CrashTick

COLLABORATORS			
	TITLE : CrashTick		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		February 12, 2023	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

CrashTick

1.1 CrashTick.guide

```
                        CrashTick 1.0
   Copyright © 1996 Johan Billing
        All rights reserved
```

1.2 Introduction

```
Introduction
============
Welcome to CrashTick, my favourite tick program for the Amiga! :-)

CrashTick is a program for handling file echos, the method used for
distributing files in Fidonet. Such a program is normally called a tick
program, since Tick was the name of the first program that used this
method. CrashTick has the functions you find in most tick programs plus a
few others which are not as common on the Amiga. For example CrashTick can
handle packed tick files and has a built-in Raid which lets your downlinks
connect to areas themselves withour having to bother you. As an extra
bonus, CrashTick also has a comfortable settings editor called
CrashTickPrefs.

Main features
=============
o Powerful Announce function
o Built-in Raid
o Comfortable settings editor
o Can handle packed tick files
```

1.3 License and warranty

```
Distribution rules
==================
CrashTick may be distributed freely as long as the following rules
are followed:

1. CrashTick must be distributed together with all its
   accompanying files such as tools, documentation and
   Arexx scripts as found in the original archive.
```

```
2. CrashTick may only be distributed as long as the program
   itself and the other files are not modified in any way.

3. You may not sell CrashTick for profit. It is allowed to charge
   a small fee to cover distribution costs.

Usage rules
===========
After quite a lot of thought, I have decided to release CrashTick as
a free program that doesn’t cost anything. That doesn’t mean that
I don’t want money, I would be very happy if someone sent me money
or other nice things. If you really like the program, why not do it? :-)

(Especially Mike Oldfield singles would be much appreciated. :-))

Warranty
========
There is no warranty for CrashTick. All use of CrashTick is
on your own risk and the author cannot be held responsible
for any damage caused by CrashTick.
```

1.4 Requirements

```
CrashTick requires the following non-standard libraries:

o gtlayout.library v24 or higher by Olaf Barthel
o reqtools.library v38 or higher by Nico François and Magnus Holmgren
```

1.5 Installation

```
The CrashTick installation script will take care of the installation for
you. After CrashTick has been installed, you should run CrashTickPrefs to
create a new configuration.
```

1.6 Interfacing with your tosser

```
Interfacing with your tosser
============================

Writing messages
================
Both CrashTick’s built-in Raid and the Announce function need to create new
messages. If you don’t want to use them, you can stop reading here.

CrashTick writes message by generating PKT-files and placing them in your
inbound directory where they later should be processed by your tosser.
What you have to do depends on what kind of tosser that you run...

CrashMail
```

```
---------
If you are using CrashMail, you don’t have to do anything at all since
CrashTick sets the file comment "No security" for the created PKT-files.
When CrashMail finds this comment, all security checks are disabled.
If you don’t want CrashMail to say that the packets are from 0:0/0.0,
set one of your own akas as the fake node.

Point programs
--------------
If you are using a simple point program without any security checks,
you most probably don’t have to to anything at all.

Other tossers
-------------
Set the fake node to a node that isn’t used by any real nodes, for example
9:9/9.9. Now you have to make sure that this node is allowed to write in
the areas where you want to announce files. If you are concerned about
security, you can also use a packet password.

Reading messages
================
If you want to use the Raid, CrashTick of course has to be able to read
messages addressed to Raid. Your tosser must then generate messages in
RFC-format, a text-based message format similar to the format used in
Internet news. To configure Raid in CrashMail, you have to create a new
Robotname like this:

Robotname: Raid (or something else, whatever you like)
Command: MAIL:CrashTick %r

Instead of executing CrashTick directly, I use a small script that makes
sure that CrashMail tosses the PKT files created by Raid directly:

---
.key rfc

MAIL:CrashTick raid <rfc>
run rx "address ’CRASHMAIL’ toss"
---

I have saved this as MAIL:Scripts/Raid and use the following command
instead: "Execute MAIL:Scripts/Raid %r"
```

