Maintenance Documentation

ihost Maintenance

ihost is a Perl-based host for use with the i-scream Distributed Central Monitoring System. This document provides an overview of how *ihost* works and how you may alter it to suit your requirements.

Revision History

11/03/01	Initial creation				
		Committed by:	pjm2	Verified by:	tdb1
				Date:	24/03/01
		Committed by:		Verified by:	
				Date:	
		Committed by:		Verified by:	
				Date:	
		Committed by:		Verified by:	
				Date:	
		Committed by:		Verified by:	
				Date:	

Introduction

ihost is a Perl-based host for use with the i-scream Distributed Central Monitoring System. It is designed to be easy to alter so that those with even a basic knowledge of Perl (or, indeed, any similar language) should be able to change it to suit any specific requirements.

Overview of what the ihost does

The ihost, like any i-scream host, is an application that harvests data from the machine it is running on. It then sends this data to a central server for processing.

Supported Platforms

The ihost performs on Solaris, Linux and FreeBSD. As it is written in Perl, it will require Perl to be installed to be able to run. It also requires the Perl Socket library, which is standard on most Perl installations.

Using ihost

ihost is very simple to use. Please check the separate usage documentation for more details.

Understanding what ihost does

ihost is written in Perl and those who know Perl should be able to understand how and what it does simply by looking at the source. However, for those who are not so sure, here is an overview of the basic architecture of the program.

Separate data collection

The ihost in its entirety consists of two main Perl programs. The host itself (i.e. the file you would execute to run the host) is called "ihost.pl". ihost.pl deals with all of the networking, configuration and obtaining some system information. However, the majority of the system information is obtained by running a separate program called "statgrab.pl". Running this provides a set of name and value pairs, each on one line and separated by the first space character on the line. Please read the separate documentation about statgrab.pl if you wish to know more about it. As far as ihost.pl is concerned, all it needs to know is that it can execute statgrab.pl and read the information it provides on the standard input.

Statgrab.pl data format

statgrab.pl was intentionally kept separate from the ihost application, as this allows it to provide data to other applications. Each individual item of data that comes from the statgrab output takes up a single line, with a linefeed character at the end. Every character from the start of the line up until (and not including) the first space character determines the preferred location of the data in an XML packet. Every character after this first space and before the linefeed character is regarded as being the actual data value.

For example, if statgrab is written to provide information about a computer's I.P. address, one of the lines read from the standard input would look like: -

```
packet.attributes.ip 129.12.4.8\n
```

This would mean that the I.P. address of the computer is 129.12.4.8 and this data value should ideally be presented as an attribute of the root packet tag. More on that later…

What does ihost do with statgrab's output?

When the host runs statgrab.pl, each line of the output is read into an array. This array is then used to populate a hash called %packet. The key of the hash is the name of the data value (e.g, "`packet.attributes.ip`") and the value of the hash is the data value (e.g. "`129.12.4.8`"). Note that all data obtained from statgrab.pl is considered to be a string. The ihost can then refer directly to any of the data contained within the hash.

Basic operation of ihost

The ihost is split into 4 distinct areas: -

1. The main program and initial startup code.
2. A subroutine to handle TCP configuration with an i-scream Filter Manager.
3. A subroutine to handle TCP heartbeats with an i-scream Filter.
4. A subroutine to handle formatting and sending a single UDP packet to an i-scream Filter.

Startup

When the user executes ihost, it declares some global values for the program. These are never accessed concurrently, so beware of introducing any concurrency issues if you alter the code of the ihost to suit your own requirements.

We recommend that `use strict` is used in the ihost. This reduces the possibility of errors being introduced into the program, as all variables will need to be declared before use.

The ihost requests to use the `IO::Socket` and `Sys::Hostname` libraries. These are used to perform the networking and hostname parts of the program. These are usually part of a typical Perl installation, but please check that you have them if you seem to be having problems that involve error messages that mention these modules.

On startup, the program checks that the necessary command line parameters have been provided. The program requires two parameters: -

1. The machine name of an i-scream filter manager (e.g. `raptor.ukc.ac.uk`)
2. The port number of the filter manager (e.g. `4567`)

If either of the above parameters is missing, then the ihost stops after reporting the correct usage of the program.

The ihost keeps a record of how many packets it has sent. This number is initialised to 1.

After this startup process has been completed, a subroutine (`&tcp_configure()`) is called to configure the ihost with the filter manager specified on the command line.

Configuration

All configuration is carried out by a call to the `&tcp_configure()` subroutine. This establishes a TCP connection to the specified i-scream filter manager and requests configuration information.

If altering this code, please remember that the ihost must conform to the configuration protocol specification at: -
http://www.i-scream.org.uk/cgi-bin/docs.cgi?doc=specification/protocols.txt

If the program cannot connect to the filter manager, it displays an appropriate error message and waits for a defined period before trying again.

