressee of the original message  
 FROM - use the address entry for the FROM address of the current message  
 REPLY - use the address entry for the REPLY-TO address of the current message

Possible values for the "**PPP**" part are:

Name - the full name  
 FirstName - the first name  
 LastName - the last name  
 MiddleName - the middle name  
 Handle - the handle (alias)  
 NamePrefix - the name prefix (such as "Mr.", "Mrs.", "Dr." etc)  
 NameSuffix - the name suffix (such as "Jr.", "PhD", etc)  
 Birthday - the date of birth  
 Email - the primary e-mail address  
 Addr - the home street address  
 City - the home city name  
 State - the home state/province name  
 ZIP - the home address ZIP/Postal code

Country	- the home address country name
Phone	- the home phone number
Fax	- the home facsimile number
Mobile	- the mobile phone number
Page	- the personal homepage URL
Company	- the company name
Job	- the job title
Dept	- the department name
Office	- the office number
BusAddr	- the business street address
BusCity	- the business address city name
BusState	- the business address state/province name
BusZIP	- the business address ZIP/Postal code
BusCountry	- the business address Country name
BusPhone	- the business telephone number
BusFax	- the business facsimile number
BusPager	- the business pager number
BusPage	- the business homepage URL
Gender	- the person's gender
Charset	- the default character set
Memo	- the Memo contents of the address entry

### **A special note about using macro parameters:**

Macro parameter can be included either in double quotes or apostrophes. To use double quote or apostrophe within a macro parameter when the embracing character is the same, use the pair of needed character instead of a single one. E.g.: in `%MACRO='my "double quoted" text'` the macro parameter is *my "double quoted" text* ; it is also possible to use this construct instead: `%MACRO="my ""double quoted"" text"` – note the doubled double quotes inside the macro parameter.

In most of macro parameters, it is possible to use macros for generating some text. E.g.:  
`%TO=%QINCLUDE="MySpecialToQuickTemplate"`

What are Regular expressions?

Regular expression matching allows you to test whether a text fits into a specific syntactic shape. You can also search a text for a sub-string that fits a pattern.

A regular expression describes a set of strings. The simplest case is one that describes a particular string; for example, the string `foo` when regarded as a regular expression matches `foo` and nothing else. Nontrivial regular expressions use certain special constructs so that they can match more than one string. For example, the regular expression `foo|bar` matches either the string `foo` or the string `bar`; the regular expression `c[ad]*r` matches any of the strings `cr`, `car`, `cdr`, `caar`, `caddar` and all other such strings with any number of `a`'s and `d`'s.

The Bat! allows using of regular expressions in:

[Message Editor](#)

[Message Finder](#)

[Sorting Office / Filters](#)

[Templates](#)

To learn about Regular Expressions, read the following topics:

[Regular Expressions Syntax \(Basic\)](#)

[Regular Expressions Syntax \(Advanced\)](#)

## Using Regular Expressions in the Message Editor/Viewer

In the message editor/viewer, regular expressions are used for search of sub-strings matching a search pattern. To find a sub-string that matches a regular expression, invoke the search dialogue and type the regular expression, make sure that the checkbox “**Regular Expressions**” is checked and start search. Regular expressions can also be used for the Replace function in the Message Editor in the same way as they are used in the search dialogue.

If a match found, a text that matches given regular expression will be selected in a message editor/viewer. If the regular expression has subpatterns, the first captured substring will be selected, i.e for the regular expression `Total amount: *(\S+)` only the word following `Total amount` will be selected but not `Total amount` itself.

Default PCRE options for message editor/viewer are PCRE\_CASELESS by default or *none* for "Case sensitive" search. String-by-string matching is done.

See also:

[Regular Expressions Syntax \(Basic\).](#)

[Regular Expressions Syntax \(Advanced\).](#)

[Message Editor.](#)

## Using Regular Expressions in the Message Finder

To make the Message Finder use regular expressions as the search pattern, enter the regular expression you want to use, click to the “**Advanced**” tab and tick the “**Regular expressions**” checkbox.

Default PCRE options for message finder are PCRE\_CASELESS+PCRE\_MULTILINE by default or PCRE\_MULTILINE for "Case sensitive search". Whole message as one string matching is done. CF/LF characters (0D/0A) are replaced to LF (0A) before matching.

See also:

[Regular Expressions Syntax \(Basic\)](#)

[Regular Expressions Syntax \(Advanced\)](#)

## Using Regular Expressions in Message Filters

It is possible to use Regular Expressions as signal strings in the [message filters](#). In a filter definition in the Sorting Office dialogue, select the **“Options”** tab and tick the **“Regular expressions”** checkbox. Note that ALL signal strings of the filter will be treated as regular expressions so make sure that they all are compliant with the syntax of Regular Expressions.

Default PCRE [options](#) for message filters are PCRE\_CASELESS+PCRE\_MULTILINE. Whole message as one string matching is done. CF/LF characters (0D/0A) are replaced to LF (0A) before matching.

See also:

[Regular Expressions Syntax \(Basic\)](#).

[Regular Expressions Syntax \(Advanced\)](#).

[Message Filters](#).



## Using Regular Expressions in Templates

The Bat! allows to use the power of the regular expressions in the [message templates](#). The general idea is to search a text for the first entry of a sub-string that matches the pattern defined by a regular expression and to insert the search result into the message text. For example, it is possible to parse incoming messages of a particular format and create new messages in other format with added or removed information.

To search the **original message text** for a sub-string, use **%REGEXPTTEXT="regex"** macro where *regex* defines the pattern. **%REGEXPQUOTES="regex"** macro do the same, but returns the result as quotation.

When it is needed to search any generic text for a particular sub-string, use **%SETPATTREGEXP="regex"** and **%REGEXPMATCH="string"** macros combination. The *regex* defines the search pattern for all subsequent occurrences of the **%REGEXPMATCH** macro until the next occurrence of the **%SETPATTREGEXP**. The *string* parameter is a template, so it is possible to use all possible macros to generate any text you need, e.g.

**%QUOTES="%SETPATTREGEXP="regex"%REGEXPMATCH="%TEXT"** is equivalent to **%REGEXPQUOTES="regex"**.

If a match found, a text that matches given regular expression will be returned. If the regular expression has [subpatterns](#), the first captured substring will be returned, i.e for the regular expression Total amount: \*(\S+) only the word following Total amount will be returned but not Total amount itself.

**%REGEXPBLINDMATCH="string"** is like **%REGEXPMATCH="string"** macro except matched string/substrings are not returned, they may be further extracted by **%SUBPATT="n"** macro where *n* is a number of captured substring; 0 means text matching entire regexp pattern, 1 - matching first subpattern, 2 - second subpattern, etc.... Thus this **%REGEXPBLINDMATCH="string"** macro allows more then one subpatterns e.g.

```
%QUOTES="%SETPATTREGEXP="total_amount + (\S+) .*flowers_type +
(\S+) "%REGEXPBLINDMATCH="%TEXT"pay %SUBPATT="1" for %SUBPATT="2"
```

Default PCRE [options](#) for templates are PCRE\_CASELESS+PCRE\_MULTILINE+ PCRE\_DOTALL. Whole message as one string matching is done. CF/LF characters (0D/0A) are replaced to LF (0A) before matching.

See also:

[Regular Expressions Syntax \(Basic\).](#)

[Regular Expressions Syntax \(Advanced\).](#)

[Templates.](#)

## Regular Expressions Syntax (Basic)

A *regular expression* is a set of rules that describes a generalised string. If the characters that make up a particular string conform to the rules of a particular regular expression, the regular expression is said to *match* that string.

A few concrete examples usually help after an overblown definition like that one. The regular expression `b.` matches the strings `bovine`, `above`, `Bobby`, and `Bob Jones`, but not the strings `Bell`, `b`, or `Bob`. That's because the expression insists that the letter *b* (lowercase) must be in the string and must be followed immediately by another character.