1.7 Configuration

```
               CrashTickPrefs settings editor
                CrashTick.prefs
================
The default configuration file is "PROGDIR:CrashTick.prefs", that is
CrashTick.prefs in the directory where you keep CrashTick. If that file is
not found, CrashTick and CrashTickPrefs will also try to load the file
"MAIL:CrashTick.prefs". You can specify other configuration files on
command-line with the SETTINGS switch.
```

```
 File notification
 =================
 When CrashTick is started with the WAIT switch waits for ARexx commands,
 it uses a feature of AmigaDOS called file notification. If the
 configuration file is changed, CrashTick will notice and re-read it
 automatically. That means that if you alter the configuration with
 CrashTickPrefs when CrashTick is waiting for ARexx commands, CrashTick
 will load the new configuration automatically.

 Format of settings file
 =======================
 The configuration file is in ASCII-format, so you can modify it with a
 texteditor, but that is not recommended unless you know what you are
 doing. I suggest that you use CrashTickPrefs as much as possible.

 Any lines beginning with a ";" are treated as comments and ignored.
 They will be lost whenever the configuration file is rewritten.
```

1.8 CrashTickPrefs

```
                CrashTickPrefs
==============
With CrashTickPrefs, you can change all settings in CrashTick with
a comfortable GUI. CrashTickPrefs has online-help, so if you don’t
understand something, all you have to to is to press the Help key
and help for the current window will be displayed.

Sections
========


                Creating a new config

                Switching from another tick program

                General...

                Commands...

                Paths...

                Options...

                Groups...

                Akas...

                Nodes...

                Areas...

                Announce...
                Other
=====
                Hotkeys

                Menus

                Startup
```

1.9 CrashTickPrefs -- Hotkeys

```
Hotkeys
=======
You can access all hotkeys in a window even when a string gadget is
active. All you have to do is keep right Amiga pressed down and
press the hotkey.

The arrow-down gadgets that sometimes can be found next to stringgadgets
and textgadgets can also be selected with the keyboard. Just keep Shift
pressed down and press the hotkey.

The Tab key always activates the first string gadget if no string
gadget is active.
```

1.10 CrashTickPrefs -- Starting CrashTickPrefs

```
Startup
=======

CLI
---
Usage: SETTINGS,PUBSCREEN=PS/K,HELPFILE/K:

 SETTINGS    Settings file to load at startup. CrashTickPrefs by default
             tries to load CrashTick.prefs in PROGDIR: and MAIL:.

 PUBSCREEN   Public Screen to open CrashTickPrefs on. CrashTickPrefs
             defaults to the default public screen.

 HELPFILE    The name of the helpfile that CrashPrefs should use. By
             default, CrashTick first tries lo load PROGDIR:CrashTick.guide
             and if that file is not found, CrashTickPrefs also tries to
             load LOCALE:help/english/CrashTick.guide

Workbench
---------
The CLI options SETTINGS, PUBSCREEN and HELPFILE can also be passed to
CrashTickPrefs with Workbench tooltypes. Select the CrashTickPrefs icon
and select Icons->Information in Workbench to edit the tooltypes.
```

1.11 CrashTickPrefs -- Menus

```
                Menus
=====

Open

   With this item you can select a new CrashTick settings file to load.

Save

   Save the current settings file with its old name.

Save As

   Save the current settings file with a new name.

About

   Show information about CrashTickPrefs.

Quit

   Quit CrashPrefs.

Import old tick configuration

   With this menuitem you can import a tick configuration. For more
   information about this, read
               Switching from another Tick program
```

1.12 CrashTickPrefs -- Creating a new config

```
If CrashTickPrefs cannot load the configuration file at startup, it
will ask you if you want to load another or if you want to create a
new configuration file.

If you select to create a new configuration file, CrashTickPrefs will
ask you the following questions:

Name

  Here you just have to enter your name.

Main node address

  This is your own address. If you have more than one address, just
  enter one of them and add the others later.

Main feed

  Here you should enter the node you receive file echos from. If you
  get file echos from more than one node, the others can be added
  in CrashTickPrefs later.
```

Directory for auto-added areas

Here you have to enter the directory CrashTick should use for auto-added areas. When the directory is created, the code "%a" will be replaced with the name of the area. Example: MAIL:Files/%a

Now a basic configuration file has been created. You should check the new settings and add other akas and nodes that you might have.

