

Debugging with Fiddler

The complete reference from the creator of the Fiddler Web Debugger

This is a **SAMPLE** containing the **Table of Contents** and a bit of content so you can decide whether the book meets your needs and renders nicely on your device.

Buy the book at <http://www.fiddlerbook.com>

Second Edition

Eric Lawrence

Debugging with Fiddler (2nd Edition)

Cover Photo: Nicholas Wave (@by_nicholas) ©iStockphoto.com

Everything else: ©2015 Eric Lawrence. All rights reserved. *Please don't pirate this book* in whole or in part. Beyond the twelve years I've spent developing Fiddler, I spent a year writing and revising this book. I now have a young son, and merry-go-round rides aren't free. :)

Book Version LULU 2.00 / Fiddler Version 2.5.0.0

Published March 5, 2015 – Austin, TX

Legalese

Fiddler and related product names are trademarks of Telerik. Other trademarks mentioned in this book are (obviously) the property of their respective owners, and are only used to identify the products or services mentioned.

This book is provided "as is." In no event shall I, the author, be liable for any consequential, special, incidental or indirect damages of any kind arising out of the delivery, accuracy, or use of this book. This book was written with care, but no one warrants that it is error-free. On the contrary, I guarantee that this book contains at least a few errors, and I promise to be suitably embarrassed when you point them out to me (<http://fiddlerbook.com/errata>) so that I may update the next version.

e_lawrence@hotmail.com

@ericlaw on Twitter

ACKNOWLEDGEMENTS

This book, and Fiddler itself, would not have been possible without myriad contributions from hundreds of people around the world.

First, I'd like to thank my wife and son for their inspiration and encouragement as I spent innumerable nights and weekends working on Fiddler and authoring this book. Next, thanks to my parents and grandmother, who instilled in me a voracious appetite for books and the idea that one day I should try my hand at writing one.

I'm grateful for the many contributions of colleagues too numerous to mention (they know who they are!), and to the broader Fiddler community for providing a steady stream of encouragement, suggestions and bug reports. I'd like to thank my employer, Telerik, who acquired Fiddler in 2012 and generously continues to fund my work on the platform.

Finally, I thank you, dear reader, for caring enough about Fiddler to pick up this book!

TABLE OF CONTENTS

Acknowledgements	iii
Table of Contents	iv
INTRODUCTION	1
Origins	2
About this book	4
A Quick Primer	6
Basic Concepts.....	6
Usage Scenarios.....	7
An Incomplete List of Things Fiddler Can Do.....	7
An Incomplete List of Things Fiddler Cannot Do	8
EXPLORE FIDDLER	9
Get Started	10
System Requirements – Windows	10
System Requirements - Mono.....	10
Install Fiddler on Windows.....	11
Permissions and XCOPY Deployment	11
Update Fiddler	11
Uninstall Fiddler	12
The Fiddler User-Interface	13
Web Sessions List	14
Drag and Drop Support	14
Icons and Colors.....	15
Keyboard Reference	16
Web Sessions Context Menu	17
Customize Columns	20
Fiddler’s Main Menu.....	24
File Menu	24
Edit Menu.....	24
Rules Menu	25
Performance Submenu	26
Tools Menu.....	27
View Menu.....	28
Help Menu	29
Fiddler’s About Box	30

Fiddler's Toolbar	31
Fiddler's Status Bar.....	32
QuickExec	34
QuickExec Selection Commands	34
Default FiddlerScript Commands	36
Application Hotkeys	39
Statistics Tab	40
Filters tab	42
Hosts	42
Client Process	43
Request Headers	43
Breakpoints	44
Response Status Code	44
Response Type and Size	45
Response Headers	45
Timeline tab	47
Mode: Timeline	47
Mode: Client Pipe Map.....	49
Mode: Server Pipe Map	49
Using the Timeline for Performance Analysis	49
AutoResponder tab	50
Specify the Match Condition	51
Match Request Method	52
Match Session Flags	52
Match Request Headers	52
Match Request Bodies	53
"Dice Roll" Matches	53
Specify the Action Text.....	54
Use RegEx Replacements in Action Text	55
Drag-and-Drop support	56
Work with the Rules List	56
FARX Files.....	57
Playback Mode	57
TextWizard	59
Composer tab.....	61
Request Options	61
Scratchpad Requests.....	61

Raw Requests	62
Parsed Requests.....	62
The Composer History List	62
Send Sequential Requests.....	63
File Upload Requests	64
Edit with an Automatic Request Breakpoint	65
Override the Host Header	66
Log tab	67
Find Sessions Window	68
Session Clipboards.....	70
SAZ AutoSave.....	71
Host Remapping Tool	72
TECHNIQUES AND CONCEPTS	73
Retarget Traffic with Fiddler	74
Method #1 - Rewrite	74
Method #2 - Reroute	74
Method #3 - Redirect	75
Features to Retarget Requests.....	75
Compare Sessions	77
UltraDiff.....	78
Compare Multiple Sessions at Once	78
Debug with Breakpoints	79
Set Breakpoints	79
Tamper Using Inspectors	80
The Breakpoint Bar	80
Resuming Multiple Sessions	81
CONFIGURE FIDDLER AND CLIENTS	83
Fiddler Options.....	84
General Tab	84
HTTPS Tab	85
Connections Tab.....	87
Gateway Tab	88
Appearance Tab	89
Extensions Tab	89
Tools Tab	90
HeaderEncoding Setting	91

Preferences.....	92
Configure Clients.....	93
Capture Traffic from Browsers.....	93
Chrome.....	93
Firefox.....	93
Opera.....	94
Other Browsers.....	94
Capture Traffic from Other Applications.....	95
WinHTTP.....	95
.NET Framework.....	95
Java.....	96
PHP / cURL.....	96
Capture Traffic from Services.....	97
Capture Traffic to Loopback.....	97
Loopback Bypasses.....	97
Loopback Authentication.....	98
Loopback Blocked from Metro-style Windows 8 Apps.....	98
Run Fiddler in a Virtual Machine on Mac OS X.....	99
Capture Traffic from Other Computers.....	101
Capture Traffic from Devices.....	102
Apple iOS Proxy Settings.....	103
Windows Phone Proxy Settings.....	103
Windows RT Proxy Settings.....	103
Other Devices.....	104
Use Fiddler as a Reverse Proxy.....	104
Acting as a Reverse Proxy for HTTPS.....	105
Chain to Upstream Proxy Servers.....	105
Chain to SOCKS / TOR.....	106
VPNs, Modems, and Tethering.....	107
DirectAccess.....	107
Windows Phone Tethering.....	108
Memory Usage and Fiddler's Bitness.....	109
Buffering vs. Streaming Traffic.....	111
Request Buffering.....	111
Response Buffering.....	111
COMET.....	112
HTML5 WebSockets.....	113