The regular expression `b+`, on the other hand, requires the lowercase letter *b* at least once. This expression matches `b` and `Bob` in addition to the example matches for `b.` in the preceding paragraph. The regular expression `b*` requires zero or more *bs*, so it matches any string. That seems to be fairly useless, but it makes more sense as part of a larger regular expression. `Bob*y`, for example, matches all of `Boy`, `Boby`, and `Bobby` but not `Boboby`.

**Assertions** Several so-called assertions are used to anchor parts of the pattern to word or string boundaries. The `^` assertion matches the start of a string, so the regular expression `^fool` matches `fool` and `foolhardy` but not `tomfoolery` or `April fool`. The following table lists the assertions.

### Regular-Expression Assertions

Assertion	Matches	Example	Matches	Doesn't Match
<code>^</code>	Start of string	<code>^fool</code>	<code>foolish</code>	<code>tomfoolery</code>
<code>\$</code>	End of string	<code>fool\$</code>	<code>April fool</code>	<code>foolish</code>
<code>\b</code>	Word boundary	<code>be\bside</code>	<code>be side</code>	<code>beside</code>
<code>\B</code>	Nonword boundary	<code>be\Bside</code>	<code>beside</code>	<code>be side</code>

**Atoms** The `.` (period) that you saw in `b.` earlier in this chapter is an example of a regular-expression atom. *Atoms* are, as the name suggests, the fundamental building blocks of a regular expression. A full list of atoms appears in the following table.

### Regular-Expression Atoms

Atom	Matches	Example	Matches	Doesn't Match
Period ( <code>.</code> )	Any character except new line	<code>b.b</code>	<code>bob</code>	<code>bb</code>
List of characters in brackets	Any one of those characters	<code>^[Bb]</code>	<code>Bob, bob</code>	<code>Rbob</code>
Regular expression in parentheses	Anything that regular expression matches	<code>^a(b.b)c\$</code>	<code>abobc</code>	<code>abbc</code>

**Quantifiers** A *quantifier* is a modifier for an atom. It can be used to specify that a particular atom must appear at least once, as in `b+`. The atom quantifiers are listed in the following table.

### Regular-Expression Atom Quantifiers

Quantifier	Matches	Example	Matches	Doesn't Match
<code>*</code>	Zero or more instances of the atom	<code>ab*c</code>	<code>ac, abc</code>	<code>abb</code>
<code>+</code>	One or more instances of the atom	<code>ab+c</code>	<code>abc</code>	<code>ac</code>
<code>?</code>	Zero or one instances of the atom	<code>ab?c</code>	<code>ac, abc</code>	<code>abbc</code>
<code>{n}</code>	<i>n</i> instances of the atom	<code>ab{2}c</code>	<code>abbc</code>	<code>abbbc</code>
<code>{n,}</code>	At least <i>n</i> instances of the atom	<code>ab{2,}c</code>	<code>abbc, abbbc</code>	<code>abc</code>
<code>{n,m}</code>	At least <i>n</i> , most <i>m</i> instances of the atom	<code>ab{2,3}c</code>	<code>abbc</code>	<code>abbbcat</code>

**Special Characters** Several special characters are denoted by backslashed letters, with `\n` being especially familiar

to C programmers, perhaps. The following table lists the special characters.

### Regular-Expression Special Characters

Symbol	Matches	Example	Matches	Doesn't Match
\d	Any digit	b\dd	b4d	bad
\D	Nondigit	b\Dd	bdd	b4d
\n	New line			
\r	Carriage return			
\t	Tab			
\f	Form feed			
\s	White-space character			
\S	Non-white-space character			
\w	Alphanumeric character	a\wb	a2b	a^b
\W	Nonalphanumeric character	a\Wb	aa^b	aabb

**Backslashed Tokens** It is essential that regular expressions be capable of using all characters, so that all possible strings that occur in the real world can be matched. With so many characters having special meanings, a mechanism is required that allows you to represent any arbitrary character in a regular expression. This mechanism is a backslash (\), followed by a numeric quantity. This quantity can take any of the following formats:

<b>Single or double digit</b>	matched quantities after a match. These matched quantities are called backreferences and are explained in a <a href="#">separate section</a> .
<b>Two-or three-digit octal number</b>	the character with that number as character code, unless it's possible to interpret it as a backreference.
<b>x, followed by two hexadecimal digits</b>	the character with that number as its character code. \x3e, for example, is >
<b>c, followed by a single character</b>	the control character. \cG, for example, matches Ctrl+G.
<b>Any other character</b>	the character itself. \&, for example, matches the & character

See also:

[Regular Expressions Syntax \(Advanced\)](#)

## Regular Expressions Syntax (Advanced)

The Bat! uses Perl-compatible regular expressions(PCRE) library. The library is a set of functions that implement regular expression pattern matching using the same syntax and semantics as Perl 5, with just a few differences. The current implementation corresponds to Perl 5.005. The syntax and semantics of the regular expressions supported by the PCRE are described in this chapter. For Perl 5 regular expression syntax, read the Perl regular expressions man page at <http://www.perl.com/CPAN-local/doc/manual/html/pod/perlre.html>

- Meta-Characters.
- Backslash.
- Circumflex and Dollar.
- Full Stop (Period, Dot).
- Square Brackets.
- Vertical Bar.
- Internal Option Setting.
- Subpatterns.
- Repetition.
- Back References.
- Assertions.
- Once-Only Subpatterns.
- Conditional Subpatterns.
- Comments.
- Performance.
- Limitations.
- Differences From Perl.

The PCRE Library, which is open source software, and these chapters (Regular Expressions Syntax (Advanced)) are written by Philip Hazel <ph10@cam.ac.uk>, University of Cambridge Computing Service, Cambridge, England. Phone: +44 1223 334714. Copyright © 1997-1999 University of Cambridge

Regular expressions are also described in the Perl documentation and in a number of other books, some of which have copious examples. Jeffrey Friedl's "Mastering Regular Expressions", published by O'Reilly (ISBN 1-56592-257-3), covers them in great detail. The description here is intended as reference documentation.

## Meta-Characters

A regular expression is a pattern that is matched against a subject string from left to right. Most characters stand for themselves in a pattern, and match the corresponding characters in the subject. As a trivial example, the pattern

The quick brown fox

matches a portion of a subject string that is identical to itself. The power of regular expressions comes from the ability to include alternatives and repetitions in the pattern. These are encoded in the pattern by the use of **meta-characters**, which do not stand for themselves but instead are interpreted in some special way.

There are two different sets of meta-characters: those that are recognised anywhere in the pattern except within square brackets, and those that are recognised in square brackets. Outside square brackets, the meta-characters are as follows:

\	general escape character with several uses
^	assert start of subject (or line, in multiline mode)
\$	assert end of subject (or line, in multiline mode)
.	match any character except newline (by default)
[	start character class definition
	start of alternative branch
(	start subpattern
)	end subpattern
?	extends the meaning of (
	also 0 or 1 quantifier
	also quantifier minimizer
*	0 or more quantifier
+	1 or more quantifier
{	start min/max quantifier

Part of a pattern that is in square brackets is called a "character class". In a character class the only meta-characters are:

\	general escape character
^	negate the class, but only if the first character
-	indicates character range
]	terminates the character class

The following sections describe the use of each of the meta-characters.

Note: This topic was taken from the PCRE library manual. The PCRE library is open source software, written by Philip Hazel <ph10@cam.ac.uk>, and copyright by the University of Cambridge, England.

## Backslash

The backslash character has several uses. Firstly, if it is followed by a non-alphameric character, it takes away any special meaning that character may have. This use of backslash as an escape character applies both inside and outside character classes.

For example, if you want to match a "\*" character, you write "\\*" in the pattern. This applies whether or not the following character would otherwise be interpreted as a meta-character, so it is always safe to precede a non-alphameric with "\" to specify that it stands for itself. In particular, if you want to match a backslash, you write "\\".