1.13 Switching from another Tick program

If you used another Tick program before, CrashTickPrefs may be able to at least import parts of the config. In the main window of CrashTickPrefs, there is a menu item called "Import old tick configuration". That option has only been tested with FTick, but it should be able to import a SkyTick configuration too since they are very similar. It is possible that it can import the configuration from other programs too, you can always try...

Read here to see what keywords are imported:

```
"AKA <node>" or "HERE <node>"
=============================
```
Adds the specified node as one of you akas. The correct aka will be set for all imported areas.

```
"FILEECHO <name> <directory>" or "AREA <name>/PATH <directory>"
===============================================================
```
Adds the area to your configuration.

```
"TO <node> [<flags>]"
=====================
```
Adds the specified node to the list of nodes that receive the area. CrashTickPrefs understands the following flags:

```
*    The node will be added as "Incoming files only"
X    The node will be added as "Outgoing files only"
```

```
"PW <node> <config>" or "PASSWORD <node> <config>"
==================================================
```
Sets the password specified here for the node.

1.14 CrashTickPrefs -- General

Sysop

The name of the sysop on the system where CrashTick is running. This is the name used when announcing files and answering to Raid requests. You should set this to your own name so that netmail replies to those messages will reach you.

Max wait before pickup

If you are using a filebase, this is the maximum number of days that files will be stored there before they will be deleted when you run "CrashTick maint". The files will of course also be deleted when they have been sent to all nodes. To find out more about the filebase, read the section
Using a filebase
.

Logfile

The logfile CrashTick should write information to. In many cases CrashTick unoperates unattended, and then a logfile can be very useful, especially if something goes wrong.

Loglevel

With this gadget, you specify how much information CrashTick should write to its logfile. There are the following levels:

1	Minimum
2	Small
3	Sparse
4	Normal
5	Extra
6	Debug

Maximum lines in log buffer
Maximum seconds to keep buffer

CrashTick can use a "buffered" logfile to make writes to the logfile faster. This has the disadvantage that anything written to the logfile after it was flushed probably will be lost if your computer crashes. The default settings are a resonable compromise between speed and security.

Basically, buffering works like this: CrashTick keeps the logfile open between writes. Flushing just means that the logfile is closed so that CrashTick has to open it again the next time a string is written to it. The logfile will be flushed if the number of lines written to it since it was last flushed exceeds the number specified here or if the number of seconds since it was last flushed exceeds the limit specified here.

Fake node
Fake password

When answering to a Raid request or when announcing files, CrashTick will leave .PKT files in your inbound so that they can be processed by your tosser. The .PKT files will then look as if they came from the fake node and have the fake password set. Read the section about

Interfacing with your tosser
for more information
about this.

1.15 CrashTickPrefs -- Commands

```
                Import command

   This command will be executed every time CrashTick has imported a file.
   It is intended to be used to import the file to an area in your BBS
   program. You can use
               %-codes
                to pass information about the file to the
   command. When you have found out how to import files to your BBS
   program, don’t forget to mail me and tell me how you did it so that I
   can include that in the manual.

Replace command

   Tick files can contain information about files they replace. If for
   example version 1.1 of a program was hatched in a file echo, the
   accompanying tick file could say that this file should replace version
   1.0 of the program. If a tick file contains such information and you
   haven’t disabled replacing of files, this command will be executed.

               %-codes
                can be used here too.

   There are two things you should note here:

   1. CrashTick doesn’t delete the old file, your command has to take
      care of that.

   2. The name of the replaced file may contain wildcards according
      to the specifications, but I have never seen that myself.

Zip extract
Zip add
LhA extract
LhA add

   These are the commands used to execute your packer if you want to
   send packed files or be able to unpack incoming archives. The
   following %-codes can be used here:

   %a      Name of the archive
   %f      Name of the file (only when adding files to an archive)
```

1.16 %-codes

```
%-codes
=======

  %a       Area name
  %n       File name
  %p       Directory where the file is placed
  %s       Full path to the file (use this instead of %p/%n)
  %b       File size in bytes
  %o       Originating node, the node who hatched the file
  %f       From node, the node you got the file from
  %m       Magic filename (if one is specified in the .tic file)
  %d       File description
```

When executing the replace command, %n, %p and %s all point to the file that should be replaced. The other %-codes contain information about the new file.