WebSocketMessage Objects	115
Payload Masking	115
Fiddler and HTTPS	116
Trust the Fiddler Root Certificate	117
Machine-wide Trust on Windows 8+	118
Manually Trust the Fiddler Root	118
Additional HTTPS Options	119
Configure Clients for HTTPS Decryption	120
Browsers	120
Firefox	120
Opera	120
Cross-machine scenarios	120
HTTPS and Devices	121
Windows Phone	121
Android and iOS	121
Buggy HTTPS Servers	123
Certificate Validation	123
Certificate Pinning	124
Use an Existing Certificate	124
Fiddler and FTP	125
Fiddler and Web Authentication	126
HTTP Authentication	126
Automatic Authentication in Fiddler	127
Authentication Problems	128
Channel-Binding	128
WinHTTP Credential Release Policy	128
Loopback Protection	128
HTTPS Client Certificates	129
INSPECTORS	131
Overview	132
Auth	134
Caching	136
Cookies	137
Headers	138
Context Menu	139
Keyboard Shortcuts	139
Editing	139

HexView	141
ImageView	143
Metadata Display	144
ImageView Tools.....	144
PNGDistill Image Tool.....	145
JSON	146
Raw	147
PDFView	148
SyntaxView	149
TextView	151
Transformer	152
Background on HTTP Encodings	152
Add and Remove Encodings using the Transformer	153
Other Ways to Remove Encodings	154
WebForms.....	155
WebView	156
XML	157
EXTENSIONS	159
Overview	160
Popular 3 rd Party Extensions	160
Performance Add-ons	160
Security Add-ons	160
Extensions I've Built	161
JavaScript Formatter	162
Gallery	163
Full-Screen View	163
Show Image Bloat	165
Content Blocker.....	167
Traffic Differ	168
FiddlerScript Editors.....	169
FiddlerScript Tab.....	169
ClassView Sidebar.....	170
Fiddler ScriptEditor.....	170
AnyWHERE.....	172
STORE, IMPORT, AND EXPORT TRAFFIC	173
Session Archive Zip (SAZ) Files.....	174

Protecting SAZ Files	174
FiddlerCap	176
Capture Box.....	176
Capture Options Box	177
Tools Box	178
Fiddler's Viewer Mode	180
Import and Export Sessions	181
Import Formats	181
Build Your Own.....	182
Export Formats	182
cURL Script	182
HTML5 AppCache Manifest	182
HTTPArchive v1.1 and v1.2	184
MeddlerScript	184
Raw Files.....	185
Visual Studio WebTest	185
WCAT Script.....	185
Build Your Own.....	186
FIDDLERSCRIPT.....	187
Extend Fiddler with FiddlerScript.....	188
About FiddlerScript.....	188
Edit FiddlerScript	189
Update FiddlerScript at Runtime	190
Reset to the Default FiddlerScript	190
FiddlerScript Functions.....	191
Session Handling Functions	191
OnPeekAtRequestHeaders	191
OnBeforeRequest	191
OnPeekAtResponseHeaders	191
OnWebSocketMessage	191
OnBeforeResponse	192
OnReturningError.....	192
OnDone.....	192
General Functions	192
Main.....	192
OnRetire.....	192
OnBoot	192

OnShutdown.....	192
OnAttach.....	193
OnDetach.....	193
OnExecAction(sParams: string[]).....	193
FiddlerScript and Automation Tools.....	194
Quiet Mode.....	194
Driving Fiddler from Batch Scripts.....	194
Driving Fiddler from Native or .NET Code.....	195
Extend Fiddler's UI - Menus.....	197
Extend the Tools Menu.....	197
Extend the Web Sessions Context Menu.....	198
Extend the Rules Menu.....	198
Boolean-bound Rules.....	198
String-bound Rules.....	200
Binding Script variables to Preferences.....	201
Creating New Top-Level Menus.....	202
Extend Fiddler's UI - Adding Tabs.....	203
Extend Fiddler's UI - Adding Columns to the Web Sessions List.....	204
Binding Columns using Attributes.....	204
Binding Columns using AddBoundColumn.....	206
FiddlerObject Functions.....	208
FiddlerObject.ReloadScript().....	208
FiddlerObject.StatusText.....	208
FiddlerObject.log(sTextToLog).....	208
FiddlerObject.playSound(sSoundFilename).....	209
FiddlerObject.flashWindow().....	209
FiddlerObject.alert(sMessage).....	209
FiddlerObject.prompt(sMessage).....	209
FiddlerObject.createDictionary().....	209
FiddlerObject.WatchPreference(sPrefBranch, oFunc).....	210
Import Assemblies.....	211
Example Scripts.....	212
Request Scripts.....	212
Add (or Overwrite) a Request Header.....	212
Remove Request Headers.....	212
Flag Requests that Send Cookies.....	212
Rewrite a Request from HTTP to HTTPS.....	212