If a pattern is compiled with the PCRE\_EXTENDED option, whitespace in the pattern (other than in a character class) and characters between a "#" outside a character class and the next newline character are ignored. An escaping backslash can be used to include a whitespace or "#" character as part of the pattern.

A second use of backslash provides a way of encoding non-printing characters in patterns in a visible manner. There is no restriction on the appearance of non-printing characters, apart from the binary zero that terminates a pattern, but when a pattern is being prepared by text editing, it is usually easier to use one of the following escape sequences than the binary character it represents:

\a	alarm, that is, the BEL character (hex 07)
\cx	"control-x", where x is any character
\e	escape (hex 1B)
\f	formfeed (hex 0C)
\n	newline (hex 0A)
\r	carriage return (hex 0D)
\t	tab (hex 09)
\xhh	character with hex code hh
\ddd	character with octal code ddd, or backreference

The precise effect of "\cx" is as follows: if "x" is a lower case letter, it is converted to upper case. Then bit 6 of the character (hex 40) is inverted. Thus "\cz" becomes hex 1A, but "\c{" becomes hex 3B, while "\c;" becomes hex 7B.

After "\x", up to two hexadecimal digits are read (letters can be in upper or lower case).

After "\0" up to two further octal digits are read. In both cases, if there are fewer than two digits, just those that are present are used. Thus the sequence "\0\x07" specifies two binary zeros followed by a BEL character. Make sure you supply two digits after the initial zero if the character that follows is itself an octal digit.

The handling of a backslash followed by a digit other than 0 is complicated. Outside a character class, PCRE reads it and any following digits as a decimal number. If the number is less than 10, or if there have been at least that many previous capturing left parentheses in the expression, the entire sequence is taken as a **back reference**. A description of how this works is given later, following the discussion of parenthesized subpatterns.

Inside a character class, or if the decimal number is greater than 9 and there have not been that many capturing subpatterns, PCRE re-reads up to three octal digits following the backslash, and generates a single byte from the least significant 8 bits of the value. Any subsequent digits stand for themselves. For example:

\040	is another way of writing a space
\40	is the same, provided there are fewer than 40 previous capturing subpatterns
\7	is always a back reference
\11	might be a back reference, or another way of writing a tab
\011	is always a tab
\0113	is a tab followed by the character "3"
\113	is the character with octal code 113 (since there can be no more than 99 back references)

\377 is a byte consisting entirely of 1 bits  
\81 is either a back reference, or a binary zero followed by the two characters "8" and "1"

Note that octal values of 100 or greater must not be introduced by a leading zero, because no more than three octal digits are ever read.

All the sequences that define a single byte value can be used both inside and outside character classes. In addition, inside a character class, the sequence "\b" is interpreted as the backspace character (hex 08). Outside a character class it has a different meaning (see below).

The third use of backslash is for specifying generic character types:

\d any decimal digit  
\D any character that is not a decimal digit  
\s any whitespace character  
\S any character that is not a whitespace character  
\w any "word" character  
\W any "non-word" character

Each pair of escape sequences partitions the complete set of characters into two disjoint sets. Any given character matches one, and only one, of each pair.

A "word" character is any letter or digit or the underscore character, that is, any character which can be part of a Perl "word". The definition of letters and digits is controlled by PCRE's character tables, and may vary if locale- specific matching is taking place (see "Locale support" above). For example, in the "fr" (French) locale, some character codes greater than 128 are used for accented letters, and these are matched by \w.

These character type sequences can appear both inside and outside character classes. They each match one character of the appropriate type. If the current matching point is at the end of the subject string, all of them fail, since there is no character to match.

The fourth use of backslash is for certain simple assertions. An assertion specifies a condition that has to be met at a particular point in a match, without consuming any characters from the subject string. The use of subpatterns for more complicated assertions is described below. The backslashed assertions are

\b word boundary  
\B not a word boundary  
\A start of subject (independent of multiline mode)  
\Z end of subject or newline at end (independent of multiline mode)  
\z end of subject (independent of multiline mode)

These assertions may not appear in character classes (but note that "\b" has a different meaning, namely the backspace character, inside a character class).

A word boundary is a position in the subject string where the current character and the previous character do not both match \w or \W (i.e. one matches \w and the other matches \W), or the start or end of the string if the first or last character matches \w, respectively.

The \A, \Z, and \z assertions differ from the traditional circumflex and dollar (described below) in that they only ever match at the very start and end of the subject string, whatever options are set. They are not affected by the PCRE\_NOTBOL or PCRE\_NOTEOL options. The difference between \Z and \z is that \Z matches before a newline that is the last character of the string as well as at the end of the string, whereas \z matches only at the end.

Note: This topic was taken from the PCRE library manual. The PCRE library is open source software, written by Philip Hazel <ph10@cam.ac.uk>, and copyright by the University of Cambridge, England.

## Circumflex and Dollar

Outside a character class, in the default matching mode, the circumflex character is an assertion which is true only if the current matching point is at the start of the subject string. Inside a character class, circumflex has an entirely different meaning (see below).

Circumflex need not be the first character of the pattern if a number of alternatives are involved, but it should be the first thing in each alternative in which it appears if the pattern is ever to match that branch. If all possible alternatives start with a circumflex, that is, if the pattern is constrained to match only at the start of the subject, it is said to be an "anchored" pattern. (There are also other constructs that can cause a pattern to be anchored.)

A dollar character is an assertion which is true only if the current matching point is at the end of the subject string, or immediately before a newline character that is the last character in the string (by default). Dollar need not be the last character of the pattern if a number of alternatives are involved, but it should be the last item in any branch in which it appears. Dollar has no special meaning in a character class.

The meaning of dollar can be changed so that it matches only at the very end of the string, by setting the `PCRE_DOLLAR_ENDONLY` option at compile or matching time. This does not affect the `\Z` assertion.

The meanings of the circumflex and dollar characters are changed if the `PCRE_MULTILINE` option is set. When this is the case, they match immediately after and immediately before an internal `"\n"` character, respectively, in addition to matching at the start and end of the subject string. For example, the pattern `/^abc$/` matches the subject string `"def\nabc"` in multiline mode, but not otherwise. Consequently, patterns that are anchored in single line mode because all branches start with `"^"` are not anchored in multiline mode. The `PCRE_DOLLAR_ENDONLY` option is ignored if `PCRE_MULTILINE` is set.

Note that the sequences `\A`, `\Z`, and `\z` can be used to match the start and end of the subject in both modes, and if all branches of a pattern start with `\A` it is always anchored, whether `PCRE_MULTILINE` is set or not.

Note: This topic was taken from the PCRE library manual. The PCRE library is open source software, written by Philip Hazel <ph10@cam.ac.uk>, and copyright by the University of Cambridge, England.



## Full Stop (Period, Dot)

Outside a character class, a dot in the pattern matches any one character in the subject, including a non-printing character, but not (by default) newline. If the `PCRE_DOTALL` option is set, then dots match newlines as well. The handling of dot is entirely independent of the handling of circumflex and dollar, the only relationship being that they both involve newline characters. Dot has no special meaning in a character class.

Note: This topic was taken from the PCRE library manual. The PCRE library is open source software, written by Philip Hazel <ph10@cam.ac.uk>, and copyright by the University of Cambridge, England.

## Square Brackets

An opening square bracket introduces a character class, terminated by a closing square bracket. A closing square bracket on its own is not special. If a closing square bracket is required as a member of the class, it should be the first data character in the class (after an initial circumflex, if present) or escaped with a backslash.

A character class matches a single character in the subject; the character must be in the set of characters defined by the class, unless the first character in the class is a circumflex, in which case the subject character must not be in the set defined by the class. If a circumflex is actually required as a member of the class, ensure it is not the first character, or escape it with a backslash.

