



# VisualEther Protocol Analyzer 1.0

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	VisualEther Protocol Analyzer	1
1.2	Benefits	1
<b>2</b>	<b>Getting Started</b>	<b>3</b>
2.1	Exporting to PDML from Ethereal	3
2.2	Generating Sequence Diagrams	3
2.3	Exploring the Field Extraction Template (FXT)	4
2.3.1	Basic Structure	6
2.3.2	Message Template	6
2.3.3	Opcode Extraction Template	9
2.3.4	Parameter Extraction Template	11
2.4	All.fxt.xml	13
<b>3</b>	<b>FXT Reference</b>	<b>14</b>
3.1	Message Templates	14
3.1.1	<tcp-message>	14
3.1.2	<udp-message>	15
3.1.3	<sctp-message>	16
3.1.4	<ip-message>	17
3.1.5	<message>	18
3.2	Field Templates	19
3.2.1	<opcode>	19
3.2.2	<param>	21
3.2.3	<source>	23
3.2.4	<destination>	25
3.2.5	<remark>	26
<b>4</b>	<b>VisualEther Options</b>	<b>28</b>
4.1	VisualEther Options Dialog Box	28
4.2	VisualEther Options	29



# 1 Introduction

## 1.1 VisualEther Protocol Analyzer

VisualEther is a system design reverse engineering and visual protocol analysis tool. VisualEther parses Ethereal logs and generates sequence diagrams and collaboration diagrams.