Swap the Host Header	212
Drop a Connection	213
Prevent Response Streaming.....	213
Response Scripts.....	213
Hide Sessions that Returned Images	214
Flag Redirections.....	214
Replace Text in Script, CSS, and HTML	214
Remove All DIV Elements.....	214
Other Scripts.....	214
Add a Systemwide Hotkey	214
Certificate Info Custom Column	215
Generate Mock Sessions.....	216
Show Response Hash.....	216
Combine Partial Responses.....	217
Remove Many Headers at Once.....	217
Override MIME Types	218
Hide Traffic based on Process Name.....	218
More Examples	219
EXTEND FIDDLER WITH .NET CODE	221
Extend Fiddler with .NET	222
Project Requirements and Settings	222
Debugging Extensions.....	222
Best Practices for Extensions.....	223
Best Practice: Use an Enable Switch	223
Best Practice: Use Delay Load	223
Best Practice: Beware “Big Data”.....	225
Best Practice: Use the Reporter Pattern for Extensions	225
Interact with Fiddler’s Objects	227
The Web Sessions List	227
Session[] GetAllSessions()	227
Session GetFirstSelectedSession().....	227
Session[] GetSelectedSessions()	227
Session[] GetSelectedSessions(int iMax).....	227
void actSelectAll()	227
void actSelectSessionsMatchingCriteria(doesSessionMatchCriteriaDelegate oDel)	227
void actRemoveSelectedSessions()	228
void actRemoveUnselectedSessions()	228

bool actLoadSessionArchive(string sFilename)	228
void actSaveSessionsToZip()	228
void actSaveSessionsToZip(string sFilename, string sPwd)	228
void actSessionCopyURL()	228
void actSessionCopySummary()	228
void actSessionCopyHeadlines()	228
int FiddlerApplication.UI.lvSessions.SelectedCount	229
SimpleEventHandler FiddlerApplication.UI.lvSessions.OnSessionsAdded	229
Session Objects	229
oRequest	229
requestBodyBytes	229
oResponse	229
responseBodyBytes	230
oFlags	230
void Abort()	230
bBufferResponse	230
bHasReponse	230
bypassGateway	230
clientIP	230
clientPort	230
bool COMETPeek()	231
fullUrl	231
PathAndQuery	231
port	231
host	231
hostname	231
bool HostnameIs(string)	231
bool HTTPMethodIs(string)	231
bool uriContains(string)	231
id	231
isFTP	231
isHTTPS	231
isTunnel	231
LocalProcessID	232
bool utilDecodeRequest()	232
bool utilDecodeResponse()	232
bool RefreshUI()	232

RequestMethod	232
responseCode	232
state	232
BitFlags	233
Timers	234
HostList Objects	235
Sending Strings to the TextWizard	236
Logging	237
Interacting with the FiddlerScript Engine	238
Program with Preferences	239
Preference Naming	239
The IFiddlerPreferences Interface	239
Store and Remove Preferences	240
Retrieve Preferences	240
Watch for Preference Changes	241
Notifications in Extensions	241
Notifications in FiddlerScript	241
Build Extension Installers	242
Build Inspectors	245
Inspect Session Objects	249
Deal with HTTP Compression and Chunking	251
Decode a Copy of the Body	251
Use the GetRe*BodyAsString Methods	252
Use the utilDecode* Methods	252
Inspector Assemblies	253
Build Extensions	254
Understand Threading	255
Integrate with QuickExec	255
Example Extension	256
Extension Assemblies	261
Build Import and Export Transcoders	262
Direct Fiddler to load your Transcoder assemblies	262
The ProfferFormat Attribute	262
The ISessionImporter Interface	263
The ISessionExporter Interface	264
Handle Options	264
Provide Progress Notifications	265

Notes on Threading and Transcoders in FiddlerCore	266
Beyond Files	266
Example Transcoder	266
FIDDLERCORE	271
Overview	273
Legalities and Licenses	273
Get Started with FiddlerCore	274
Compile the Sample Application.....	274
FiddlerCoreStartupFlags	276
The FiddlerApplication Class	278
FiddlerApplication Events.....	278
OnReadRequestBuffer Event.....	278
RequestHeadersAvailable Event	278
BeforeRequest Event	278
OnValidateServerCertificate Event	278
OnReadResponseBuffer Event	279
ResponseHeadersAvailable Event.....	280
BeforeResponse Event.....	280
BeforeReturningError Event.....	280
AfterSessionComplete Event.....	281
OnWebSocketMessage	281
FiddlerAttach Event	281
FiddlerDetach Event	281
OnClearCache Event.....	281
OnNotification Event	281
FiddlerApplication Methods	281
Startup()	281
Shutdown()	281
IsStarted().....	281
IsSystemProxy()	281
CreateProxyEndpoint()	282
DoImport()	282
DoExport()	282
GetVersionString()	282
GetDetailedInfo()	283
ResetSessionCounter().....	283
FiddlerApplication Properties and Fields.....	283

isClosing	283
Log	283
oDefaultClientCertificate	283
oProxy	283
oTranscoders.....	283
Prefs	283
The Rest of the Fiddler API	283
Common Tasks with FiddlerCore	284
Keep Track of Sessions	284
Get Traffic to FiddlerCore	284
Trust the FiddlerCore Certificate.....	285
Generate Responses	286
Other Resources	286
APPENDICES	287
Appendix A: Troubleshooting	288
Missing Traffic.....	288
Interference from Security Software	289
Problems Downloading Fiddler	289
Problems Installing Fiddler	289
Problems Running Fiddler	289
Corrupted Proxy Settings	290
Resetting Fiddler	290
Troubleshoot Certificate Problems	291
Wipe all traces of Fiddler.....	292
Fiddler complains about the "Configuration System"	292
Fiddler randomly stops capturing traffic.....	293
Fiddler may stall when streaming RPC-over-HTTPS traffic	293
Appendix B: Command Line Syntax.....	295
Option Flags	295
Examples.....	295
Appendix C: Session Flags.....	296
Session Display Flags	296
Breakpoint and Editing Flags	298
Networking Flags.....	298
Authentication Flags	300
Client Information Flags.....	300
Performance Simulation Flags.....	301

HTTPS Flags.....	301
Request Composer Flags	303
Other Flags	304
Appendix D: Preferences	307
Network Preferences	307
HTTPS Preferences.....	312
Fiddler UI Preferences.....	314
FiddlerScript Preferences.....	319
TextWizard Preferences.....	319
Request Composer Preferences	319
Path Configuration.....	320
Miscellaneous.....	321
Extension Preferences.....	322
Raw Inspector	322
JavaScript Formatter	322
Certificate Maker Add-on	323
Index.....	325

About this book

Over twelve years and well over a hundred version releases, Fiddler has evolved into a powerful utility and platform that can perform a wide variety of tasks. It has a rich extensibility model and a community of add-on developers who have broadened its usefulness as a performance, security, and load-testing tool. Questions in email, online discussion groups, and numerous conferences over the years made it overwhelmingly apparent that most users only exploit a tiny fraction of Fiddler's power. I came to realize that thousands of users would get a lot more out of Fiddler if there were a complete reference to the tool available. The first version of this book, released in 2012, was the product of that realization. This Second Edition, released in early 2015, builds upon the first and adds new content covering major enhancements to Fiddler since the book was originally published.