For example, the character class `[aeiou]` matches any lower case vowel, while `^[aeiou]` matches any character that is not a lower case vowel. Note that a circumflex is just a convenient notation for specifying the characters which are in the class by enumerating those that are not. It is not an assertion: it still consumes a character from the subject string, and fails if the current pointer is at the end of the string.

When caseless matching is set, any letters in a class represent both their upper case and lower case versions, so for example, a caseless `[aeiou]` matches "A" as well as "a", and a caseless `^[aeiou]` does not match "A", whereas a careful version would.

The newline character is never treated in any special way in character classes, whatever the setting of the `PCRE_DOTALL` or `PCRE_MULTILINE` options is. A class such as `^[a]` will always match a newline.

The minus (hyphen) character can be used to specify a range of characters in a character class. For example, `[d-m]` matches any letter between d and m, inclusive. If a minus character is required in a class, it must be escaped with a backslash or appear in a position where it cannot be interpreted as indicating a range, typically as the first or last character in the class.

It is not possible to have the literal character "]" as the end character of a range. A pattern such as `[W-]46]` is interpreted as a class of two characters ("W" and "-") followed by a literal string "46]", so it would match "W46]" or "-46]". However, if the "]" is escaped with a backslash it is interpreted as the end of range, so `[W-]46]` is interpreted as a single class containing a range followed by two separate characters. The octal or hexadecimal representation of "]" can also be used to end a range.

Ranges operate in ASCII collating sequence. They can also be used for characters specified numerically, for example `[\000-\037]`. If a range that includes letters is used when caseless matching is set, it matches the letters in either case. For example, `[W-c]` is equivalent to `[^\^_`wxyzabc]`, matched caselessly, and if character tables for the "fr" locale are in use, `[\xc8-\xcb]` matches accented E characters in both cases.

The character types `\d`, `\D`, `\s`, `\S`, `\w`, and `\W` may also appear in a character class, and add the characters that they match to the class. For example, `[\dABCDEF]` matches any hexadecimal digit. A circumflex can conveniently be used with the upper case character types to specify a more restricted set of characters than the matching lower case type. For example, the class `^[^W_]` matches any letter or digit, but not underscore.

All non-alphameric characters other than `\`, `-`, `^` (at the start) and the terminating `]` are non-special in character classes, but it does no harm if they are escaped.

Note: This topic was taken from the PCRE library manual. The PCRE library is open source software, written by Philip Hazel <ph10@cam.ac.uk>, and copyright by the University of Cambridge, England.

## Vertical Bar

Vertical bar characters are used to separate alternative patterns. For example, the pattern

```
gilbert|sullivan
```

matches either "gilbert" or "sullivan". Any number of alternatives may appear, and an empty alternative is permitted (matching the empty string). The matching process tries each alternative in turn, from left to right, and the first one that succeeds is used. If the alternatives are within a subpattern (defined below), "succeeds" means matching the rest of the main pattern as well as the alternative in the subpattern.

Note: This topic was taken from the PCRE library manual. The PCRE library is open source software, written by Philip Hazel <ph10@cam.ac.uk>, and copyright by the University of Cambridge, England.

## Internal Option Setting

The settings of `PCRE_CASELESS`, `PCRE_MULTILINE`, `PCRE_DOTALL`, and `PCRE_EXTENDED` can be changed from within the pattern by a sequence of Perl option letters enclosed between "(?" and ")". The option letters are

i	<code>PCRE_CASELESS</code>	Letters in the pattern match both upper and lower case letters. It is equivalent to Perl's <code>/i</code> option.
m	<code>PCRE_MULTILINE</code>	<p>By default, PCRE treats the subject string as consisting of a single "line" of characters (even if it actually contains several newlines). The "start of line" metacharacter (^) matches only at the start of the string, while the "end of line" metacharacter (\$) matches only at the end of the string, or before a terminating newline (unless <code>PCRE_DOLLAR_ENDONLY</code> is set). This is the same as Perl.</p> <p>When <code>PCRE_MULTILINE</code> it is set, the "start of line" and "end of line" constructs match immediately following or immediately before any newline in the subject string, respectively, as well as at the very start and end. This is equivalent to Perl's <code>/m</code> option. If there are no "\n" characters in a subject string, or no occurrences of ^ or \$ in a pattern, setting <code>PCRE_MULTILINE</code> has no effect.</p>
s	<code>PCRE_DOTALL</code>	If this bit is set, a dot metacharacter in the pattern matches all characters, including newlines. Without it, newlines are excluded. This option is equivalent to Perl's <code>/s</code> option. A negative class such as <code>[^a]</code> always matches a newline character, independent of the setting of this option.
x	<code>PCRE_EXTENDED</code>	If this bit is set, whitespace data characters in the pattern are totally ignored except when escaped or inside a character class, and characters between an unescaped # outside a character class and the next newline character, inclusive, are also ignored. This is equivalent to Perl's <code>/x</code> option, and makes it possible to include comments inside complicated patterns. Note, however, that this applies only to data characters. Whitespace characters may never appear within special character sequences in a pattern, for example within the sequence <code>(?(</code> which introduces a conditional subpattern.

For example, `(?im)` sets caseless, multiline matching. It is also possible to unset these options by preceding the letter with a hyphen, and a combined setting and unsetting such as `(?im-sx)`, which sets `PCRE_CASELESS` and `PCRE_MULTILINE` while unsetting `PCRE_DOTALL` and `PCRE_EXTENDED`, is also permitted. If a letter appears both before and after the hyphen, the option is unset.

The scope of these option changes depends on where in the pattern the setting occurs. For settings that are outside any subpattern (defined below), the effect is the same as if the options were set or unset at the start of matching. The following patterns all behave in exactly the same way:

```
(?i)abc
a(?i)bc
ab(?i)c
abc(?i)
```

which in turn is the same as compiling the pattern `abc` with `PCRE_CASELESS` set. In other words, such "top level" settings apply to the whole pattern (unless there are other changes inside subpatterns). If there is more than one setting of the same option at top level, the rightmost setting is used.

If an option change occurs inside a subpattern, the effect is different. This is a change of behaviour in Perl 5.005. An option change inside a subpattern affects only that part of the subpattern that follows it, so

`(a(?i)b)c`

matches `abc` and `aBc` and no other strings (assuming `PCRE_CASELESS` is not used). By this means, options can be made to have different settings in different parts of the pattern. Any changes made in one alternative do carry on into subsequent branches within the same subpattern. For example,

`(a(?i)b|c)`

matches `"ab"`, `"aB"`, `"c"`, and `"C"`, even though when matching `"C"` the first branch is abandoned before the option setting. This is because the effects of option settings happen at compile time. There would be some very weird behaviour otherwise.

The PCRE-specific options `PCRE_UNGREEDY` and `PCRE_EXTRA` can be changed in the same way as the Perl-compatible options by using the characters `U` and `X` respectively. The `(?X)` flag setting is special in that it must always occur earlier in the pattern than any of the additional features it turns on, even when it is at top level. It is best put at the start.

Note: This topic was taken from the PCRE library manual. The PCRE library is open source software, written by Philip Hazel <ph10@cam.ac.uk>, and copyright by the University of Cambridge, England.

## Subpatterns

Subpatterns are delimited by parentheses (round brackets), which can be nested. Marking part of a pattern as a subpattern does two things:

1. It localizes a set of alternatives. For example, the pattern

```
cat(aract|erpillar|)
```

matches one of the words "cat", "cataract", or "caterpillar". Without the parentheses, it would match "cataract", "erpillar" or the empty string.

2. It sets up the subpattern as a capturing subpattern (as defined above). Opening parentheses are counted from left to right (starting from 1) to obtain the numbers of the capturing subpatterns.

For example, if the string "the red king" is matched against the pattern

```
the ((red|white) (king|queen))
```

the captured substrings are "red king", "red", and "king", and are numbered 1, 2, and 3.

