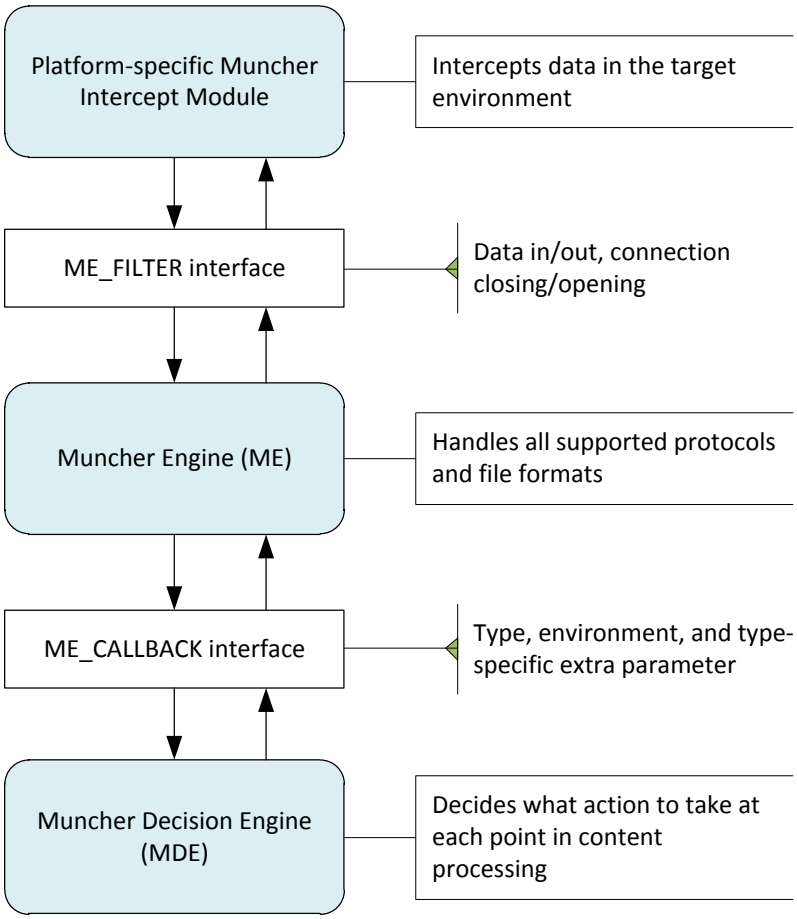
MUNCHER ENGINE OVERVIEW

CONTENTS

GENERAL LAYOUT

LAYOUT DIAGRAM



INTENDED USE

Different implementations are intended to use their own intercept module for the required kind of data intercept, and then either use our decision engine (if applying our style of filters), or implement their own decision engine if needing more customized behaviour.

MUNCHER INTERCEPT MODULES (MI)

Below are a few short examples of current and planned intercept modules.

SOCKTRAP

This intercept module can be used on Windows to intercept connections created by other applications. No proxy configuration is needed, and the system works without the need for rebooting.

PROXY

(planned)

A combined HTTP and Socks4/5 proxy service that can be deployed on a single port and act appropriately by detecting the type of request received.

MOBILE BROWSER

Because of the restrictions placed on apps in the iOS and Metro environments (and to a lesser extent, Android), this intercept method involves creating a standalone implementation of the Webkit engine and configuring it to pass data through the Muncher filtering engine.

This in effect will create a “Muncher Browser” that can avoid the limitations of these platforms and provide filtered browsing.

MUNCHER ENGINE (ME)

Below are the most important functions and structures. Please see ME.h for complete interface information.

FUNCTIONS

```
M_RESULT ME_Filter  (
                        ME_INFO * info
                );
```

Calls all internal filters for the supplied “info”.

Call info->callback with all notifications and decision requests regarding the content.

Info->type should be set to the type of the content, for example ME_TYPE_TCP for TCP content.