1.17 CrashTickPrefs -- Paths

Inbound

This is the directory where CrashTick will look for incoming tick files. Any .PKT files generated by CrashTick will also be put here.

Outbound

CrashTick will write the flow files to this directory. This is the directory that should be configured as Outbound in your mailer too.

Tick directory

This is the directory where outgoing tick files will be stored until they are sent. This can be set to the same directory to your outbound, but it isn’t necessary. If you send files to lots of nodes, setting this to a different directory is a good idea since the mailer will scan your outbound directory faster when a downlink calls.

File base

If you want to use a filebase, you can set the path to it here. If you don’t want to use a filebase, just enter an empty string here. Read the section
Using a filebase
for more information about the filebase.

Bad tick files

CrashTick will move any files it considers "bad" here. Files can be bad if CrashTick doesn’t understand the tickfile, the checksum is wrong or if they are coming from a node which hasn’t got the required access to send files.

Auto-add directory

This is the directory CrashTick should use for auto-added areas. When the directory is created, the code "%a" will be replaced with the name of the area. Example: MAIL:Files/%a

Statistics file

The name of the file where CrashTick should store the statistics. The statistics are shown in the lists generated by the Raid and can be

displayed anytime with the
CrashTickStats
command.

Raid helpfile

This is the textfile that should be send when a node requests help
from your Raid using the %HELP command. A default helpfile called
RaidHelp.doc is included in the archive.

1.18 CrashTickPrefs -- Options

Check CRC

If this is switch is turned on, CrashTick will calculate the checksum
for incoming files and compare it with the checksum in the tickfile.
If the checksums don't match, the file has probably been changed
somewhere on its way to you and CrashTick will move it to the directory
for bad files.

Allow replace

If a tick file contains information that says that it replaces another
file, the Replace command will be executed if this flag is set.

Always use filebase

If you set this switch, CrashTick will always use the filebase even if
the area has an own directory. Read
Using the filebase
for more
information about the filbase.

Check Seen-By:s

If this switch is turned on, CrashTick will check the Seen-By lines in
the tickfiles and never send a file to a node that already is in the
Seen-By:s. It is a good idea to turn this on since it can avoid dupes.

Unattened

If you set this switch, CrashTick will suppress requesters like
"Disk full" and "Insert disk xyz".

Set filenote

If this switch is turned on, CrashTick will set the description as the
filenote for all files that it imports.

1.19 CrashTickPrefs -- Groups

Groups are used whenever it is needed to select several areas easily, for example when you are deciding what areas a node should have access to in Raid or when you decide what files you want to announce.

Groupnames

All groups may have a name. It is a good idea to use this feature, otherwise you can easily forget what the different groups are for. Examples of good groupnames are "FidoNet" or "AmigaNet". Choose your groupnames with care since they are also showed when a downlink requests an area list from your Raid.

Nodes

In this list, all configured nodes are shown. You can toggle the access of the nodes by double-clicking on them or selecting the desired access with the gadget below the list.

These access modes are available:

No access

This means that the node isn’t allowed to connect to areas in this group at all in your Raid.

Full access

This node has full access to areas in this group in your Raid.

Outgoing files only

This node has only limited access to areas in this group in your Raid. If the node connects to an area, it will receive files in the area, but any incoming files will be rejected and put in the area for bad tick files.

1.20 CrashTickPrefs -- Aka

Aka stands for Also Known As. The akas are the different node and point numbers of your system. Here you should configure all your addresses.

Removing akas

Since Akas are linked to areas and announce statements, they can’t be without changing areas and announce statements. If you try to remove an aka that is used, CrashTickPrefs will ask you if you want to change the aka in the areas and announce statements of if you want to delete them.

1.21 CrashTickPrefs -- Nodes

Here you must configure all nodes you want to send files to or ↩
receive

files from. These are the different settings:

Tick password

This is the password that CrashTick will write to the tick files it sends to this node. Incoming files must also have this password or they will be moved to the directory for bad tick files. If you specifify an empty password here, the password on incoming files will not be checked at all.