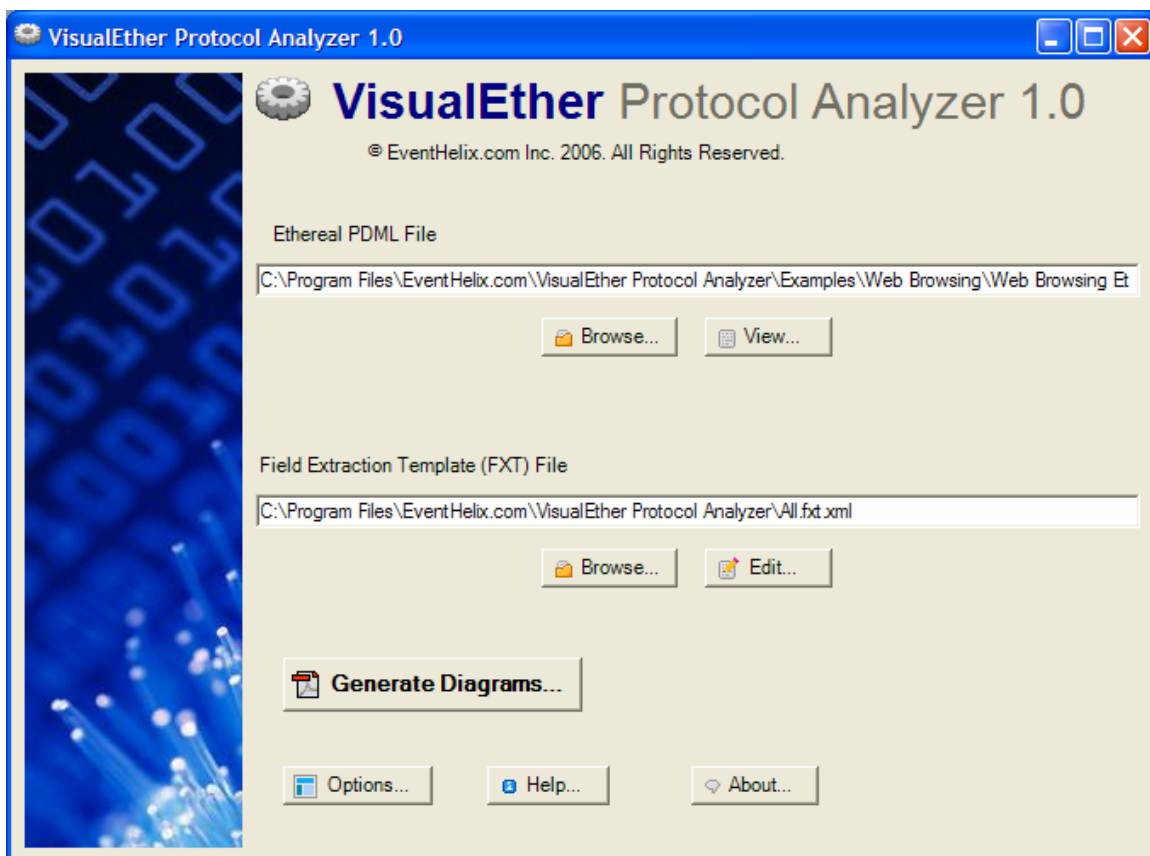


Figure 1-1 VisualEther Main Screen

## 1.2 Benefits

### Visually debug protocol interactions

- Generate Sequence Diagrams from the Ethereal output. The sequence diagrams provide a visual trace of the packet flow between the different nodes.
- VisualEther generates a context diagram of the packet exchange between different nodes. This provides a birds-eye-view of all the interactions

### **Auto Summarize Ethereal Output**

in the log.

- The packets and fields to be included in the generated diagrams are defined by a simple template definition.
- You can completely customize the template to select the message fields that need to be represented in the generated diagrams.

### **Post Process Diagrams to Reverse Engineer System Design**

- Use VisualEther to better understand complex interactions that take place in the system.
- VisualEther can be used to generate design documentation from Ethereal logs.
- The diagrams are generated using EventStudio System Designer, so you can easily edit the output files to reverse engineer the design documentation..



## 2 Getting Started

### 2.1 Exporting to PDML from Ethereal

VisualEther requires the Ethereal logs to be exported to the PDML XML format. This requires the following steps:

1. Invoke Ethereal.
2. Use the "File > Open" command to open the example Ethereal log "Web Browsing Ethereal Capture.bin" located in the "Examples\Web Browsing" directory in the VisualEther installation.
3. Now invoke the "File > Export > as XML PDML (packet details file)" Ethereal menu.
4. Name this file as "Web Browsing Ethereal Capture.pdml.xml". (Note that the file should be given a **.pdml.xml** extension.)

### 2.2 Generating Sequence Diagrams

The next step is to generate Sequence Diagrams output from the PDML file:

1. Double click on the VisualEther icon.
2. Select the Ethereal PDML file that was generated in section 2.1.
  - a. Click the "Browse..." button for the Ethereal PDML file.
  - b. VisualEther will display a file selection dialog box.
  - c. Change directory to "**Examples\Web Browsing**" in the VisualEther installation directory.  
("c:\Program Files\EventHelix.com\VisualEther Protocol Analyzer\Examples\Web Browsing" is the path for a default installation.)
  - d. Select the "**Web Browsing Ethereal Capture.pdml.xml**" file from the list (note that the PDML files have a **.pdml.xml** file extension).
3. Now select the Field Extraction Template (FXT) file to specify the fields that need to be included in the diagram.
  - a. Click the "Browse..." button for the FXT file.
  - b. VisualEther will display a file selection dialog box.
  - c. Change directory to "**Examples\Web Browsing**" in the VisualEther installation directory.
  - d. Select the "**Web Browsing.fxt.xml**" file from the list (note that the FXT files have a **.fxt.xml** file extension).
4. Click on the "Generate Documents" button.
5. VisualEther will perform the following steps:
  - a. Extract the fields from the PDML file based on the FXT file template.
  - b. Generate the FDL and SCN files needed by the EventStudio System Designer.
  - c. Invoke the EventStudio System Designer to generate the sequence diagram.
  - d. Invoke Adobe Acrobat Reader to open the sequence diagram.

The above example covered sequence diagram generation for web browsing. You can generate sequence diagrams for pretty much any protocol. All you need is an **.fxt.xml** file with the appropriate extraction templates. For templates for a large number of protocols, refer to section 2.4 titled **All.fxt.xml**.

### 2.3 Exploring the Field Extraction Template (FXT)

VisualEther uses FXT files to define the contents of the generated sequence diagrams. The FXT file specifies the fields in the Ethereal capture that should be included in the generated sequence diagram.

