The EFSD Programming Manual

Christian Kreibich

cK@whoop.org

The EFSD Programming Manual
by Christian Kreibich

Table of Contents

Chapter 1. Introduction

Chapter 2. Overview of EFSD

EFSD Architecture Overview

Using EFSD in Your Application

Chapter 3. `libefsd` - The Client Side of EFSD

Using `libefsd`

The way applications talk to `efsd` is through `libefsd`. `libefsd` handles all the dirty low-level work (r23F as the I/O details of sending commands and receiving replies) and provides a convenient API to the clients. The following sections will briefly introduce general concepts before the API is explained in detail.

Opening a Connection to `efsd`

Before you can laun3F any commands, you need to establish a connection to `efsd` via

Description

Returns pointer to a newly allocated and initialized Efsd connection object. You need this object for all other calls in order to identify the connection to libefsd.

efsd_close

Name

efsd_close — Closes a connection to Efsd.

Synopsis

```
int efsd_close (EfsdConnection *  ec);
```

Arguments

ec

The Efsd connection

Description

Use this to close an efsd connection. Frees the allocated EfsdConnection object. Returns value < 0 if the the final command could not Re sent to Efsd.

efsd_get_connection_fdrder to identify the connection tog83 0 Td[88iction to

Synopsis

Arguments

ec

Description

When issuing this command, your client will receive events for all files in the directory, or, if for some reason the command is issued on a file, for the file only. The directory is not monitored afterwards, you just get the contents delivered once.

efsd_copy

Name

efsd_copy — copies a number of files to a target file.

Synopsis

efsd_copy

Arguments

ec

efsd_move

Name

`efsd_move` — moves a number of files to a target file.

Synopsis

```
EfsdCmdId efsd_move (EfsdConnection * ec, int num_files, char ** files,
EfsdOptions * ops);
```

Arguments

`ec`

The Efsd connection

`num_files`

The number of files passed

`files`

Array of strings, must contain at least *`num_files`* items.

`ops`

EfsdOptions poJ/Fer, created using either`efsd_ops` or `efsd_ops_create` and `efsd_ops_add`

Synopsis

```
EfsdCmdId efsd_remove(EfsdConnection * ec, int num_files, char ** files,
EfsdOptions * ops);
```

Arguments

ec

The Efsd connection

num_files

The number of files passed

files

Array of strings, must contain at least *num_files* items.

ops

Efsd[illegible]

`val`

A floating-point value.

Description

This command sens a floating point value as metadata on a file. The data is labeled by the given key, which must therefore be unique among all the metadata sen for a file.

efsd_sen_metadata_str

Nametio38.894efsd_sen_metadription

efsd_get_metadata

Name

efsd_get_metadata — retrieve file metadata.

Synopsis

```
EfsdCmdId
```

Arguments

ee

The received EfsdEvent.

Description

efsd_metadata_get_float

Name

Description

Convenience function that returns the filename from a metadata reply event. If the event does not contain metadata, the function returns `NULL`, the filename otherwise. You do not need to `free` the string, it gets deallocated when the event gets cleaned up. So when you want to keep it around, you need to `strdup` it.

efsd_reply_filename

Name

`efsd_reply_filename` — returns filename contained in an event.

Synopsis

```
char * efsd_reply_filename (EfsdEvent * ee);
```

Arguments

ee

The EfsdEvent.

Description

Convenience function to access the filenames in reply or filechange events. If the event is a reply event and the contained command is an efsd_file_cmd, it returns the

efsd_start_monitor_dir

Name

efsd_start_monitor_dir — start monitoring a directory for file events.

Synopsis

```
EfsdCmdId efsd_start_monitor_dir
```

Synopsis

```
EfsdCmdId efsd_stop_monitor (EfsdConnection * ec, char * filename);
```

Arguments

ec

The Efsd connection.

filename

efsd_lstat

Name

efsd_lstat — returns the result of lstat on a file.

Synopsis

```
EfsdCmdId efsd_lstat (EfsdConnection * ec, char * filename);
```

Arguments

ec

The Efsd connection.

filename

The name of the file that is to be lstated.

efsd_readlink

Name

efsd_readlink — returns the file a symlink points to.

Synopsis

```
EfsdCmdId efsd_readlink (EfsdConnection * ec, char * filename);
```

Arguments

ec

The Efsd connection.

filename

The name of the symbolic link whose target is to be read.

. . .

variable arguments

Description

This is the solution for passing options to commands when you know at compile time what options you want to pass. Returns a pointer to a ready-made EfsdOptions object. You do NOT need to free it after you've launched the command, it is freed by

efsd_op_get_lstat

Name

efsd_op_get_lstat — requests lstat events

Synopsis

```
EfsdOption * efsd_op_get_lstat ( void);
```

Arguments

void

Description

Creates an option that causes lstat results to be sent to the client for all files seen in EFSD_FILE_EXIST events.

efsd_op_get_metadata

Name

efsd_op_get_metadata — request metadata information

Synopsis

```
EfsdOption * efsd_op_get_metadata (char * key, EfsdDatatype type);
```

Arguments

key

character string that identifies the metadata

type

datatype of

Description

Creates an option that causes metadata of certain key and type to be sent to the client, for all files reported in EFSD_FILE_EXIST events.

efsd_op_get_filetype

Arguments

void

Description

Creates an option that causes commands like `efsd_move`, `efsd_copy` and `efsd_remove` to operate like the command-line versions, when passed the “-f” option.

efsd_op_recursive

Name

`efsd_op_recursive`

Synopsis

```
EfsdOption * efsd_op_sort ( void);
```

Arguments

void

Description

This option constructor returns an EfsdOption that causes EFSD_FILE_EXISTS events to be reported in alphabetical order.

efsd_op_list_all

Name

efsd_op_list_all — include hidden files in EFSD_FILE_EXISTS events.

Synopsis

```
EfsdOption * efsd_op_list_all ( void);
```

Arguments

void

Chapter 4. efsd - The Server Side

This chapter explains the inner workings of the `efsd` daemon.:

Multithreading

To Do

Well, software is never finished, and this also applies to efsd. You can always look at the TODO file in CVS to see what's currently missing. Patches are appreciated. At the time of this writing, the following things are missing:

- A thorough testsuite, especially for the cache interaction and the functions in efsd_fs.c i.e. the file copy / move / remove code. Ideally this would be an automated set of tests to check if changes broke anything. Think XP.
- A great feature would be to not only get the filetypes in a MIME type-ish notation, but also afullyeadable form of the ou would include internationalization.

Chapter 6. Summary