Send priority

Here you can decide which priority files to this node should have. Choose between Normal, Hold, Crash and Direct...

Pack mode
Packer

CrashTick can pack tick files and files for a node. Here you can decide if CrashTick should pack files and which packer it should use. Read the section about
Packing tick files
for more information.

Tiny Seen-By

When this switch is turned on, the Seen-By in tick files sent to this node will only contain your address and the address of the destination node. This can make the tick files you send to your downlinks shorter, but it is not a good idea to use it when sending files to other nodes...

Auto-add

If this switch is turned on, CrashTick will auto-add unknown areas to your configuration automatically. If Auto-add is turned off, the areas will still be added but with the "Unconfirmed" flag set. Areas with the "Unconfirmed" flag set will not be used until you remove the flag. When you enter CrashTickPrefs and have areas with the "Unconfirmed" flag set, CrashTickPrefs will warn you about it.

Passive

This is set if the node is passive. When a node is passive, no files at all will be sent to it. Downlinks can set this flag in Raid with %PAUSE %PAUSE if they don’t want to receive echomail for a while but want to keep the areas connected. %RESUME in Raid turns off the Passive flag.

Groups

In this list, all groups are shown. You can toggle the access of the node by double-clicking on the group or by selecting the desired access with the gadget below the list.

These access modes are available:

No access

This means that the node isn’t allowed to connect to areas in this group at all in your Raid.

Full access

This node has full access to areas in this group in your Raid.

Outgoing files only

This node has only limited access to areas in this group in your Raid. If the node connects to an area, it will receive files in the area, but any incoming files will be rejected and put in the area for bad tick files.

Areas

This is a list of areas that the node is connected to. Files from a node are only accepted if the node is connected to the area with the correct access mode. These access modes are available:

Full access

This node has full access this areas.

Outgoing files only

This node has only limited access to the area. The node will receive files in the area, but any incoming files will be rejected and put in the area for bad tick files. You can set this for your downlinks if you don’t want to allow them to hatch files.

Incoming files only

This node has only limited access to the area. Any incoming files will be accepted, but CrashTick will never send files to this node. You can set this for your uplinks if you don’t want to send files to them accidentally.

When adding areas to the list, you can decide if you want to be able to choose from all available areas or only from the areas that the node would have had access to in Raid with a cycle gadget.

Sysop name

The sysop of the node or point. This is only used when sending area notifications or configuration info to the node.

Raid password

This is the Raid password of the node. A Raid password is required to turn on and off areas via your Raid.

Default group

This is the group that any auto-added areas from this node will be put in.

```
Notify

   When a "NOTIFY ALL" or "SENDINFO ALL" command is sent to the Arexx
   port or passed on command-line, information/notify messages will
   only be sent to nodes with this flag set.

Menus
=====

Duplicate...

   This will make a copy of the current node. This can be very useful
   when adding many nodes.

Add mandatory areas...

   This will add all areas to the node that

     a) Have the Mandatory flag set in Areas
     ß) Are in a group the node has access to
     3) Are not already connected to the node
```

1.22 CrashTickPrefs -- Areas

```
                This is one of the most important sections in CrashTickPrefs.  ←
                   Here you
must configure all your areas. The most convenient way of adding new
areas is to just let them be auto-added and then edit them later.
All unknown areas will be added to the area list.

Sorting areas

   Press "Sort areas..." to sort the areas in your config. You can sort
   the areas in two ways, on the Tagname and on the Group.

Unconfirmed

   If Auto-add is turned off, unknown areas are added to the configuration
   with this flag set and will not be used until it is turned off. If you
   try to load a configuration file will unconfirmed areas, CrashTickPrefs
   will inform you and encourage you to edit them. You can easily find the
   unconfirmed areas using the "Find unconfirmed" menu item.

Aka

   This is your address for this area.

Path

   This is the directory where incoming files in this area should be
   stored. Several areas may share the same directory. If you are not
   interested in the files in an area yourself and just pass them on
   to your downlinks, you can set an empty directory and CrashTick will
   use the filebase instead. Read
```

Using a filebase
for more information.