A typical FXT template file is shown in Figure 2-1.

```

<?xml version="1.0" encoding="utf-8"?>

<!--
Description:
  Template for documenting Web Browsing interactions with VisualEther.

Protocols:
  DNS: Domain Name System
  HTTP: Hypertext Transfer Protocol

Copyright © EventHelix.com Inc. 2006. All Rights Reserved.
-->

<FXT xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="http://www.eventhelix.com/Schemas/FXT_0_3.xsd">

  <!-- Message Template for Domain Name System (DNS) Extraction -->
  <udp-message>
    <opcode>dns</opcode>
    <param>dns.flags</param>
    <param>dns.qry.name</param>
    <param>dns.qry.type</param>
    <param>dns.qry.class</param>
    <param>dns.resp.name</param>
    <param>dns.resp.type</param>
    <param>dns.resp.class</param>
    <param>dns.resp.ttl</param>
    <param>dns.Addr</param>
  </udp-message>

  <!-- Message Template for Hypertext Transfer Protocol (HTTP) Request Extraction -->
  <tcp-message>
    <opcode>http.request.method</opcode>
    <param>http.request.uri</param>
    <param>http.request.version</param>
    <param>http.response.code</param>
    <param>http.If-Modified-Since</param>
    <param>tcp.len</param>
  </tcp-message>

  <!-- Message Template for Hypertext Transfer Protocol (HTTP) Response Extraction -->
  <tcp-message>
    <opcode>http.response.code</opcode>
    <param>http.request.uri</param>
    <param>http.request.version</param>
    <param>tcp.len</param>
  </tcp-message>

  <!-- Default Message Template for Hypertext Transfer Protocol (HTTP) -->
  <tcp-message>
    <opcode>http</opcode>
    <param>http.request.uri</param>
    <param>http.request.version</param>
    <param>http.response.code</param>
    <param>tcp.len</param>
  </tcp-message>

</FXT>

```

Figure 2-1 Field Extraction Template (FXT) Example

### 2.3.1 Basic Structure

The basic structure of the Field Extraction Template (FXT) file is shown below.

- The file begins with a standard **xml** starting tag.
- Following XML syntax, all elements are enclosed in a "begin tag" and an "end tag". (**<begin-tag>** **</end-tag>**)
- **<fxt>** is the root element; the element also specifies the location of the schema file ([http://www.eventhelix.com/Schemas/FXT\\_0\\_3.xsd](http://www.eventhelix.com/Schemas/FXT_0_3.xsd)). The schema file defines the validation rules for the FXT format.
- The root element contains pattern matching templates enclosed by the **<message>**, **<ip-message>**, **<udp-message>**, **<tcp-message>** or **<sctp-message>** element. Each template definition specifies the protocol fields that need to be extracted. Typically you would define a pattern definition for every protocol needed in the final diagram.

```
<?xml version="1.0" encoding="utf-8"?>

<FXT xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="http://www.eventhelix.com/Schemas/FXT_0_3.xsd">

  <!-- Message Template for Domain Name System (DNS) Extraction -->
  <udp-message>

  </udp-message>

  <!-- Message Template for Hypertext Transfer Protocol (HTTP) Request Extraction -->
  <tcp-message>

  </tcp-message>

  <!-- Message Template for Hypertext Transfer Protocol (HTTP) Response Extraction -->
  <tcp-message>

  </tcp-message>

  <!-- Default Message Template for Hypertext Transfer Protocol (HTTP) -->
  <tcp-message>

  </tcp-message>

</FXT>
```

Figure 2-2 Basic Structure of a Field ExtractionTemplate (FXT) File

### 2.3.2 Message Template

A typical message template is shown in the following figure. The template contains specification for extraction of various components of a captured packet. The following field extraction templates are commonly used:

<b>&lt;opcode&gt;</b>	The opcode extraction template is used to extract the packet name in the sequence diagram. (Details in section 2.3.3)
<b>&lt;param&gt;</b>	Message parameters are extracted by the parameter extraction template. (Details in section 2.3.4)

These extraction templates will be discussed in detail in the following sections.

```
<!-- Message Template for Domain Name System (DNS) Extraction -->
<udp-message>
  <opcode>dns</opcode>
  <param>dns.flags</param>
  <param>dns.qry.name</param>
  <param>dns.qry.type</param>
  <param>dns.qry.class</param>
  <param>dns.resp.name</param>
  <param>dns.resp.type</param>
  <param>dns.resp.class</param>
  <param>dns.resp.ttl</param>
  <param>dns.Addr</param>
</udp-message>
```

