

IHSH Internals¹

Arne Helme
Stelvio
the Netherlands

Draft of October 22, 2003

¹Development of IHSH has been sponsored by SURFnet BV, the Netherlands

Contents

1	Overview	1
2	IHSH Structure	1
3	IHSH Internals	1
3.1	Main Program	1
3.2	Configuration File Support	1
3.3	IHSH Command Support	2
3.4	Misc. Support	3
3.5	ARS Module Extension	3
3.5.1	ARS Module Interface	3
3.5.2	ARS IODEF Functions	4
3.5.3	Support Functions	4
A	ARS Field IDs	5
A.1	IODEF-Document	5
A.2	IODEF-Documen-Address	6

1 Overview

This document describes the internals of the IHSH program. The document is intended for those whoe, for some reason, have to modify the source code to extend the program or implement improvements.

2 IHSH Structure

The IHSH program source code consists of the following components:

Main program: Contains the IHSH configuration management and main loop of the program.

Configuration file support: Contains functions to operate on IHSH configuration files.

IHSH command support: Contains functions that implement the IHSH command interpreter and functions to operate on the internal XML tree structures.

Misc. support: Contains miscellaneous functions to work with different kinds of input.

ARS module extension: Contains functions to communicate with an external ARS/Remedy server.

XML input/output extension: Contains functions to read and write XML formatted documents.

In the rest of this document each of the components is described in more detail with emphasis on how to modify it.

3 IHSH Internals

3.1 Main Program

The IHSH main program is located in the file `ihsh.c`, and contains code to configure and run the IHSH command interpreter and extension modules.

3.2 Configuration File Support

The IHSH configuration file support functions are located in the file `ihsh_cfg.c`. The module exports the following functions:

- `int ihsh_cfg_readfile(char *fname)`
- `char *ihsh_cfg_match_str(char *name)`
- `int ihsh_cfg_match_int(char *name, int *num_p)`
- `void ihsh_cfg_fdump(FILE *fp)`

The function *ihsh_cfg_readfile()* reads the file with the name *fname* into IHSH configuration table. Parse errors and file operation errors are reported back to the caller.

The function *ihsh_cfg_match_str()* return the string value attribute associated with the config name *name* back to the caller. If the variable is not found in the configuration table the value NULL is returned.

The function *ihsh_cfg_match_int()* sets **num_p* to the numerical value associated with the config name *name*. If the variable is not found in the configuration table or the value set is not an integer the value -1 is returned.

The function *ihsh_cfg_fdump()* dumps the contents of the IHSH configuration to the output stream associated with the file pointer *fp*.

Below follows a piece of source code that illustrates the use of the configuration file functions to read variables out of the configuration table:

```
/* Fetch ars_server entry from configuration file */
cfgstr = ihsh_cfg_match_str("ars_server");
if (cfgstr == NULL) {
    ARS_DEBUG("'ars_server' not set in IHSH config file");
    return -1;
}
ars_server = cfgstr;

/* Fetch ars_port entry from IHSH configuration file */
errno = 0;
if (ihsh_cfg_match_int("ars_port", &ars_port) < 0) {
    ARS_DEBUG("'ars_port' not set in IHSH config file (or ERANGE)");
    return -1;
}
if (ars_port < 0 || ars_port > 65535) {
    ARS_DEBUG("'ars_port' must be 0 <= x < 65535");
    return -1;
}
```

Note that in the case of integers, it is the caller's responsibility to check whether the integer value is sane for a given application.

3.3 IHSH Command Support

The IHSH command support functions are located in the file *ihsh_cmds.c*. The module exports the following functions:

- `int ihsh_cmd(int cmd, char *arg, FILE *fp)`
- `char *ihsh_parse(char *buf, int *cmd)`

and the enumerate type `enum cmd`.

The function *ihsh_cmd()* runs the command *cmd* with the command line argument *arg* with eventual output written to stream *fp*.

The function *ihsh_parse()* parses the buffer *buf* and determines the IHSH command type, and sets **cmd* to the determined command type.

Extending the command set of IHSH with new commands is relatively straightforward. For each new command, a new values needs to be added to the enumerated type *enum cmd*, and to the command table *cmdtab*. In the **switch** statement of the function *ihsh_parse()*, an entry must be made for the new value including the code required to implement the new command. The new command can be implemented in-line, or as a new function local to the module.

3.4 Misc. Support

The IHSH miscellaneous functions are located in the file *ihsh_misc.c*. The module exports the following functions:

- `char *ihsh_getline(FILE *fp)`
- `*ihsh_getlines(FILE *fp)`

The function *ihsh_getline()* reads a line including newline character from stream *fp*. The caller is responsible for freeing the allocated memory.

The function *ihsh_getlines()* reads multiple lines including newline character from stream *fp*. The caller is responsible for freeing the allocated memory. Multi-line input is ended by a single dot (.) on an otherwise empty line of input.

3.5 ARS Module Extension

The IHSH ARS/Remedy module extension is divided into three sub-components: the IHSH ARS module interface, the ARS IODEF functions, and a set of generic support functions.