Area command

You can use this to execute a special command everytime a file arrives in this area. As opposed to the Import command which is intended to import files to your BBS, this is intended for special tasks such as updating your nodelist every time you get a nodediff. You can of course üse the
%-codes
here as well.

If you don’t want to execute a special command, leave this string empty.

Ignore Seen-By

If this is turned, CrashTick will not check the Seen-By lines in this area. This should normallt be turned off...

No replace

If you don’t want to allow new files to replace old files in this area, turn this on and the Replace command will never be executed for files in this area.

Nodes

This list contains the nodes that are connected to this area. Files from a node are only accepted if the node is connected to the area with the correct access mode. These access modes are available:

Full access

This node has full access this areas.

Outgoing files only

This node has only limited access to the area. The node will receive files in the area, but any incoming files will be rejected and put in the area for bad tick files. You can set this for your downlinks if you don’t want to allow them to hatch files.

Incoming files only

This node has only limited access to the area. Any incoming files will be accepted, but CrashTick will never send files to this node. You can set this for your uplinks if you don’t want to send files to them accidentally.

When adding nodes to the list, you can decide if you want to be able to choose from all available nodes or only from the nodes who would have had access to this area in Raid with a cycle-gadget.

Mandatory

Nodes are not allowed to disconnect from this area using Raid.

```
Default outgoing only

   Any nodes that connect to this area will be added with the accessmode
   "Outgoing files only".

Description

   This description for this area. This will be displayed in area lists
   generated by your Raid can can be used when announcing files.

Group

   The group of the area. This is used in Raid to determine what
   areas a node/point may subscribe to. It is also used in Announce
   to let you select what files you want to announce in an area.

 Menu items
 ==========

 Duplicate...

   This will make a copy of the current area. This can be very useful
   when manually adding many areas.

 Remove all in one group...

   With this menu item you can remove all areas in a group.

 Find

   If you have many areas, it can be hard to find a specific area and
   then you can use this to find it. You only have to enter a substring,
   "iff" would find "NODEDIFF".

 Find next

   Search for the next occurrence of the search string.

 Find unconfirmed

   Searches for the next unconfirmed area.
```

1.23 CrashTickPrefs -- Announce

```
Announce
========
CrashTick is able to write announcement messages in echomail areas when you
have received files. You can announce some groups in one area and some other
groups in another area. It is also possible to announce a group in several
areas. You can also decide what the announcement message should look like
in this section.

Test
```

```
   Press this button to see what an announcement message with the current
   settings would look like.

Groups

   This announce statement will announce files that came in an area that
   belongs to one of this groups. Click on the arrow next to the string
   to edit the groups.

Aka

   Use this address as sender of the announcement message.

Format
Area header
Area footer

   These three strings determine what the announcement message will look
   like. When files are to be announced, they are first sorted by the
   area name. After that, these three steps will be executed for all
   areas you have received files in:

   1. The Area header will be written
   2. The Format string will be written once for each file you have
      received in the area.
   3. The Area footer will be written

   There are several %-codes than can be used in these three strings so
   that the strings can contain information about the file. See below
   for %-codes and how they can be formatted:

Header

   Here you can specify a textfile that will be included first in the
   announcement message. Just enter an empty string if you don’t want
   to include a textfile.

Footer

   Here you can specify a textfile that will be included last in the
   announcement message. Just enter an empty string if you don’t want
   to include a textfile.

Subject

   The subject for the announcement messages

Origin

   The origin string for the announcement messages

Menus
=====

Duplicate...
```

```
   This will make a copy of the current announce statement.

Presets

   In this menu, you will find some presets for the strings in Format, Area
   header and Area footer. Create a new announce statement and see what they
   look like using the Test button.

%-codes and formatting
======================
These codes can be used in the Format string:

  %a       Area name
  %n       File name
  %p       Directory where the file is placed
  %s       Full path to the file (use this instead of %p/%n)
  %b       File size in bytes
  %o       Originating node, the node who hatched the file
  %f       From node, the node you got the file from
  %m       Magic filename (if one is specified in the .tic file)
  %d       File description

  \n       New line

These codes can be used in the Area header/footer strings:

  %a       The name of the area
  %d       The description of the area

  \n       New line

The %-codes can be formatted in a way similar to the formatting in the
printf()-command in the programming language 'C'. This is probably best
described with a few examples:

 %20s      Make the string 20 characters wide and write it justified to
           the right. If the string is longer than 20 characters, it
           will not be truncated.

 %-20s     Make the string 20 characters wide and write it justified to
           the left. If the string is longer than 20 characters, it will
           not be truncated.

 %.20s     Truncate the string after 20 characters

They can also be combined to form things like this:

 %-20.20s  Make the string 20 characters wide and write it justified to
           the left. The string will be truncated if it is longer than
           20 characters.

Have a look at the preset strings for more examples...
```