Figure 2-3 Message Template

The pattern matching example will focus on a DNS Query packet that is represented in PDML in the following figure. The figure shows all the interesting fields in color, rest of the fields have been grayed out.

```

<packet>
  <proto name="geninfo" pos="0" showname="General information" size="143">
    <field name="timestamp" pos="0" show="Apr 20, 2005 06:13:42.118718000"
      showname="Captured Time" value="1113992022.118718000" size="143"/>
  </proto>
  <proto name="frame" showname="Frame 4 (143 bytes on wire, 143 bytes captured)"
  size="143" pos="0">
  </proto>
  <proto name="eth" showname="Ethernet II, Src: 00:80:ae:cf:6f:dc, Dst:
  08:00:46:43:79:cf"
  size="14" pos="0">
  </proto>
  <proto name="ip" showname="Internet Protocol, Src Addr: 66.82.4.8 (66.82.4.8), Dst
  Addr: 192.168.0.2 (192.168.0.2)" size="20" pos="14">
    <field name="ip.src" showname="Source: 66.82.4.8 (66.82.4.8)" size="4" pos="26"
    show="66.82.4.8" value="42520408"/>
    <field name="ip.dst" showname="Destination: 192.168.0.2 (192.168.0.2)" size="4"
    pos="30" show="192.168.0.2" value="c0a80002"/>
  </proto>
  <proto name="udp" showname="User Datagram Protocol, Src Port: domain (53), Dst Port:
  1035 (1035)" size="8" pos="34">
    <field name="udp.srcport" showname="Source port: domain (53)" size="2" pos="34"
    show="53" value="0035"/>
    <field name="udp.dstport" showname="Destination port: 1035 (1035)" size="2" pos="36"
    show="1035" value="040b"/>
  </proto>
  <proto name="dns" showname="Domain Name System (response)" size="101" pos="42">
    <field name="dns.id" showname="Transaction ID: 0x82a3" size="2" pos="42"
    show="0x82a3" value="82a3"/>
    <field name="dns.flags" showname="Flags: 0x8180 (Standard query response, No error)"
    size="2" pos="44" show="0x8180" value="8180"/>
  </field>
    <field show="Queries" size="24" pos="54"
    value="037777770a6576656e7468656c697803636f6d0000010001">
      <field show="www.eventhelix.com: type A, class IN" size="24" pos="54"
      value="037777770a6576656e7468656c697803636f6d0000010001">
        <field name="dns.qry.name" showname="Name: www.eventhelix.com" size="20" pos="54"
        show="www.eventhelix.com" value="037777770a6576656e7468656c697803636f6d00"/>
        <field name="dns.qry.type" showname="Type: A (Host address)" size="2" pos="74"
        show="0x0001" value="0001"/>
        <field name="dns.qry.class" showname="Class: IN (0x0001)" size="2" pos="76"
        show="0x0001" value="0001"/>
      </field>
    </field>
    <field show="Answers" size="16" pos="78" value="c00c000100010001518000043f63d153">
      <field show="www.eventhelix.com: type A, class IN, addr 63.99.209.83" size="16"
      pos="78" value="c00c000100010001518000043f63d153">
        <field name="dns.resp.name" showname="Name: www.eventhelix.com" size="2" pos="78"
        show="www.eventhelix.com" value="c00c"/>
        <field name="dns.resp.type" showname="Type: A (Host address)" size="2" pos="80"
        show="0x0001" value="0001"/>
        <field name="dns.resp.class" showname="Class: IN (0x0001)" size="2" pos="82"
        show="0x0001" value="0001"/>
        <field name="dns.resp.ttl" showname="Time to live: 1 day" size="4" pos="84"
        show="86400" value="00015180"/>
        <field name="dns.resp.len" showname="Data length: 4" size="2" pos="88" show="4"
        value="0004"/>
        <field show="Addr: 63.99.209.83" size="4" pos="90" value="3f63d153"/>
      </field>
    </field>
  </proto>
</packet>

```

Figure 2-4 Packet Representation in PDML

### 2.3.3 Opcode Extraction Template

The opcode field defines the packet name in the generated sequence diagrams. The opcode extraction template definition for the DNS protocol is shown below.

```
<udp-message>
  <!--Pattern Matching Template -->
  <opcode>dns</opcode>

  <!-- Other Extraction Templates -->
</udp-message>
```

Figure 2-5 Opcode Extraction Template Shown in a UDP Message Template

The figure below shows an example of an opcode extracted by the <opcode> extraction template.