As Fiddler's developer, I've found it both easy and challenging to write this book. It's easy, because I understand Fiddler deeply, down to its very foundation, and can consult the source code to research obscure details. On the other hand, it's been very challenging, as every time I choose an interesting scenario or feature to write about, I'm forced to think deeply about that scenario or feature. Commonly, I've found myself developing improvements to revise Fiddler and minimize or eliminate the need to write about the topic in the first place. As a result, I've rewritten large portions of both this book and Fiddler itself. It's been a slow process, but both projects have benefitted.

Publication of this edition roughly coincided with the release of Fiddler version 2.5 in the winter of 2015. If you're using a later version of Fiddler, you will find some minor differences, but the core concepts will remain the same.

This book is deliberately limited in scope—it covers nearly every aspect of Fiddler and FiddlerCore, but it is not a tutorial on HTTP, SSL/TLS, HTML, Web Services or the myriad other topics you may want to understand to fully exploit Fiddler's feature set. If you want a deeper understanding of web protocols, I can recommend the references that I consulted during the development of Fiddler:

- [Hypertext Transfer Protocol -- HTTP/1.1](http://www.ietf.org/rfc/rfc2616.txt) from <http://www.ietf.org/rfc/rfc2616.txt>, later obsoleted by the HTTPbis Working Group's release of RFC7230 through RFC7235.
- [HTTP: The Definitive Guide](#) by David Gourley
- [Web Protocols and Practice: HTTP/1.1, Networking Protocols, Caching, and Traffic Measurement](#) by Balachander Krishnamurthy and Jennifer Rexford
- [SSL & TLS Essentials: Securing the Web](#) by Stephen A. Thomas
- [Bulletproof SSL and TLS](#) by Ivan Ristić
























This book can be read either "straight through" or you can use the Table of Contents and Index to find the topics most interesting to you. Please consider skimming all of the chapters, even those that don't seem relevant to your needs, because each chapter often contains tips and tricks you might not find elsewhere.






I encourage you to begin by reading the primer in the next chapter, which lays out some terminology and the basic concepts that you'll need to understand to get the most out of Fiddler and this book. Enjoy!

Icons and Colors

The default text coloring of each row in the Web Sessions list derives from the HTTP Status (red for errors, yellow for authentication demands), traffic type (**CONNECT**s appear in grey), or response type (CSS in purple, HTML in blue; script in green, images in grey). You can override the font color by setting the Session's **ui-color** flag from Fiddler-Script.

Each row is also marked with an icon for quick reference as to the Session's progress, Request type, or Response type:

	Request is being sent to the server.
	Response is being downloaded from the server.
	Request is paused at a breakpoint to allow tampering.
	Response is paused at a breakpoint to allow tampering.
	Request used the HEAD or OPTIONS methods, or returned a HTTP/204 status code. The HEAD and OPTIONS methods allow the client to acquire information about the target URL or server without downloading the content. The HTTP/204 status code indicates that there is no response body.
	Request used the POST method to send data to the server.
	Response is HTML.
	Response is an image.
	Response is a script.
	Response is a Cascading Style Sheet (CSS).
	Response is Extensible Markup Language (XML).
	Response is JavaScript Object Notation (JSON).
	Response is an audio file.
	Response is a video file.
	Response is a Silverlight applet.
	Response is a Flash applet.
	Response is a font.
	Response's Content-Type is not a type for which a more specific icon is available.
	Request used the CONNECT method. This method is used to establish a tunnel through which encrypted HTTPS traffic flows.
	Session is a CONNECT tunnel inside which Google's SPDY protocol is used.
	Session is a CONNECT tunnel inside which the HTTP2 protocol is used.
	Session is an RPC-over-HTTP tunnel; most commonly used by Microsoft Outlook.
	Session wraps a HTML5 WebSocket connection.
	Response is a HTTP/3xx class redirect.
	Response is a HTTP/401 or HTTP/407 demand for client credentials, or a HTTP/403 error indicating that access was denied.

	Response has a HTTP/4xx or HTTP/5xx error status code.
	Session was aborted by the client application, Fiddler, or the Server. This commonly occurs when the client browser began downloading of a page, but the user then navigated to a different page. The client browser responds by canceling all in-progress requests, leading to the Aborted Session state.
	Response is a HTTP/206 partial response. Such responses are returned as a result of the client performing a Range request for only a portion of the file at the target URL.
	Response has a HTTP/304 status, indicating that the client's cached copy is fresh.
	Session is unlocked, enabling modification after normal Session processing has completed.

FIDDLER'S TOOLBAR

The toolbar provides quick access to popular commands and settings.



The buttons and their functions are:

WinConfig	<i>Shown only on Windows 8 and later.</i> Launches a tool that configures “Immersive” apps to permit sending traffic to Fiddler. Hold the CTRL key while clicking to automatically configure all applications.
<i>Comment icon</i>	Add a Comment to all selected Sessions. The comment appears in a column of the Web Sessions list. Hold the SHIFT key while clicking to add a new mock Session to the list with the comment of your choice.
Replay	Reissue the selected requests to the server. Hold the CTRL key while clicking to reissue the requests without any Conditional Request headers (e.g. If-Modified-Since and If-None-Match). Hold the SHIFT key while clicking to be prompted to specify the number of times each request should be reissued.
<i>Remove icon</i>	Show a menu of options for removing Sessions from the Web Sessions list: <ul style="list-style-type: none">• Remove all removes all Sessions from the list.• Images removes all Sessions that returned an image.• CONNECTs removes all CONNECT tunnels.• Non-200s removes all non-HTTP/200 responses.• Non-Browser removes all requests that were not issued by a web browser.• Complete & Unmarked removes Sessions in the Done or Aborted states that are unmarked and have no Comment set.• Duplicate response bodies removes any Session which has no response body or has a response body identical to one received in an earlier Session.
Go	Resume all Sessions which are currently paused at a Request or Response breakpoint. Hold the SHIFT key while clicking to resume only selected Sessions.
Stream	Enable the Stream toggle to deactivate response buffering for all responses except those for which a breakpoint is set.
Decode	Enable the Decode toggle to remove all HTTP Content and Transfer encodings from requests and responses.
Keep: <i>value</i>	Control how many Sessions are stored in the Web Sessions list. When the limit is reached, Fiddler will begin removing older Sessions to attempt to limit the list to the desired value. Incomplete Sessions and those with comments, markers, or open Inspector windows are not removed.
Process Filter	Drag and drop the Process Filter icon to an application to create a filter that hides all traffic except that originating from the selected process. Right-click the Process Filter icon to clear a previously set filter.