The fact that plain parentheses fulfil two functions is not always helpful. There are often times when a grouping subpattern is required without a capturing requirement. If an opening parenthesis is followed by "?:", the subpattern does not do any capturing, and is not counted when computing the number of any subsequent capturing subpatterns. For example, if the string "the white queen" is matched against the pattern

```
the ((?:red|white) (king|queen))
```

the captured substrings are "white queen" and "queen", and are numbered 1 and 2. The maximum number of captured substrings is 99, and the maximum number of all subpatterns, both capturing and non-capturing, is 200.

As a convenient shorthand, if any option settings are required at the start of a non-capturing subpattern, the option letters may appear between the "?:" and the ":". Thus the two patterns

```
(?i:saturday|sunday)  
(?:(?i)saturday|sunday)
```

match exactly the same set of strings. Because alternative branches are tried from left to right, and options are not reset until the end of the subpattern is reached, an option setting in one branch does affect subsequent branches, so the above patterns match "SUNDAY" as well as "Saturday".

Note: This topic was taken from the PCRE library manual. The PCRE library is open source software, written by Philip Hazel <ph10@cam.ac.uk>, and copyright by the University of Cambridge, England.

## Repetition

Repetition is specified by quantifiers, which can follow any of the following items:

- a single character, possibly escaped
- the `.` metacharacter
- a character class
- a back reference (see next section)
- a parenthesized subpattern (unless it is an assertion - see below)

The general repetition quantifier specifies a minimum and maximum number of permitted matches, by giving the two numbers in curly brackets (braces), separated by a comma. The numbers must be less than 65536, and the first must be less than or equal to the second. For example:

`z{2,4}`

matches "zz", "zzz", or "zzzz". A closing brace on its own is not a special character. If the second number is omitted, but the comma is present, there is no upper limit; if the second number and the comma are both omitted, the quantifier specifies an exact number of required matches. Thus

`[aeiou]{3,}`

matches at least 3 successive vowels, but may match many more, while

`\d{8}`

matches exactly 8 digits. An opening curly bracket that appears in a position where a quantifier is not allowed, or one that does not match the syntax of a quantifier, is taken as a literal character. For example, `{,6}` is not a quantifier, but a literal string of four characters.

The quantifier `{0}` is permitted, causing the expression to behave as if the previous item and the quantifier were not present.

For convenience (and historical compatibility) the three most common quantifiers have single-character abbreviations:

- `*` is equivalent to `{0,}`
- `+` is equivalent to `{1,}`
- `?` is equivalent to `{0,1}`

It is possible to construct infinite loops by following a subpattern that can match no characters with a quantifier that has no upper limit, for example:

`(a?)*`

Earlier versions of Perl and PCRE used to give an error at compile time for such patterns. However, because there are cases where this can be useful, such patterns are now accepted, but if any repetition of the subpattern does in fact match no characters, the loop is forcibly broken.

By default, the quantifiers are "greedy", that is, they match as much as possible (up to the maximum number of permitted times), without causing the rest of the pattern to fail. The classic example of where this gives problems is in trying to match comments in C programs. These appear between the sequences `/*` and `*/` and within the sequence, individual `*` and `/` characters may appear. An attempt to match C comments by applying the pattern

```
/\*.\*\*/
```

to the string

```
/* first command */ not comment /* second comment */
```

fails, because it matches the entire string due to the greediness of the `.*` item.

However, if a quantifier is followed by a question mark, then it ceases to be greedy, and instead matches the minimum number of times possible, so the pattern

```
/\*.*?\*/
```

does the right thing with the C comments. The meaning of the various quantifiers is not otherwise changed, just the preferred number of matches. Do not confuse this use of question mark with its use as a quantifier in its own right. Because it has two uses, it can sometimes appear doubled, as in

```
\d??\d
```

which matches one digit by preference, but can match two if that is the only way the rest of the pattern matches.

If the `PCRE_UNGREEDY` option is set (an option which is not available in Perl) then the quantifiers are not greedy by default, but individual ones can be made greedy by following them with a question mark. In other words, it inverts the default behaviour.

When a parenthesized subpattern is quantified with a minimum repeat count that is greater than 1 or with a limited maximum, more store is required for the compiled pattern, in proportion to the size of the minimum or maximum.

If a pattern starts with `.*` or `{0,}` and the `PCRE_DOTALL` option (equivalent to Perl's `/s`) is set, thus allowing the `.` to match newlines, then the pattern is implicitly anchored, because whatever follows will be tried against every character position in the subject string, so there is no point in retrying the overall match at any position after the first. PCRE treats such a pattern as though it were preceded by `\A`. In cases where it is known that the subject string contains no newlines, it is worth setting `PCRE_DOTALL` when the pattern begins with `.*` in order to obtain this optimization, or alternatively using `^` to indicate anchoring explicitly.

When a capturing subpattern is repeated, the value captured is the substring that matched the final iteration. For example, after

```
(tweedle[dume]{3}\s*)+
```

has matched "tweedledum tweedledee" the value of the captured substring is "tweedledee". However, if there are nested capturing subpatterns, the corresponding captured values may have been set in previous iterations. For example, after

```
/(a|(b))+/
```

matches "aba" the value of the second captured substring is "b".

Note: This topic was taken from the PCRE library manual. The PCRE library is open source software, written by Philip Hazel <ph10@cam.ac.uk>, and copyright by the University of Cambridge, England.



## Back References

Outside a character class, a backslash followed by a digit greater than 0 (and possibly further digits) is a back reference to a capturing subpattern earlier (i.e. to its left) in the pattern, provided there have been that many previous capturing left parentheses.

However, if the decimal number following the backslash is less than 10, it is always taken as a back reference, and causes an error only if there are not that many capturing left parentheses in the entire pattern. In other words, the parentheses that are referenced need not be to the left of the reference for numbers less than 10. See the section entitled "Backslash" above for further details of the handling of digits following a backslash.

A back reference matches whatever actually matched the capturing subpattern in the current subject string, rather than anything matching the subpattern itself. So the pattern

```
(sens|respons)e and \1libility
```

matches "sense and sensibility" and "response and responsibility", but not "sense and responsibility". If careful matching is in force at the time of the back reference, then the case of letters is relevant. For example,

```
((?i)rah)\s+\1
```

matches "rah rah" and "RAH RAH", but not "RAH rah", even though the original capturing subpattern is matched caselessly.

There may be more than one back reference to the same subpattern. If a subpattern has not actually been used in a particular match, then any back references to it always fail. For example, the pattern

```
(a|(bc))\2
```

always fails if it starts to match "a" rather than "bc". Because there may be up to 99 back references, all digits following the backslash are taken as part of a potential back reference number. If the pattern continues with a digit character, then some delimiter must be used to terminate the back reference. If the PCRE\_EXTENDED option is set, this can be whitespace. Otherwise an empty comment can be used.

A back reference that occurs inside the parentheses to which it refers fails when the subpattern is first used, so, for example, (a\1) never matches. However, such references can be useful inside repeated subpatterns. For example, the pattern

```
(a|b\1)+
```

matches any number of "a"s and also "aba", "ababaa" etc. At each iteration of the subpattern, the back reference matches the character string corresponding to the previous iteration. In order for this to work, the pattern must be such that the first iteration does not need to match the back reference. This can be done using alternation, as in the example above, or by a quantifier with a minimum of zero.

Note: This topic was taken from the PCRE library manual. The PCRE library is open source software, written by Philip Hazel <ph10@cam.ac.uk>, and copyright by the University of Cambridge, England.

## Assertions

An assertion is a test on the characters following or preceding the current matching point that does not actually consume any characters. The simple assertions coded as `\b`, `\B`, `\A`, `\Z`, `\z`, `^` and `$` are described above. More complicated assertions are coded as subpatterns. There are two kinds: those that look ahead of the current position in the subject string, and those that look behind it.