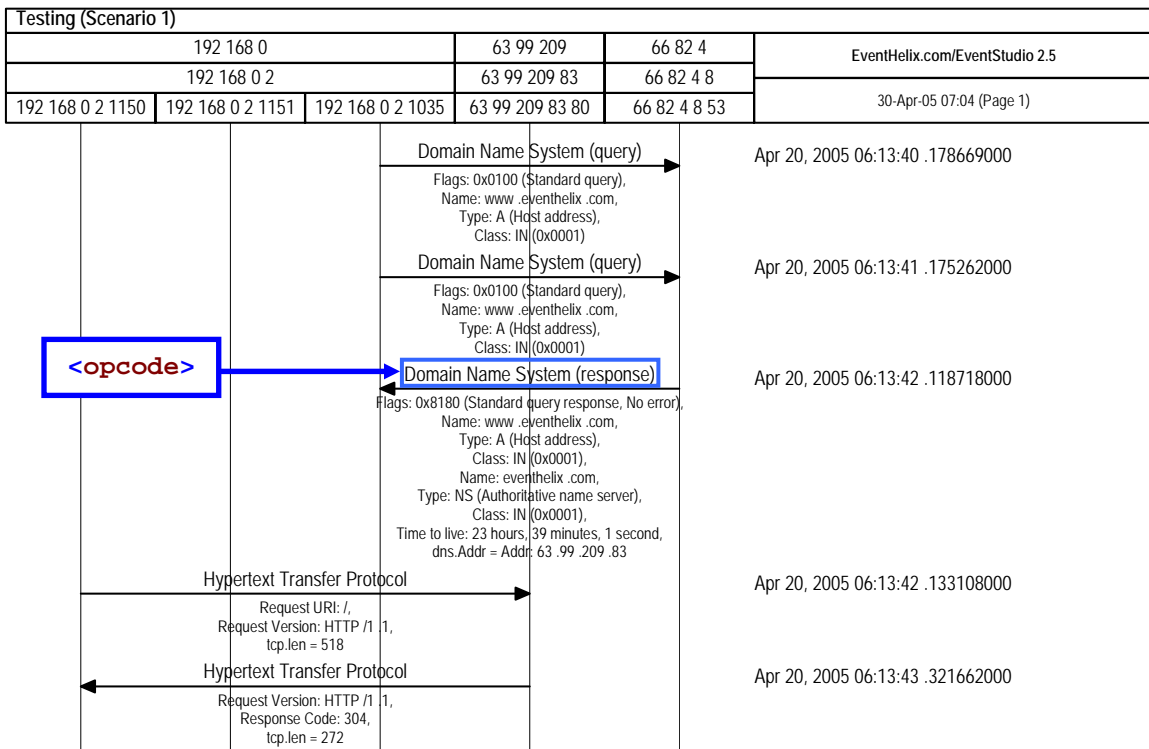


Figure 2-6 Extracted Opcode Shown in a Sequence Diagram

The selected field has been picked from the highlighted PDML entry shown in Figure 2-7. The field name specified in the extraction template (<opcode>dns</opcode>) is matched with the name attribute specified in the PDML file (<proto name="dns"....).

Note that VisualEther picks the **showname** attribute's value for representing the opcode.<sup>1</sup>

<sup>1</sup> By default VisualEther picks the entry defined by the **showname** attribute. The **show** attribute can be selected by using the **display="brief"** attribute in the extraction template.

```
<packet>

  <proto name="geninfo" pos="0" showname="General information" size="143">
  </proto>

  <proto name="frame" showname="Frame 4 (143 bytes on wire, 143 bytes captured)"
  </proto>

  <proto name="ip" showname="Internet Protocol, Src Addr: 66.82.4.8 (66.82.4.8), Dst
  Addr: 192.168.0.2 (192.168.0.2)" size="20" pos="14">
  </proto>

  <proto name="udp" showname="User Datagram Protocol, Src Port: domain (53), Dst Port:
  1035 (1035)" size="8" pos="34">
  </proto>

  <proto name="dns" showname="Domain Name System (response)" size="101"
  pos="42">
    <field name="dns.id" showname="Transaction ID: 0x82a3" size="2" pos="42"
  show="0x82a3" value="82a3"/>
  </proto>
</packet>
```

Figure 2-7 PDML Field Selected for Opcode Extraction

### 2.3.4 Parameter Extraction Template

The parameter extraction template is used to display addition attributes of the packet. Packet parameters are represented below the arrow in a sequence diagram. The parameter contents are extracted using the `<param>` extraction template. Extracted representation of two such parameters is shown in the following figure.