Simply declaring the attributed `m_Hide304s` variable doesn't yet do anything useful—the variable simply tracks the state of the menu item. The *functionality* of the rule is provided by a block of code added to the `OnBeforeResponse` method found later in the script:

```
if (m_Hide304s && (304 == oSession.responseCode)) {  
    oSession["ui-hide"] = "true";  
  
    // Note: This block could be placed in the OnPeekAtResponseHeaders method,  
    // since it does not depend upon the availability of the response body.  
}
```

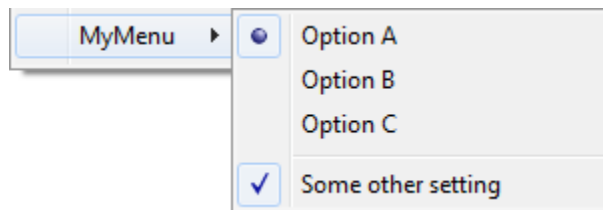
This block first checks to see if the rule is enabled and, if so, checks that the server returned a `HTTP/304`. If so, the block sets the `ui-hide` flag on the Session, which causes it to be hidden from the Web Sessions list.

You can also use different forms of the `RulesOption` attribute to create submenus of options. To do so, provide the name of the submenu as the second parameter of the attribute. For instance, the following three fields create a Performance submenu that exposes the three options:

```
RulesOption("Simulate &Modem Speeds", "Per&formance")  
public static var m_SimulateModem: boolean = false;  
  
RulesOption("&Disable Caching", "Per&formance")  
public static var m_DisableCaching: boolean = false;  
  
RulesOption("&Show Time-to-Last-Byte", "Per&formance")  
public static var m_ShowTTLB: boolean = false;
```

If you would like some of the items on your submenu to be mutually exclusive (showing as a radio group instead of a set of checkboxes), you can set a third boolean parameter to `true`, and you can add a splitter after a menu item by setting yet a fourth boolean parameter to `true`.

To create this menu:



Add the following script:

```
RulesOption("Option A", "MyMenu", true)  
public static var m_OptionA: boolean = true;  
  
RulesOption("Option B", "MyMenu", true)
```

EXTEND FIDDLER'S UI - ADDING TABS

While most major UI extensions are built in C#, you can add simple UI tabs very easily using FiddlerScript.

For example, say that you've decided that <http://httpstatusdogs.com> is the coolest site on the Internet, and you want to enhance Fiddler with this meme. Doing so is super-simple with FiddlerScript.

At the very top of the script file, add the line:

```
import System.Text;
```

Then, move the cursor to just inside the Handlers class. There, add the following code:

```
public BindUITab("HTTPStatusDogs", true)
static function ShowStatusDogs(arrSess: Session[]):String
{
    if (arrSess.Length < 1) return "<html>Please select one or more Sessions.</html>";

    var oSB: System.Text.StringBuilder = new System.Text.StringBuilder();
    oSB.Append("<html><head>");
    oSB.Append("<style>iframe { width: '100%'; height: 600px; frameBorder:0 }</style>");
    oSB.Append("</head><body>");
    for (var i:int = 0; i<arrSess.Length; i++)
    {
        oSB.AppendFormat(
            "<iframe frameBorder=0 scrolling='no' src='http://httpstatusdogs.com/{0}'></iframe>",
            arrSess[i].responseCode);
    }
    oSB.Append("</body></html>");
    return oSB.ToString();
}
```

When you save the script, it will automatically recompile and a new "HTTPStatusDogs" tab will appear; when you activate it, the image for each Selected Session's HTTP response code will be shown in the tab.

The "magic" that makes this work is invoked by the `BindUITab` attribute atop the function declaration:

```
public BindUITab("HTTPStatusDogs", true)
static function ShowStatusDogs(arrSess: Session[]):String
```

The presence of this attribute informs Fiddler that the following function will provide data to be rendered to a new tab, whose name is provided by the first parameter, "HTTPStatusDogs". The second parameter, `true`, indicates that the string returned by the function should be rendered as HTML in a web browser view. To easily debug your HTML, change that `true` to `false`, and Fiddler will instead show the returned string as plain text in a textbox.

BitFlags

Each Session's `BitFlags` property holds zero or more `SessionFlags` that supply commonly-queried state information. The `SessionFlags` enumeration includes:

