Crossfire - Remote Debugging Architecture for Firebug

Subtitle Text, if any

Michael G. Collins
IBM Research - Almaden
mgcollin@us.ibm.com

John J. Barton
IBM Research - Almaden
johnjbarton@johnjbarton.com

Abstract

We present Crossfire, a protocol designed to enable remote debugging with the Firebug Web debugging tool, and an implementation of this tool as a Firefox extension. We also present an architecture in which the user interface of Firebug is separated from the backend debugger into separate processes connected via the Crossfire protocol.

Categories and Subject Descriptors CR-number [*subcategory*]: third-level

General Terms term1, term2

Keywords keyword1, keyword2

1. Introduction

Web Applications continue to grow in size and complexity. The emergence of common toolkits and libraries for Javascript along with performance increases in Web Browsers fuels growth in client-side Web application development. As more functionality shifts from server- to client-side, the size of Javascript codebases naturally increases. Unfortunately for developers, the tools to develop, manage, and debug larger codebases have not followed the applications into the Web application development space.

Sophisticated development tools become crucial for understanding and working with larger codebases. Compilers, debuggers, and other tools are often combined to create an Integrated Development Environment (IDE) where most, if not all, development tasks can be performed. Web development tools, on the other hand, often reside in the Web Browser. In 2011, all of the major Web browsers ship with some kind of Web development tools included[**?**], and still more are available as plugins or add-ons. These tools operate directly on the runtime Web application that is loaded into the browser, and do not retain any knowledge of or connection to the original source code. Therefore a developer must manually apply any changes made in one of these tools to his or her source code.

The rise of Mobile and Tablet computers which ship with fully functional Web browsers means that Web application developers have additional form-factors to consider.

Tools such as "Inspectors" and Firebug's HTML Breakpoints[**?**] enable developers to quickly locate the section of code they are interested in.

[Copyright notice will appear here once 'preprint' option is removed.]

2. Protocol

2.1 Overview

The Crossfire protocol is an asynchronous, bi-directional protocol designed to enable the full functionality of the Firebug debugger in a multi-process or remote scenario. Where it was possible, the design of the protocol took cues from existing debug protocols as well as common Web technologies (e.g. HTTP[**?**], JSON[**?**]). However certain features unique to Firebug and to debugging code running inside a Web Browser have to be taken into account.

Code that the user wishes to debug may not always be running.

The user may have several user interfaces in which to interact and debug the runtime.

2.2 Connection and Handshake

Crossfire does not specify a standard or well-known port. Port agreement is left up to the user, or the client software must start the server listening on the same port it will attempt to connect to.

The Crossfire server listens on for a TCP connection on the specified port (greater than 1024). A client wishing to connect sends the string "CrossfireHandshake" followed by a CRLF. The server replies with the same string, at which point the connection is established and the client may begin sending requests and receiving events from the server.

2.3 Message Packets

A well-formed Crossfire packet contains one or more headers consisting of the header name, followed by a colon (":"), the header value, and terminated by a CRLF. A "Content-Length" header containing the number of characters in the message body is required.

The message body is separated from the headers by a blank line (CRLF), followed by a well-formed JSON string, and terminated by a CRLF. The message must contain a "type" field with the value one of "request", "response", or "event", and a "seq" field which contains the sequence number of the packet.

Example: TODO example

2.4 Extensibility

2.5 Contexts

A context in Firebug represents a single Web page.

2.6 Breakpoints

Breakpoint debugging is a standard tool for debugging software at runtime in many languages and environments. The Web Browser environment creates several challenges for designing a remote protocol which supports breakpoint debugging. Firebug also introduces several types of breakpoints which are not present in other environments [**?**]

3. Implementation

3.1 Crossfire Firefox Extension

The Crossfire extension implements the protocol as an extension to Firefox and Firebug.

3.2 Crossfire Tools API

The Crossfire extension also implements an API, called the "Crossfire Tools API" which enables extensibility of the Crossfire system and protocol.

3.3 Modules

3.4 Browser Tools Interface

4. Related Work

4.1 GDB

4.2 JNDI/JDWP

4.3 DBGP

4.4 Opera Scope

Opera Scope Protocol [**?**]

4.5 V8 / Chrome Dev Tools

4.6 wienre

5. Future Work

5.1 Web Sockets

Multi-user Debugging

A. Appendix Title

This is the text of the appendix, if you need one.

Acknowledgments

Acknowledgments, if needed.