1.24 Packing tick files

```
Packed tick files
=================
CrashTick can unpack files packed with Zip or Lha (named .zic and .lic) you
have configured commands for Lha and Zip extraction.

You can also pack tick files and also the files yourself if you want to. If
you have selected "Pack ticks", the tick files will be packed with the
selected packer and the files will be sent unpacked. When "Pack all" is set,
the files will also be send packed. Packing files has in my opinion a rather
big drawback: If you want to send a file to 10 nodes, it will be packed to
10 different archives. If you send files unpacked, you only have to have
the file one time instead of ten times.

My recommendation is that you don't pack files. Pack tick files if you like.
They are generally small and sending all tick files in one archive will save
a bit of transmission time.
```

1.25 Using a filebase

```
Using a filebase
================
If you don't want the files you receive in an area for yourself and just
want to send them to your downlinks, you can put them in a filebase. Then
your mailer will send them from there, and when all downlinks have received
the file, it will be removed the next time you do a "CrashTick maint".
Files in areas where no directory is configured will always be placed in the
filebase. If you always want the files to be copied to the filebase and sent
from there, use the "Always use filebase" switch.

When you want to remove old files from your filebase, use the command
"CrashTick maint". Then files that already have been sent to all nodes
and files that have been lying there longer than the configured maximum
time will be deleted.
```

1.26 Operating modes and commandline switches

```
Operating modes and command-line arguments
==========================================
The command-line arguments are best described by describing CrashTick's
different operating modes and the options that affect them.

CrashTick's command template:

TICK/S,HATCH/K,MAINT/S,NOTIFY/K,SENDINFO/K,RAID/K,SETTINGS/K,WAIT/S,
QUIT/S,AREA/K,DESCRIPTION=DESC/K,REPLACE/K

Miscellaneous options
---------------------
SETTINGS <filename>

This can always be used to specify a settings file that CrashTick
```

```
should use. Normally, CrashTick looks for PROGDIR:CrashTick.prefs,
and if that file isn’t found, it checks MAIL: too.

WAIT

If you use this option, CrashTick will remain loaded in the memory
and wait for any ARexx commands.

QUIT

Quits CrashTick if a copy is running in WAIT-mode

TICK
----
If you use this mode, CrashTick will process all files found in your
inbound directory. They will be placed in the correct areas and sent
to your downlinks if you have any.

HATCH <filename>
----------------
Use this command if you want to send out a new file in a file echo.
These options are associated with HATCH:

AREA <areaname>        The area the file should be hatched in
DESC <desc>            File description (optional)
REPLACE <file>         Old file that this file replaces (optional)

If you don’t specify an area, CrashTick will ask you about area,
description and replace in the Shell window. The file will then be
moved to the correct area and sent to all downlinks if you have any.

MAINT
-----
This file scans your filebase and deletes all files that have been sent
to all downlinks or have been lying there longer than the maximum allowed
time. It will only delete files that have the filenote set to "Filebase"
to make sure that no files are deleted accidentally. CrashTick sets that
filenote for all files that are written to the filebase.

NOTIFY <node|ALL>
-----------------
This command forces the Raid to send a list of connected areas to the
specified node. If you use "ALL" instead of a node, a list will be sent
to all nodes that have the "Notify" flag set.

SENDINFO <node|ALL>
-------------------
This command forces the Raid to send a dump of the configuration for
the node to the specified node. If you use "ALL" instead of a node,the
configuration will be sent to all nodes that have the "Notify" flag set.

RAID <filename>
---------------
Lets CrashTick’s built-in Raid execute the Raid commands in the specified
message. The message must be stored in RFC-format. CrashMail and TrapToss
can generate messages in RFC-format, and probably other tossers too...
```