SessionFlags	Description
None	No flags are set.
IsHTTPS	The request's URI has a HTTPS target.
IsFTP	The request's URI has a FTP target.
Ignored	The Session should be ignored: hide from the Web Sessions list, do not store the request or response, and do not invoke event handlers.
ClientPipeReused	The request was read from a previously used connection from the client.
ServerPipeReused	The request reused an existing connection to the server.
RequestStreamed	The request was transmitted to the server as soon as the headers were complete; any request body was streamed to the server as it was read from the client.
ResponseStreamed	The response was not buffered and was instead streamed to the client as it was read from the server.
RequestGeneratedByFiddler	The request was generated by Fiddler itself (e.g. from the Composer tab).
ResponseGeneratedByFiddler	The response was generated by Fiddler itself (e.g. AutoResponder or the <code>utilCreateResponseAndBypassServer</code> method).
LoadedFromSAZ	This previously-captured Session was reloaded from a SAZ file.
ImportedFromOtherTool	The Session was imported by a Transcoder.
SentToGateway	The request was sent to an upstream (CERN) gateway proxy.
IsBlindTunnel	This <code>CONNECT</code> tunnel "blindly" shuttles bytes across without decryption.
IsDecryptingTunnel	This <code>CONNECT</code> tunnel decrypts HTTPS traffic as it flows through.
ServedFromCache	The response was served from a client cache, bypassing Fiddler. Fiddler only "sees" this Session if other software reports it to Fiddler, because Fiddler only observes network traffic.
ProtocolViolationInRequest	There was a HTTP Protocol violation in the client's request.
ProtocolViolationInResponse	There was a HTTP Protocol violation in the server's response.
ResponseBodyDropped	The response body was not stored, e.g. because the Preference <code>fiddler.network.streaming.ForgetStreamedData</code> is set.
IsWebSocketTunnel	This <code>CONNECT</code> tunnel is used for WebSocket traffic.
SentToSOCKSGateway	The request was proxied using the SOCKS protocol.
RequestBodyDropped	The request body was not stored, e.g. because the Session's <code>Log-Drop-Request-Body</code> flag was set.
IsRPCTunnel	The request created an RPC-over-HTTPS tunnel.

Modify the default `class1.cs` (or create a new class) in your project as follows:

```
using System;
using System.IO;
using System.Text;
using System.Collections.Generic;
using System.Windows.Forms;
using Fiddler;

[assembly: AssemblyVersion("1.0.0.0")]
[assembly: Fiddler.RequiredVersion("2.4.9.5")]

// Note that this Transcoder only works when loaded by Fiddler itself; it will
// not work from a FiddlerCore-based application. The reason is that the output
// uses the columns shown in Fiddler's Web Sessions list, and FiddlerCore has
// no such list.

// Ensure your class is public, or Fiddler won't see it!
[ProfferFormat("Tab-Separated Values", "Session List in Tab-Delimited Format")]
[ProfferFormat("Comma-Separated Values",
    "Session List in Comma-Delimited Format; import into Excel or other tools")]
public class CSVTranscoder : ISessionExporter
{
    public bool ExportSessions(string sFormat, Session[] oSessions,
        Dictionary<string, object> dictOptions,
        EventHandler<ProgressCallbackEventArgs> evtProgressNotifications)
    {
        bool bResult = false;
        string chSplit;

        // Determine if we already have a filename
        // from the dictOptions collection
        string sFilename = null;
        if (null != dictOptions && dictOptions.ContainsKey("Filename"))
        {
            sFilename = dictOptions["Filename"] as string;
        }

        // If we don't yet have a filename, prompt the user
        // with a File Save dialog, using the correct file extension
        // for the export format they selected
    }
}
```

APPENDIX B: COMMAND LINE SYNTAX

`Fiddler.exe` accepts zero or more command-line arguments, consisting of zero or more option flags and a single filename of a file to load on startup.

```
fiddler.exe [options] [FileToLoad]
```

The `FileToLoad` argument may either be a SAZ file or a file of an importable type (e.g. `.pcap` or `.har`).

Fiddler registers itself in Windows' `AppPaths` key so that you can launch it by typing `fiddler` in the shell's Start > Run prompt (hit `Windows+R`) instead of specifying a fully-qualified path to `fiddler.exe`.

Option Flags

Option flags may be preceded by either a `/` or `-` character.

Flag	Description
<code>-?</code>	Show the list of available command line arguments.
<code>-viewer</code>	Open a Error! Reference source not found. instance.
<code>-quiet</code>	Launch in "quiet" mode, where prompts and alerts are suppressed, and the main window is minimized to the system tray. This mode is most often used when Fiddler is running as a part of an automated script.
<code>-noattach</code>	Do not register as the system proxy on startup, even if otherwise configured to do so. You can manually register Fiddler as the proxy for an individual application or set it as the system proxy using the option on the File menu.
<code>-noversioncheck</code>	Do not send a web service request to check for updates on startup.
<code>-extoff</code>	Do not load Fiddler Inspectors or Extensions. This flag is used to troubleshoot problems related to buggy extensions.
<code>-noscript</code>	Do not load FiddlerScript. This flag is used to determine whether your FiddlerScript is causing some problem.
<code>-port:####</code>	Specify the port that Fiddler should listen on, overriding the default setting configured in the Fiddler Options window.

Examples

Launch Fiddler without attaching:

```
"C:\program files (x86)\Fiddler2\fiddler.exe" -noattach
```

Launch Fiddler with no UI, running on port 1234:

```
fiddler -port:1234 -quiet
```

Open a SAZ file in a new Fiddler Viewer instance:

```
fiddler -viewer "C:\users\joe\desktop\Sample.saz"
```

Networking Flags

The following flags control and log Fiddler's use of the network:

Flag Name	<code>x-overrideHost</code>
Explanation	Controls the hostname used for DNS resolution (and optionally the target port) when deciding what address this request should be sent to. This flag will not change any of the Headers in the request itself.
Supported Values	Specify either an alternative hostname or an IP Address (and optionally a Port) to which this request should be targeted. Setting this flag may have no effect if the request is sent to an upstream gateway, because the upstream gateway will perform its own DNS resolution.

Flag Name	<code>x-overrideHostName</code>
Explanation	Controls the hostname used for DNS resolution when deciding what address this request should be sent to. This flag will not change any of the Headers in the request itself.
Supported Values	Specify either an alternative hostname or an IP Address to which this request should be targeted. Setting this flag may have no effect if the request is sent to an upstream gateway, because the upstream gateway will perform its own DNS resolution.