STRUCTURES

```
typedef struct
{
        ME_CALLBACK callback;
        void * callback_extra;
        ENV environment;
        ME_TYPE type;
        ME_FLAGS flags;
        void * extra;
        VB in;
        VB out;
        VB content;
        void * extra_parent;
}
ME_INFO;
```

EXTRA MODULES

VB (MEMORY MANAGEMENT)

The VB module allows buffers to be managed easily and passed between modules. It is the standard management system for passing data to and from the Muncher components.

Details can be found in "/Core/VBlock/VBlock.h".

ENV (ENVIRONMENT MANAGEMENT)

The ENV module allows a series of key/value associations to be saved and passed between Muncher modules. This is used to save the state of each session, and allow the callback to retrieve extra information about what is happening.

Details can be found in "FilterEngine/ENV/ENV.h".

CFG (CONFIGURATION MANAGEMENT)

Allows easy navigation of the configuration files.

Details can be found in "Core/CfgParser/CFG.h"

MRESULT (RESULT CODE MANAGEMENT)

Used for all return values.

Details can be found in "Core/MResult/MResult.h"

MUNCHER DECISION ENGINE (MDE)

This module is responsible for responding to callbacks from the filtering engine and applying Muncher-style filters to them.

FUNCTIONS

```
M_RESULT MDE_New  (
                        CFG * configuration,
                        ME_CALLBACK * nextcallback,
                        void *nextcallback_extra,
                        MDE_HANDLE * newhandle
                );
```

Initializes a decision engine with the supplied filters and default environment.

Further callbacks can be chained together using “nextcallback”.

M_RESULT MDE_Callback (M_RESULT result, ENV environment, void * extra, void * callback_extra);

Callback function passed to ME_Filter (via “info” parameter).

“result” describes the type of notification, request, warning or error being sent.

“environment” is the current environment for the session.

“extra” is a result-specific pointer that allows the callback to return decisions to the filtering engine.

“callback_extra” is the value supplied to ME_Filter (via the “info” parameter). This should be set to the MD_HANDLE value returned by MDE_New.

Any return value other than S_OK will abort ME_Filter, returning the result to the caller.

M_RESULT MDE_Free (MDE_HANDLE handle);

Frees allocated resources.

FOLDER DESCRIPTIONS

Binaries	Contains the most recently compiled binaries.
Core	Contains generic modules that are used by multiple other codebases (eg: string and memory handling).
DecisionEngine	The module responsible for replying to decision requests from the FilterEngine module. In our case, applying our filters. In other cases, end users might replace it with their own logic module.
Engine	Handles all supported protocols and file formats as passed to it by an intercept module.
InterceptModules	Contains modules for generic intercept methods that might want to be used by other apps, eg: SockTrap socket intercept module, a proxy mode, etc. For more complex implementations (eg: mobile browser), they should be implemented in other locations.
Notes	Various implementation notes.
Project	Visual studio project files and generated binaries.
TestEnvironment	Binaries for testing under the current Ad Muncher environment.
Utilities	Contains utilities needed for compilation and testing.

PACKAGING

The compilation targets for the different modules are as follows:

MUNCHERENGINE.DLL

Exports:

- ME_Filter function.
- All ENV functions.
- All VB functions.
- All MResult functions.

For other platforms, this library would be changed to a platform-appropriate library format, for example .so for OSX, or a static library for iOS.

MUNCHERDECISIONENGINE.DLL

Exports:

- MDE_Callback function and associated startup/free functions.
- All CFG functions.

For other platforms, this library would be changed to a platform-appropriate library format, for example .so for OSX, or a static library for iOS.

SOCKTRAPCHILD32.DLL AND SOCKTRAPCHILD64.DLL

Contains the 32bit and 64bit versions of the SockTrap child hook module, which can be loaded into all processes to redirect outbound connections to a local target.

The functions to install are inside these DLLs, and the DLLs themselves are what are loaded by the SetWindowsHookEx function.

SOCKTRAPPARENT.DLL

Contains the parent code that the above child DLLs will look for. This will receive connections, create outbound connections, and route content through the ME_Filter function for filtering.

USE BY THIRD PARTIES

Distribution is intended to happen with the above modules and their associated .h header files.