1.27 ARexx port

```
ARexx port
==========
Actually, there isn’t much point in having an Arexx port in CrashTick.
CrashTick is run that often and doesn’t take very long to load. The main
reason why CrashTick has an ARexx port is that it uses it internally to
make sure that two copies of CrashTick aren’t running at the same time.
If you start a second copy of CrashTick while another copy is already
running, the commands will just be passed on to the first copy...

If you want to use the ARexx port yourself, start CrashTick with
"CrashTick wait". CrashTick will then remain in memory and accept
commands sent to the ARexx port "CRASHTICK".

The commands are basically the same as the commands that you can give to
CrashTick on command-line, so please look there for better descriptions
of them. These are the commands that CrashTick understands:

QUIT

TICK

MAINT

RAID <RFC filename>

NOTIFY <node|ALL>

SENDINFO <node|ALL>

HATCH <filename> [AREA <areaname>] [DESC <description>] [REPLACE <filename>]
(Description and replace are optional...)
```

1.28 CrashTickStats

```
CrashTickStats
==============
As you might know, CrashTick keeps statistics on all areas and nodes in your
settings file. These statistics are shown when a downlink requests a list of
your areas, but can also be shown using this program.

If you want to read the stats in an own program, have a look at the file
TickStatsFormat.doc for a description of the file format.

The syntax for CrashTickStats is the following:

CrashTickStats FILE/A,SORT,LAST7/S,NOAREAS/S,NONODES/S,GROUP

FILE      The statistics file CrashTickStats should read. This is usually
          called CrashTick.stats and can be found in the directory where
          you keep CrashTick.

SORT      How the output of the areas should be sorted. The node output is not
```

```
          affected by this. You can choose between the following methods:

          A -- Alpha      Sort the list alphabetically (default)

          F -- Files      Sort on the total number of files that you have
                          have received in the area

          B -- Bytes      Sort on the total number of bytes that you have
                          have received in the area

          W -- Week       Sort on the number of bytes you have received
                          in the last week

          D -- Date       Sort the areas after the date when the first
                          file arrived in them

          L -- Last Date  Sort the areas after the date when you last got
                          any files in them

 NOAREAS  Don’t show any statistics for the areas

 NONODES  Don’t show any statistics for the nodes

 LAST7    Show the statistics for the last 7 days (and the files that have
          arrived today so far)

 GROUP    Only areas in one of the groups specified here will be shown. Please
          note that you can specify multiple groups like "ABCEF" etc.
```

1.29 Acknowledgements

I want to thank the following persons:

o Johannes Nilsson, Fredrik Bennison, Mathias Axelsson and Rickard Olsson
  for beta-testing and suggestions.

o Olaf Barthel for gtlayout.library. Without it, CrashTickPrefs would
  have been much harder to make if I had made it at all. :-)

o Nico François and Magnus Holmgren for reqtools.library. CrashTick
  uses reqtools.library for its string and file requesters.

  ReqTools is Copyright (c) Nico François and Magnus Holmgren

o Johan Billing for his really great tosser CrashMail. :-)

1.30 How to contact the author

```
How to contact the author
===========================
FidoNet:   Johan Billing@2:200/108.7
InterNet:  johan.billing@kcc.ct.se
```

```
SnailMail: Johan Billing
           Östra Storgatan 22
           S-260 60 Kvidinge
           SWEDEN

Please don't hesitate to send bugreports and suggestions. Redirect any
flames to your nearest trashcan.

There are a few things I'd really like if you sent me:

o Translations of the program. Just modify the included translation file
  "catalogs/empty.ct" and mail it to me!

o Descriptions of how to get CrashTick to work with your BBS program and
  any scripts that are needed.

o A better icon for CrashTickPrefs. :-)
```

1.31 Program History

```
History
=======

1.0  - First release
```