Flag Name	<code>x-OverrideGateway</code>
Explanation	Controls which upstream gateway proxy, if any, this request is sent to.
Supported Values	<p>Provide a string that specifies the target gateway proxy in the format <code>ProxyHost:Port</code>, for example, <code>myproxy:8080</code>. The provided address information will be used instead of any default gateway proxy.</p> <p>If you prefix the string with <code>socks=</code> the provided gateway will be used as a SOCKS proxy. For instance, the string <code>socks=127.0.0.1:9150</code> can be used to send traffic to a locally running instance of the Tor SOCKS proxy.</p> <p>A value of <code>DIRECT</code> means that the request will be sent directly to the server, bypassing any gateway. This value is equivalent to setting the Session's <code>bypassGateway</code> boolean to <code>true</code>.</p>

Flag Name	<code>x-ReplyWithTunnel</code>
Explanation	<p>When set on a <code>CONNECT</code> tunnel's Session, this flag causes Fiddler to automatically respond with a <code>200 OK</code> response without contacting the server or gateway.</p> <p>This flag is set by the AutoResponder to enable capture of HTTPS requests from a client even when the system is offline. Otherwise, the <code>CONNECT</code>'s failure would prevent subsequent HTTPS requests from being sent by the client.</p>
Supported Values	Any value will result in returning a <code>200 OK</code> response to the <code>CONNECT</code> .

Flag Name	<code>FTP-UseASCII</code>
------------------	---------------------------

APPENDIX D: PREFERENCES

Fiddler's Preferences system allows you to control myriad aspects of Fiddler's behavior. This appendix contains a list of the Preferences that affect Fiddler and its default set of extensions.

Network Preferences

The following preferences control Fiddler's network behavior:

Name	<code>fiddler.network.timeouts.dnscache</code>
Default	150000 (2.5 minutes)
Explanation	Number of milliseconds for which Fiddler should cache DNS lookup results.

Name	<code>fiddler.network.timeouts.serverpipe.reuse</code>
Default	115000 (115 seconds)
Explanation	Number of milliseconds for which Fiddler is willing to leave a server connection idle. A connection which has been idle for this time without being reused will be closed. The Firefox team has found that values over 115 seconds can cause problems with buggy servers.

Name	<code>fiddler.network.timeouts.clientpipe.receive.initial</code>
Default	60000 (1 minute)
Explanation	Number of milliseconds for which Fiddler is willing to wait for a client to begin sending a request on a newly established connection. After this timeout expires, Fiddler will send a HTTP/408 timeout and close the connection from the client.

Name	<code>fiddler.network.timeouts.clientpipe.receive.reuse</code>
Default	60000 (30 seconds)
Explanation	Number of milliseconds for which Fiddler is willing to wait for a client to begin sending a request on a previously used connection. After this timeout expires, Fiddler will close the connection from the client.

Name	<code>fiddler.network.timeouts.serverpipe.send.initial</code>
Default	-1 ("infinite")
Explanation	Number of milliseconds for which Fiddler is willing to wait when sending to a newly established server connection. After this timeout expires, Fiddler will close the connection to the server.

Name	<code>fiddler.network.timeouts.serverpipe.send.reuse</code>
Default	-1 ("infinite")
Explanation	Number of milliseconds for which Fiddler is willing to wait when sending to a server on a previously used connection. After this timeout expires, Fiddler will close the connection to the server.

INDEX

A

- Android
 - HTTPS Decryption, 121
- AppContainers, 99
- Apple
 - HTTPS Decryption, 121
 - iOS, 103
 - Mac OS X, 99
- audio and video
 - previewing, 156
 - streaming, 111
- authentication
 - Auth Inspector, 134
 - automatic, 127
 - channel-binding, 128
 - client certificates, 129
 - loopback, 98, 128
 - methods, 126
 - problems, 128
 - WinHTTP, 128
- AutoResponder
 - action text, 54
 - drag-and-drop, 56
 - FARX files, 56
 - match condition, 51
 - playback mode, 57
 - regular expressions, 55

B

- bindpref, 201
- breakpoints
 - Breakpoint bar, 80
 - Filters tab, 44
 - overview, 79
 - resuming, 81
 - using QuickExec, 80

C

- caching
 - Inspector, 136
- capturing
 - .NET Framework, 95
 - browsers, 93
 - devices, 102
 - DirectAccess, 107
 - FTP, 125
 - Java, 96
 - loopback, 97
 - Mac OS X, 99
 - other PCs, 101
 - PHP/cURL, 96
 - reverse proxy, 104
 - VPNs/Modems, 107
 - Win8 apps, 98
 - Windows Phone, 103
 - Windows RT, 103
 - WinHTTP, 95
- certificates
 - CertMgr.msc, 118
 - troubleshooting, 291
 - trusted root, 117
 - Windows 8, 118
- clipboard
 - pasting from, 25
 - pasting to, 17
- Clone Response, 20
- code samples
 - extension example, 256
 - general scripts, 214
 - implementing OnValidateServerCertificate, 278
 - RegisterCustomHotkey, 214
 - scripts to manipulate requests, 212
 - scripts to manipulate responses, 213
 - transcoder example, 266
 - trusting a root certificate, 285
- columns, 14

- adding from code, 204
- Customization UI, 20
- SetColumnOrderAndWidth function, 207

COMETPeek, 20

- overview, 112

Composer

- automatic breakpoints, 65
- history list, 62
- options, 61
- overview, 61
- parsed, 62
- raw requests, 62
- scratchpad, 61
- sequential requests, 63
- uploading files, 64

cookies

- Inspector, 137

CPU architecture and bitness, 109

cURL

- capturing from, 96
- export script, 182
- mimic with Composer, 62
- paste as Sessions, 25

E

EnableLoopback.exe, 99

encoding

- decoding from Transformer, 153
- decoding programmatically, 251
- decoding using UI, 154
- headers, 91
- HTTP Chunking, 152
- HTTP Compression, 152
- text encoding overview, 91
- TextWizard, 59

execaction

- driving FiddlerScript, 194

export

- cURL Script, 182
- HTML5 AppCache, 182
- HTTPArchive format, 184
- MeddlerScript, 184
- Raw Files, 185
- Visual Studio WebTest, 185

- WCAT Script, 185

extensions

- AnyWHERE, 172
- assemblies, 261
- best practices, 223
- building in .NET, 222
- Content Blocker, 167
- debugging, 222
- delay-loading, 223
- Developing, 254
- directory, 160, 161
- example code, 256
- FiddlerScript tab, 169
- for Performance analysis, 160
- for Security analysis, 160
- Gallery tab, 163
- installing, 242
- JavaScript Formatter, 162
- options, 89
- overview, 160
- Show Image Bloat, 165
- third-party, 160
- Traffic Differ, 168