An assertion subpattern is matched in the normal way, except that it does not cause the current matching position to be changed. Lookahead assertions start with `(?=` for positive assertions and `(?!` for negative assertions. For example,

```
\w+(?=;)
```

matches a word followed by a semicolon, but does not include the semicolon in the match, and

```
foo(?!bar)
```

matches any occurrence of "foo" that is not followed by "bar". Note that the apparently similar pattern

```
(?!foo)bar
```

does not find an occurrence of "bar" that is preceded by something other than "foo"; it finds any occurrence of "bar" whatsoever, because the assertion `(?!foo)` is always true when the next three characters are "bar". A lookbehind assertion is needed to achieve this effect.

Lookbehind assertions start with `(?<=` for positive assertions and `(?<!` for negative assertions. For example,

```
(?<!foo)bar
```

does find an occurrence of "bar" that is not preceded by "foo". The contents of a lookbehind assertion are restricted such that all the strings it matches must have a fixed length. However, if there are several alternatives, they do not all have to have the same fixed length. Thus

```
(?<=bullock|donkey)
```

is permitted, but

```
(?<!dogs?|cats?)
```

causes an error at compile time. Branches that match different length strings are permitted only at the top level of a lookbehind assertion. This is an extension compared with Perl 5.005, which requires all branches to match the same length of string. An assertion such as

```
(?<=ab(c|de))
```

is not permitted, because its single top-level branch can match two different lengths, but it is acceptable if rewritten to use two top-level branches:

```
(?<=abc|abde)
```

The implementation of lookbehind assertions is, for each alternative, to temporarily move the current position back by the fixed width and then try to match. If there are insufficient characters before the current position, the match is deemed to fail. Lookbehinds in conjunction with once-only subpatterns can be particularly useful for matching at the ends of strings; an example is given at the end of the section on once-only subpatterns.

Several assertions (of any sort) may occur in succession. For example,

```
(?<=\d{3})(?<!999)foo
```

matches "foo" preceded by three digits that are not "999". Furthermore, assertions can be nested in any combination. For example,

```
(?<=(?<!foo)bar)baz
```

matches an occurrence of "baz" that is preceded by "bar" which in turn is not preceded by "foo".

Assertion subpatterns are not capturing subpatterns, and may not be repeated, because it makes no sense to assert the same thing several times. If an assertion contains capturing subpatterns within it, these are always counted for the purposes of numbering the capturing subpatterns in the whole pattern. Substring capturing is carried out for positive assertions, but it does not make sense for negative assertions.

Assertions count towards the maximum of 200 parenthesized subpatterns.

Note: This topic was taken from the PCRE library manual. The PCRE library is open source software, written by Philip Hazel <ph10@cam.ac.uk>, and copyright by the University of Cambridge, England.

## Once-Only Subpatterns

With both maximizing and minimizing repetition, failure of what follows normally causes the repeated item to be re-evaluated to see if a different number of repeats allows the rest of the pattern to match. Sometimes it is useful to prevent this, either to change the nature of the match, or to cause it fail earlier than it otherwise might, when the author of the pattern knows there is no point in carrying on.

Consider, for example, the pattern `\d+foo` when applied to the subject line

```
123456bar
```

After matching all 6 digits and then failing to match "foo", the normal action of the matcher is to try again with only 5 digits matching the `\d+` item, and then with 4, and so on, before ultimately failing. Once-only subpatterns provide the means for specifying that once a portion of the pattern has matched, it is not to be re-evaluated in this way, so the matcher would give up immediately on failing to match "foo" the first time. The notation is another kind of special parenthesis, starting with `(?>` as in this example:

```
(?>\d+)bar
```

This kind of parenthesis "locks up" the part of the pattern it contains once it has matched, and a failure further into the pattern is prevented from backtracking into it. Backtracking past it to previous items, however, works as normal.

An alternative description is that a subpattern of this type matches the string of characters that an identical standalone pattern would match, if anchored at the current point in the subject string.

Once-only subpatterns are not capturing subpatterns. Simple cases such as the above example can be thought of as a maximizing repeat that must swallow everything it can. So, while both `\d+` and `\d+?` are prepared to adjust the number of digits they match in order to make the rest of the pattern match, `(?>\d+)` can only match an entire sequence of digits.

This construction can of course contain arbitrarily complicated subpatterns, and it can be nested.

Once-only subpatterns can be used in conjunction with lookbehind assertions to specify efficient matching at the end of the subject string. Consider a simple pattern such as

```
abcd$
```

when applied to a long string which does not match it. Because matching proceeds from left to right, PCRE will look for each "a" in the subject and then see if what follows matches the rest of the pattern. If the pattern is specified as

```
^.*abcd$
```

then the initial `.*` matches the entire string at first, but when this fails, it backtracks to match all but the last character, then all but the last two characters, and so on. Once again the search for "a" covers the entire string, from right to left, so we are no better off. However, if the pattern is written as

```
^(?>.*)(?<=abcd)
```

then there can be no backtracking for the `.*` item; it can match only the entire string. The subsequent lookbehind assertion does a single test on the last four characters. If it fails, the match fails immediately. For long strings, this approach makes a significant difference to the processing time.

Note: This topic was taken from the PCRE library manual. The PCRE library is open source software, written by

Philip Hazel <ph10@cam.ac.uk>, and copyright by the University of Cambridge, England.

## Conditional Subpatterns

It is possible to cause the matching process to obey a subpattern conditionally or to choose between two alternative subpatterns, depending on the result of an assertion, or whether a previous capturing subpattern matched or not. The two possible forms of conditional subpattern are

```
(?(condition)yes-pattern)
(?(condition)yes-pattern|no-pattern)
```

If the condition is satisfied, the yes-pattern is used; otherwise the no-pattern (if present) is used. If there are more than two alternatives in the subpattern, a compile-time error occurs.

There are two kinds of condition. If the text between the parentheses consists of a sequence of digits, then the condition is satisfied if the capturing subpattern of that number has previously matched. Consider the following pattern, which contains non-significant white space to make it more readable (assume the PCRE\_EXTENDED option) and to divide it into three parts for ease of discussion:

```
(\()?  [^()]+  (?(1)\))
```

The first part matches an optional opening parenthesis, and if that character is present, sets it as the first captured substring. The second part matches one or more characters that are not parentheses. The third part is a conditional subpattern that tests whether the first set of parentheses matched or not. If they did, that is, if subject started with an opening parenthesis, the condition is true, and so the yes-pattern is executed and a closing parenthesis is required. Otherwise, since no-pattern is not present, the subpattern matches nothing. In other words, this pattern matches a sequence of non-parentheses, optionally enclosed in parentheses.

If the condition is not a sequence of digits, it must be an assertion. This may be a positive or negative lookahead or lookbehind assertion. Consider this pattern, again containing non-significant white space, and with the two alternatives on the second line:

```
(?(?=[^a-z]*[a-z])
\d{2}[a-z]{3}-\d{2}  |  \d{2}-\d{2}-\d{2} )
```

The condition is a positive lookahead assertion that matches an optional sequence of non-letters followed by a letter. In other words, it tests for the presence of at least one letter in the subject. If a letter is found, the subject is matched against the first alternative; otherwise it is matched against the second. This pattern matches strings in one of the two forms dd-aaa-dd or dd-dd-dd, where aaa are letters and dd are digits.

Note: This topic was taken from the PCRE library manual. The PCRE library is open source software, written by Philip Hazel <ph10@cam.ac.uk>, and copyright by the University of Cambridge, England.

## Comments

The sequence `(?#` marks the start of a comment which continues up to the next closing parenthesis. Nested parentheses are not permitted. The characters that make up a comment play no part in the pattern matching at all.

If the `PCRE_EXTENDED` option is set, an unescaped `#` character outside a character class introduces a comment that continues up to the next newline character in the pattern.