3.5.1 ARS Module Interface

The ARS module interface functions are located in the file *ihsh_ars.c*. The module exports the following functions:

- `int ihsh_ars_init(void)`
- `int ihsh_ars_getentry(ih_tree_t *tree, char *entryid)`
- `int ihsh_ars_setentry(ih_tree_t *tree)`

The function *ihsh_ars_init()* initializes the ARS module and reads required ARS configuration variables from the internal IHSH configuration table, and is typically invoked only once from the main IHSH program. In order for the module to configure successfully, the following configuration variables are required:

ars_server: DNS name of the machine where the ARS server is running.

ars_port: TCP port that the ARS server is listening to.

ars_user: ARS user name that IHSH shall use when establishing a session with the ARS server.

ars_passwd: Password to use together with the user name.

ars_debug: Toggles the ARS debug output. In the current version of IHSH, ARS debug information is written to `stderr`. The debug variable is the only variable that is optional.

The function *ihsh_ars_getentry()* establishes a session with an ARS server and fetches the entry with ARS Entry ID *entryid* and builds an internal XML tree representation of the received data.

The function *ihsh_ars_setentry()* establishes a session with the ARS server and updates it with the contents of the internal XML tree representation of an IODEF document.

3.5.2 ARS IODEF Functions

The ARS IODEF functions are located in the file `ihsh_ars_iodef.c`. The module exports the following functions:

- `int ihsh_ars_getentry_iodef(ARControlStruct *control, ih_tree_t *tree, char *entryid)`
- `int ihsh_ars_setentry_iodef(ARControlStruct *control, ih_tree_t *tree)`

The function *ihsh_ars_getentry_iodef()* invokes and ARS API *ARGetEntry()* to fetch an entry from the ARS server, performs the actual conversion between the ARS API representation of entry information and the IHSH internal representation of IODEF documents. The function assumes that a session referred to by *control* has already been established with the server.

The function *ihsh_ars_setentry_iodef()* performs the actual conversion between the IHSH internal representation of IODEF documents and the ARS API representation of entry information, and invokes an ARS API *ARSetEntry()* call to update the server. The function assumes that a session referred to by *control* has already been established with the server.

3.5.3 Support Functions

The ARS support functions are located in the file `ihsh_ars_misc.c`. The module exports the following functions:

- `void ARS_DEBUG(char *, ...)`
- `void PrintARFieldValueList(FILE *fp, ARFieldValueList *value)`
- `void PrintARStatusList(FILE *fp, ARStatusList *fp)`

together with the integer variable `int ars_debug`.

The function `ARS_DEBUG()` writes debug information from the ARS module to `stderr`. The syntax of the parameters to the function is the same as for the `printf(3)` family of formatted output functions in the standard C library.

The function `PrintARFieldValueList()` is a modified version of a similar function in the driver program delivered with the ARS API. The function writes a formatted ARS Field Value List to the stream point to by `fp`.

The function `PrintARStatusList()` is a modified version of a similar function in the driver program delivered with the ARS API. The function writes a formatted ARS Status List to the stream point to by `fp`.

A ARS Field IDs

A.1 IODEF-Document

```
#
# File exported Wed Sep 24 08:03:47 2003
#
#AR-Message-Begin                Do Not Delete This Line
Schema: CERT: IODEF-Document
Server: weetmuts.surfnet.nl
Login:
Password:
Action: Submit
# Values: Submit, Query
Format: Short
# Values: Short, Full

        purpose !536870946!: handling
# Values: handling, statistics, warning, other
        restriction !536870947!: need-to-know
# Values: public, need-to-know, private, default
        IODEF-Message type !536870948!:
# Values: incoming, outgoing
        Status !          7!: New
# Values: New, Processed, Archived
        IncidentID !          8!: VOID
        AlternativeIDs !536870945!:
        Description !536870949!:
        Submitter !          2!: $USER$
        Assignee !          4!: auto-submitter
        Impact.completion !536870950!: succeeded
# Values: failed, succeeded
        Expectation.priority !536870955!:
# Values: low, medium, high
```

```

Expectation.Description !536870954!:
        Impact.type !536870953!:
# Values: admin, dos, file, recon, user, none,
#   unknown
        History !           10!:
        Impact !536870951!:
        Method.Description !536870952!:
        Type !536870933!: Organization
# Values: Organization, Person
        Role !536870943!: irt
# Values: Creator of IODEF doc, admin, tech,
#   irt, cc
Record.RecordData.Item !536870970!:
        name !536870917!: CERT-NL
        Email !536870915!: cert-nl@surfnet.nl
        Telephone !536870925!: +31 302305305
        Timezone !536870942!: +02:00
#AR-Message-End                        Do Not Delete This Line

```

A.2 IODEF-Documen-Address

```

#
# File exported Wed Sep 17 17:21:56 2003
#
#AR-Message-Begin                        Do Not Delete This Line
Schema: CERT: IODEF-Document-Address
Server: weetmuts.surfnet.nl
Login:
Password:
Action: Submit
# Values: Submit, Query
Format: Short
# Values: Short, Full

RequestID !1!:
System.category !536870914!:
# Values: source, target, intermediate
Address.category !536870915!:
# Values: unknown, atm, e-mail, lotus-notes,
#   mac, sna, vm, ipv4-addr, ipv4-addr-hex,
#   ipv4-net, ipv4-net-mask, ipv6-addr, ipv6-addr-hex,
#   ipv6-net, ipv6-net-mask
Address !536870913!:
Submitter !           2!: $USER$
IODEF-Document ID !536870916!:
Short Description !           8!: a
Status !           7!: New

```


Values: New, Assigned, Fixed, Rejected,

Closed

#AR-Message-End

Do Not Delete This Line