Upon successful connection, the ihost starts the configuration process by sending the "`STARTCONFIG`" command. A linefeed character follows each command to and from the server. This makes Perl ideal for this task as a simple file handle may be used to read the server's response. Please check the above URL for the full configuration protocol specification.
During the configuration process, the ihost stores the time at which the server's configuration was last modified and the files used for this configuration. The ihost does not use this

information itself (other than echoing it to screen for information purposes), but it must retain this information for future heartbeats.

The server also informs the ihost of its [the host's] fully qualified domain name. This may be used in UDP packets when the ihost must send the machine name of itself. It may, of course, derive this information itself, but on some systems this has proved quite tricky.

The configuration process also tells the ihost what time intervals it should use for sending UDP packets and performing TCP heartbeats. Typically, the `UDPUpdateTime` is less than the `TCPUpdateTime`. These times are in seconds.

The server provides the ihost with details of a filter address to use for subsequent TCP heartbeats. When the configuration has ended, the ihost will only contact the filter manager again if it encounters an error during a heartbeat with this filter.

TCP Heartbeats

Heartbeats are used as a reliable means of contacting a filter. These are used in addition to the UDP packets, as UDP packets may get lost on a busy or unreliable network. It acts as an "I'm alive" delivery mechanism to the filter. These should occur according to the time interval set by the filter manager for the host machine.

The ihost starts the process by sending a line to the filter containing "HEARTBEAT".

The rest of the heartbeat is carried out as per the specification at: -
http://www.i-scream.org.uk/cgi-bin/docs.cgi?doc=specification/protocols.txt

Note that during a heartbeat, the ihost is expected to send back the configuration file list and last modified times that were received from the filter manager.

The ihost silently ignores most errors encountered during this process, except for the `ERROR` response regarding the last modified time. Such a response indicates that the server configuration has been modified, and hence, the ihost reconfigures itself with the original filter manager address. It is, however, important that the ihost completes the heartbeat before reconfiguring, as not doing so causes unwarranted alerts.

If you run the ihost in a terminal window, then it will echo a "^" character to the screen each time it successfully delivers a heartbeat to a filter.

Data formatting for UDP packet sending

The ihost periodically sends UDP packets to a filter. These packets contain information about a machine and are sent according to the `UDPUpdateTime` interval set by the filter manager for the host machine. UDP is not a reliable delivery mechanism, but it is used to cut down on the amount of network traffic as the `UDPUpdateTime` is typically less than the `TCPUpdateTime`.

The `send_udp_packet()` subroutine is used to deal with collating the data for the packet and sending it to the filter address.

The subroutine begins by executing statgrab.pl and storing the output from it in a hash. This hash is used to populate the packet with data.

Some of the packet's data contents are generated by ihost itself. These include the current date in seconds since the epoch; the machine's I.P. address; the sequence number for the packet and the fully qualified domain name (typically obtained from the filter manager).

Note that XML is used to send information to the filter in UDP packets. It is important that valid XML is sent, otherwise it will be rejected by the filter. To ensure that the packet contents are valid, this subroutine lets you build the XML in a logical, nested fashion, into which you may insert any variables that you want. Please note that a filter will reject UDP packets that are larger than 8kb in size.

An unmodified ihost application will build the packet contents in the following manner: -

```
    my($xml) = <<EOF;

<packet seq_no="$seq_no" machine_name="$fqdn" date="$date" . . .
```

This allows the contents of the packet to span more than one line, hence resulting in the possibility of nicely laid-out contents, including indenting. Variables are easily included in the XML contents by adding them with their $ prefix.

To reduce the overall size of the packet when it is sent, all leading spaces and all linefeed characters are removed from the contents of the XML string, i.e for those who understand Perl: -

```
      $xml =~ s/\n\s*//g;
```

The packet number is incremented each time a UDP packet is successfully sent. If you are running the ihost in a terminal window, then you will also see a "-" character echoed to the screen. This makes a typical running host appear like such when running: -

```
      ------^------^------^------^------^------^--- etc
```

More about XML packet contents

The data that the ihost sends in UDP packets must be formatted as valid XML. This is so that third parties may come along and make their own host applications with ease. Not only that, but it also means that any host many send any data it wants to. This allows the entire i-scream system (hosts, server, clients) to be flexible in the way that it works. The XML via UDP protocol is very simple and the text-based format makes it usable in so many ways.

The Perl ihost, as with all other i-scream hosts, should provide what is regarded to be the minimum essential data in the XML. This includes the machine's I.P. address, packet sequence number, date, machine name and packet type.

There are also recommendations on what other data the ihost should send. But on top of this, the ihost may send any other data that it wants to.

A more detailed description of the essential and recommended data may be found at the following URL: -

http://www.i-scream.org.uk/cgi-bin/docs.cgi?doc=specification/expected_data.txt

The above document also lists a full example layout of a typical XML packet.