Note: This topic was taken from the PCRE library manual. The PCRE library is open source software, written by Philip Hazel <ph10@cam.ac.uk>, and copyright by the University of Cambridge, England.

## Performance

Certain items that may appear in patterns are more efficient than others. It is more efficient to use a character class like [aeiou] than a set of alternatives such as (a|e|i|o|u). In general, the simplest construction that provides the required behaviour is usually the most efficient. Jeffrey Friedl's book contains a lot of discussion about optimizing regular expressions for efficient performance.

When a pattern begins with .\* and the PCRE\_DOTALL option is set, the pattern is implicitly anchored by PCRE, since it can match only at the start of a subject string. However, if PCRE\_DOTALL is not set, PCRE cannot make this optimization, because the . metacharacter does not then match a newline, and if the subject string contains newlines, the pattern may match from the character immediately following one of them instead of from the very start. For example, the pattern

(.\*) second

matches the subject "first\nand second" (where \n stands for a newline character) with the first captured substring being "and". In order to do this, PCRE has to retry the match starting after every newline in the subject.

If you are using such a pattern with subject strings that do not contain newlines, the best performance is obtained by setting PCRE\_DOTALL, or starting the pattern with ^.\* to indicate explicit anchoring. That saves PCRE from having to scan along the subject looking for a newline to restart at.

Note: This topic was taken from the PCRE library manual. The PCRE library is open source software, written by Philip Hazel <ph10@cam.ac.uk>, and copyright by the University of Cambridge, England.



## Limitations

There are some size limitations in PCRE but it is hoped that they will never in practice be relevant. The maximum length of a compiled pattern is 65539 (sic) bytes. All values in repeating quantifiers must be less than 65536. The maximum number of capturing subpatterns is 99. The maximum number of all parenthesized subpatterns, including capturing subpatterns, assertions, and other types of subpattern, is 200.

The maximum length of a subject string is the largest positive number that an integer variable can hold. However, PCRE uses recursion to handle subpatterns and indefinite repetition. This means that the available stack space may limit the size of a subject string that can be processed by certain patterns.

Note: This topic was taken from the PCRE library manual. The PCRE library is open source software, written by Philip Hazel <ph10@cam.ac.uk>, and copyright by the University of Cambridge, England.

## Differences From Perl

The differences described here are with respect to Perl 5.005.

1. By default, a whitespace character is any character that the C library function **isspace** recognizes, though it is possible to compile PCRE with alternative character type tables. Normally **isspace** matches space, formfeed, newline, carriage return, horizontal tab, and vertical tab. Perl 5 no longer includes vertical tab in its set of whitespace characters. The `\v` escape that was in the Perl documentation for a long time was never in fact recognized. However, the character itself was treated as whitespace at least up to 5.002. In 5.004 and 5.005 it does not match `\s`.

2. PCRE does not allow repeat quantifiers on lookahead assertions. Perl permits them, but they do not mean what you might think. For example, `(?!a){3}` does not assert that the next three characters are not "a". It just asserts that the next character is not "a" three times.

3. Capturing subpatterns that occur inside negative lookahead assertions are counted, but their entries in the offsets vector are never set. Perl sets its numerical variables from any such patterns that are matched before the assertion fails to match something (thereby succeeding), but only if the negative lookahead assertion contains just one branch.

4. Though binary zero characters are supported in the subject string, they are not allowed in a pattern string because it is passed as a normal C string, terminated by zero. The escape sequence `"\0"` can be used in the pattern to represent a binary zero.

5. The following Perl escape sequences are not supported: `\l`, `\u`, `\L`, `\U`, `\E`, `\Q`. In fact these are implemented by Perl's general string-handling and are not part of its pattern matching engine.

6. The Perl `\G` assertion is not supported as it is not relevant to single pattern matches.

7. Fairly obviously, PCRE does not support the `(?{code})` construction.

8. There are at the time of writing some oddities in Perl 5.005\_02 concerned with the settings of captured strings when part of a pattern is repeated. For example, matching "aba" against the pattern `/(a(b)?)+$/` sets \$2 to the value "b", but matching "aabbaa" against `/(aa(bb)?)+$/` leaves \$2 unset. However, if the pattern is changed to `/(aa(b(b)))+$/` then \$2 (and \$3) get set.

In Perl 5.004 \$2 is set in both cases, and that is also true of PCRE. If in the future Perl changes to a consistent state that is different, PCRE may change to follow.

9. Another as yet unresolved discrepancy is that in Perl 5.005\_02 the pattern `/(a)?(?(1)a|b)+$/` matches the string "a", whereas in PCRE it does not. However, in both Perl and PCRE `/(a)?a/` matched against "a" leaves \$1 unset.

10. PCRE provides some extensions to the Perl regular expression facilities:

(a) Although lookbehind assertions must match fixed length strings, each alternative branch of a lookbehind assertion can match a different length of string. Perl 5.005 requires them all to have the same length.

(b) If `PCRE_DOLLAR_ENDONLY` is set and `PCRE_MULTILINE` is not set, the `$` meta-character matches only at the very end of the string.

(c) If `PCRE_EXTRA` is set, a backslash followed by a letter with no special meaning is faulted.

(d) If `PCRE_UNGREEDY` is set, the greediness of the repetition quantifiers is inverted, that is, by default they are not greedy, but if followed by a question mark they are.

Note: This topic was taken from the PCRE library manual. The PCRE library is open source software, written by

Philip Hazel <ph10@cam.ac.uk>, and copyright by the University of Cambridge, England.

## iKey

available in Ritlabs AuthenticBat! and Ritlabs SecureBat! only

Most computers and networks use simple usernames and passwords to protect themselves. But passwords alone do not provide adequate protection – they are easily shared or guessed. Rainbow Technologies' iKey1000 token was designed specifically as a portable and secure authentication token addressing the password replacement needs of both users and systems administrators.

The iKey1000 hardware token is an integrated circuit housing a processor, non-volatile random access memory and a USB interface to a PC. It provides the security of smart cards without the hassle and cost of a reader. The token is small and lightweight, making it easy to carry on a key chain or in a daily planner.

The iKey1000 is a portable two-factor authentication token ideally suited for email security.

Each iKey contains a unique 64-bit factory-installed serial number that differentiates an individual token from all other tokens. An iKey also has customisable friendly name to differentiate them in more convenient and friendly way, e.g. "Bob's iKey."

An iKey can stay in one of three security states: Security Officer State, User Security State and Guest Security State. The security state of the token defines what kinds of actions it can be used to perform. Once plugged in, the token sets itself to Guest Security State until the User PIN or Security Officer (SO) PIN has been entered and verified.

## Customising a New iKey Token

available in Ritlabs AuthenticBat! and Ritlabs SecureBat! only

If you received an iKey token from your system administrator, simply follow the instructions provided by the administrator. You will also receive a User PIN (Personal Identification Number) for your iKey.

If you received a factory-initialised token, you should first change the User PIN and SO PIN. The token is shipped from the factory with the following settings: User PIN: 12345678, SO PIN: rainbow.

## Security Officer PIN

available in Ritlabs AuthenticBat! and Ritlabs SecureBat! only

The Security Officer (usually a system administrator) is the person who is entrusted with initialising and configuring the iKey. The token is placed in the Security Officer State when the Security Officer (SO) PIN has been entered and verified. This is the highest-level security state. When the SO PIN has been entered and verified and the token is in the SO state, in addition to performing actions are only available to the Security Officer, any actions that are valid for the User Security State can also be performed.

The person acting as Security Officer can configure the token to insist upon entry of the user PIN to perform certain actions. The Security Officer can also define the maximum number of times invalid user PINs can be entered before the user is blocked from entering a PIN. Once a user is blocked from entering a user PIN, only the Security Officer can reset the token to unblock user PIN entry.

**Warning!** Since the SO PIN offers the highest level of security, if a Security Officer forgets the SO PIN, the SO PIN cannot be retrieved from the iKey, and the token cannot be configured.