F

FiddlerApplication overview, 278

FiddlerCap, 176

FiddlerCore

- common tasks, 284
- get started, 274
- overview, 273
- StartupFlags, 276

FiddlerScript

- adding new tabs, 203
- automation, 194
- ClassView sidebar, 170
- cross-process messaging, 195
- Editing, 169
- engine, 238
- examples, 212
- extending menus, 197
- FiddlerObject, 208
- general functions, 192
- JScript.NET, 188

- new top-level menus, 202
- overview, 188
- referencing assemblies, 211
- resetting, 190
- session-handling functions, 191
- standalone editor, 170

filtering

- Filters tab, 42

Find, 68

fonts

- previewing, 156

FTP

- capturing, 125

H

headers

- create from script, 216
- Inspector, 138
- RemoveRange method, 217

HTTPS

- Certificate Maker plugin, 121
- certificate pinning, 124
- certificate validation, 123
- client certificates, 129
- configuring devices, 121
- configuring other clients, 120
- configuring remote computers, 120
- CONNECT tunnels, 116
- decryption, 116
- handshake failures, 123
- limiting decryption, 119
- options, 85
- overview, 116
- protocol versions, 123
- use existing certificate, 124

I

IFiddlerExtension, 254

import and export

- export formats, 182
- import formats, 181
- overview, 181
- Transcoders, 262

Inspectors

- assemblies, 253
- Auth, 134
- Caching, 136
- Cookies, 137
- develop, 245
- Headers, 138
- HexView, 141
- ImageView, 143
- JSON, 146
- PDFView, 148
- Raw, 147
- SyntaxView, 149
- TextView, 151
- Transformer, 152
- WebForms, 155
- XML, 157

installation, 11

J

JavaScript

- formatting, 162

K

keyboard shortcuts

- general, 13
- hotkeys, 39
- Web Sessions list, 16

L

Log

- macro commands, 237
- writing to, 237

loopback

- authentication, 98
- bypass, 97
- capturing, 97
- Windows 8, 98

M

memory

- conserving, 109

overview, 109

usage, 30

Menus

Copy submenu, 17

Edit menu, 24

extending with extensions, 256

extending with FiddlerScript, 197

File menu, 24

Filter submenu, 18

Help menu, 29

Mark submenu, 19, 25

Performance submenu, 26

Replay submenu, 19

Rules menu, 25

Save submenu, 18

Select submenu, 19

Tools menu, 27

View menu, 28

Web Sessions context menu, 17

N

NetMon, 2

NSIS installer, 242

O

object model

BitFlags, 233

FiddlerApplication events, 278

FiddlerApplication methods, 281

FiddlerApplication properties and fields, 283

FiddlerScript engine, 238

HostList objects, 235

Log, 237

Preferences, 239

Session objects, 229

SessionStates, 232

TextWizard, 236

Timers, 234

Web Sessions list, 227

WebSocketMessage objects, 115

P

P3P, 137

Parallels, 100

performance

add-ons, 160

Caching Inspector, 136

image optimization, 165

simulation, 26

Timers object, 234

PNGDistill, 145

preferences

binding script fields to, 201

Preferences

about:config, 92

IFiddlerPreferences, 239

observe changes, 241

overview, 92

programming with, 239

Reference List, 307

protocols

references, 5

support, 8

proxy server, 6

client applications, 93

SOCKS, 106

upstream chaining, 105

Q

QuickExec

commands, 34

default FiddlerScript commands, 36

extension integration, 255

general commands, 35

overview, 34

selecting Sessions, 34

R

request object, 229

response object, 229

reverse proxy, 104

S

SAZ Files

- encrypting with password, 174
- overview, 174
- save automatically, 71
- saving with FiddlerCap, 176

searching

- Find Sessions window, 68

security

- add-ons, 160

Sessions

- aborting, 20
- adding mock Sessions, 216
- combining partial responses, 217
- compare, 77
- comparing multiple, 168
- hashing responses, 216
- inspecting in a new window, 20
- properties, 20
- rerouting, 74
- Selecting with QuickExec, 34
- Session Clipboards, 70
- unlocking, 20, 25

settings

- Appearance tab, 89
- command-line, 295
- Connections tab, 87
- Extensions tab, 89
- Fiddler Options, 84
- FTP, 125
- gateway tab, 88
- General tab, 84
- HeaderEncoding, 91
- HTTPS tab, 85
- Preferences, 92
- tools tab, 90

status bar

- overview, 32

streaming

- COMET, 112
- minimizing memory usage, 109
- requests, 111
- responses, 111
- Timeline, 48

- vs. buffering, 111

System Requirements, 10

T

Tabs

- adding from extensions, 261
- Composer, 61
- FiddlerScript, 169
- Filters, 42
- Gallery, 163
- Inspectors, 132
- Log, 67
- Statistics, 40

TextWizard

- invoking from code, 236
- overview, 59

threading

- IAutoTamper, 255
- Transcoders, 266

Timeline

- client pipe map, 49
- server pipe map, 49
- timeline mode, 47

timing

- Session Timers, 234
- Statistics tab, 40
- timer resolution, 235

toolbar

- hiding text, 32

tools

- PNGDistill, 145

Tools

- Configure AutoSave, 71
- HOSTS..., 72
- New Session Clipboard, 70

Tor

- chain to, 106

Transcoders, 262

- example code, 266
- progress events, 265
- supporting options, 264

troubleshooting

- IPsec, 103
- out-of-memory, 109

WiFi isolation, 103
Windows Phone tethering, 108

U

upstream gateway
 user-interface, 88

V

viewer mode, 180

W

Web Sessions list
 colors, 15
 columns, 14
 icons, 15
 overview, 14
WebSockets

message objects, 115
overview, 113

windows

 About box, 30
 Fiddler Options, 84
 Find Sessions, 68
 HOSTS, 72
 Session Clipboards, 70
Windows 8 Metro
 capturing, 98
Windows Phone
 HTTPS Decryption, 121
Windows RT, 103

Z

Zopfli

 Transformer option, 153
 used by PNGDistill, 145