## User PIN

available in Ritlabs AuthenticBat! and Ritlabs SecureBat! only

The iKey is placed in the User Security State when the user PIN has been entered and verified. Users are able to set and update their own user PINs.

When a user enters an invalid user PIN, an iKey counter decrements. The counter is reset when a valid user PIN is again entered, but if the counter decrements to zero, the user is blocked from entering a user PIN. Only the Security Officer can reset the token so that user PINs can again be entered. The factory-default value for the counter is 5.

## Using iKey for POP3/SMTP Authentication

available in Ritlabs AuthenticBat! and Ritlabs SecureBat! only

In **Account Properties**, select the **Transport** tab and click the **Authentication** button in the **Send Mail** or **Receive Mail** group box. If the iKey driver and the API have been properly installed on your computer, you will see a selectable checkbox for iKey: **iKey MD-5 CRAM-HMAC Challenge/Response** in the **Mail Retrieval (POP3) Authentication** configuration window or a **Store password on iKey** option in the **Mail Transmission (SMTP) Authentication** configuration window.

Once the iKey checkbox has been ticked, the Authentication configuration window will change its content to display two new group boxes:

The **“iKey Token to Use”** group box displays the serial number and friendly name of the token selected to perform POP3 or SMTP authentication for that account. The serial number of the selected iKey is stored in the account configuration file. The Bat! won't perform authentication with that account using a token with a different serial number unless you click the “Browse” button and select another iKey. In order to display the friendly name, The Bat! requires that the token with a given serial number is inserted and available (i.e. it is not in use by another application), because the friendly name is retrieved directly from the token. If the friendly name can't be displayed (e.g. the token with a selected serial number wasn't plugged in or it was unavailable), you can try to resolve the problem (e.g. plug in the token with the valid serial number). and press the “Refresh” button to retry the attempt to display the friendly name.

The **“iKey Password”** group box offers three buttons to Set, Change or Remove a password that is stored on the iKey token. Once set, the password can only be used for authentication with that account, but it cannot be retrieved. However, the password can be changed (replaced by another password when, for instance, the server administrator has assigned a new password for your account) or removed (if you no longer wish to use this token for authentication with that account).



## Browsing for an iKey

available in Ritlabs AuthenticBat! and Ritlabs SecureBat! only

There can be several iKey tokens plugged into your computer. You may however choose which token to use for secure POP3/SMTP authentication for a specific account.

When you ‘browse’ for an iKey token to use, you will see a list of available (plugged in) iKey tokens. The tokens that are plugged in but currently exclusively used by other applications are also displayed, but are not available for selection, editing or viewing. You can however press the “Refresh” button to update the current status of the iKey tokens in the list, e.g. to see whether an application that was previously using a particular token has released ownership, making it available for The Bat!

When a token is available, you can click the “View” button to see its technical specification and the amount of available token memory. The total amount of memory depends on which iKey model is used and, at the time of writing, these are 8Kbytes for the iKey1000 and 32Kbytes for iKey1032.

Clicking the “Edit” button in a list of available iKey tokens invokes the built-in iKey Editor.

## Editing an iKey

available in Ritlabs AuthenticBat! and Ritlabs SecureBat! only

The Bat! has its own, built-in iKey Editor which can be invoked while browsing a list of available iKey tokens.

While the iKey Editor is a tool intended mostly for use by Security Officers (persons authorised to perform Security Officer State operations), it also allows the users (persons authorised to perform User Security State operations) to change the User PIN.

The Security Officer can Set Token Name, Change Security Officer PIN and Unblock User PIN.  
The users can change the User PIN.

## Change User PIN

available in Ritlabs AuthenticBat! and Ritlabs SecureBat! only

The “**Change User PIN**” button allows you to change the User PIN. You will be first asked to enter the current User PIN. The factory default User PIN is 12345678. Next, you will be asked to enter the new PIN and to confirm it by typing it again. Type a number from one to eight digits (you may only use the characters 0-9).

## Set Token Name

available in Ritlabs AuthenticBat! and Ritlabs SecureBat! only

The “**Set Token Name**” button allows you to set a friendly name for the iKey. Type in a short description to give the token a name, like "Bob's iKey." You can enter up to 16 alphanumeric characters.

## Change Security Officer PIN

available in Ritlabs AuthenticBat! and Ritlabs SecureBat! only

The “**Change Security Officer PIN**” button allows you (or the Security Officer) to change the SO PIN. You will first be asked to enter the current SO PIN. The factory default SO PIN is `rainbow`. Next, you will be asked to enter the new SO PIN and to confirm it by typing it again. Type a PIN consisting of from 6 to 25 letters or numbers.

## Unblock User PIN

available in Ritlabs AuthenticBat! and Ritlabs SecureBat! only

The “**Unblock User PIN**” button allows the Security Officer to unblock the User PIN after the retry counter has been decremented to zero, or if the User has forgotten the PIN. To unblock the User PIN you must create a new user PIN for the iKey. You will be first asked to enter the current Security Officer PIN. Next, you will be asked to enter the new PIN and to confirm it by typing it again. Type a number consisting of from one to eight digits (you may only use the characters 0-9).

## Running Ritlabs SecureBat!

available in Ritlabs SecureBat! only

When the user starts Ritlabs SecureBat!, the program looks for the token that contains the assigned ID, reads the ID from the token, and uses the 128-bit encryption key from that ID for on-the-fly encryption of the message base, address books and configuration files. Make sure that the token is plugged in while working with Ritlabs SecureBat! Once the token is unplugged, the program hides all its windows and prompts the user, asking whether they want to exit (losing changes) or to plug the token back and continue working with Ritlabs SecureBat!. The user may, however, use this feature as an emergency exit from the program.

## SecureBat! ID

available in Ritlabs SecureBat! only

SecureBat! ID is a small record that is used as a key for transparent, on-the-fly encryption of Ritlabs SecureBat! operations. Each ID consists of three components: a 128-bit encryption key, a distinctive User Name and a textual Description. The IDs are stored on iKey tokens and retrieved from the tokens by Ritlabs SecureBat! at start-up. The IDs may also be archived by the Security Officer in the Token Manager ID store file.



## Assigning the ID

available in Ritlabs SecureBat! only

The Ritlabs SecureBat! installer prompts the user to assign an ID from a token to the newly installed copy. Once installed, Ritlabs SecureBat! is permanently tied to the User Name of a specific ID, and it is not possible to reassign another ID to the existing Ritlabs SecureBat!. The 128-bit encryption key from the assigned ID is used for on-the-fly encryption of Ritlabs SecureBat! operations.

## Installing SecureBat!

available in Ritlabs SecureBat! only

The Security Officer does not need to install each copy of Ritlabs SecureBat! – all installation steps can be easily performed by the users. During the installation, a user assigns an ID to Ritlabs SecureBat! and this ID will be used for all Ritlabs SecureBat! operations. To assign an ID, the user should first select a token and then choose an ID stored on that token. After this assignment has been made, the installation continues.

## Selecting a Token

available in Ritlabs SecureBat! only

Once the user runs the installer, a window appears that prompts the user to **Select iKey Token**. The window displays a list of all inserted iKey tokens. In most cases, only one token is inserted, however, the window shows the serial number and the friendly name for each currently inserted token. It helps to more clearly identify the required token and avoid confusion when an inappropriate token has been inserted by accident.

## Selecting an ID from a Token

available in Ritlabs SecureBat! only

When the user has selected a token from the list, Ritlabs SecureBat! displays all of the IDs that were copied to the selected token by the Security Officer. The User Name and the Description of each ID are listed. Please note that the token must contain at least one ID. Whichever ID is selected is then assigned and used to continue the installation. It is extremely important to understand that it is not possible to reassign another ID to the Ritlabs SecureBat! after this point so double check carefully when selecting an ID.