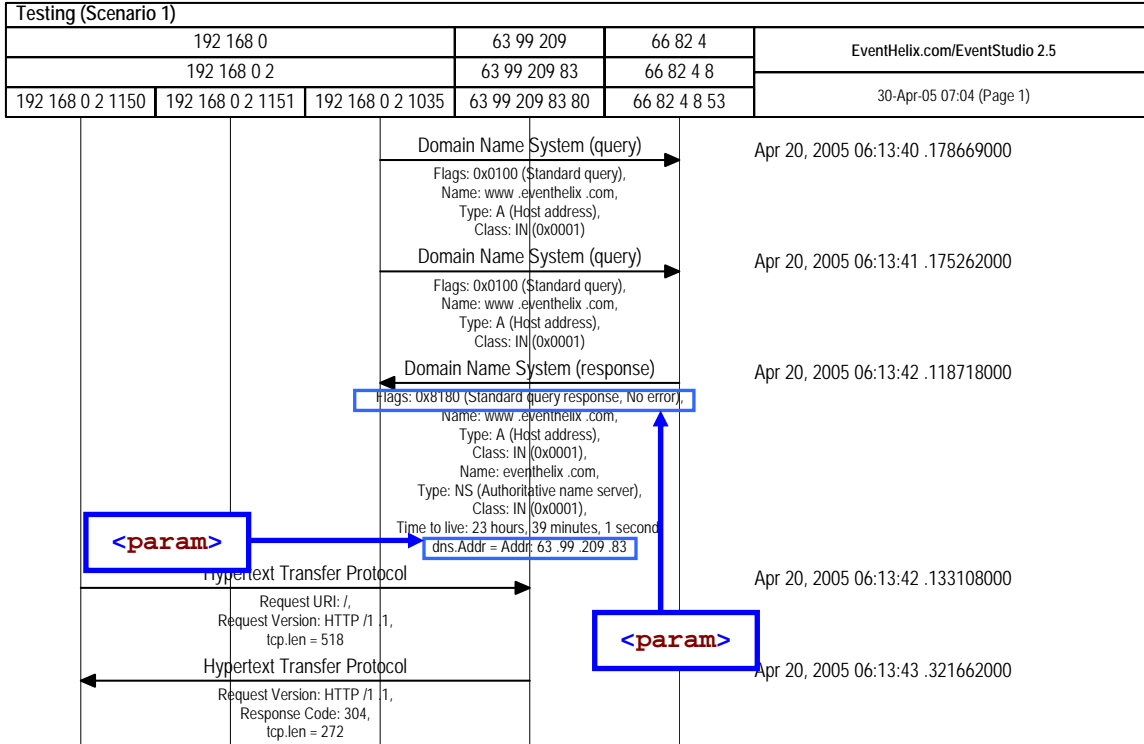


Figure 2-8 Some of the Extracted Parameters Shown in a Sequence Diagram

Here we will have a closer look at the extraction of the first and the last parameters of the "Domain Name System (response)" packet. The template for these parameters is shown below.

```

<udp-message>
  <!-- Opcode Extraction Template -->
  <opcode>dns</opcode>

  <!-- Parameter Extraction Template -->
  <param>dns.flags</param>

  <!-- More Parameters -->
  <param>dns.Addr</param>

  <!-- More Extraction Templates -->
</udp-message>
    
```

Figure 2-9 Parameter Extraction Template Shown in a Message Template

The first highlighted parameter in Figure 2-9 extracts the value of `dns.flags`. The `showname` attribute is extracted from a field with the name `"dns.flags"`. (See Figure 2-10)

The second highlighted parameter in Figure 2-9 really defines a special case. Here Ethereal has not included a name attribute in the field element (See the second highlighted field in Figure

2-10). Such fields are addressed by internally naming the field. In this case the field has been named `dns.Addr`. The name has been derived from the name of the enclosing `<proto>` attribute (`dns`) and the first part of the `show` string (`Addr` - The part before the colon).

```

<proto name="dns" showname="Domain Name System (response)" size="101" pos="42">
  <field name="dns.id" showname="Transaction ID: 0x82a3" size="2" pos="42"
  show="0x82a3" value="82a3"/>
  <field name="dns.flags" showname="Flags: 0x8180 (Standard query
  response, No error)" size="2" pos="44" show="0x8180" value="8180">
  </field>
  <field show="Queries" size="24" pos="54" >
    <field show="www.eventhelix.com: type A, class IN" size="24" pos="54">
      <field name="dns.qry.name" showname="Name: www.eventhelix.com" size="20" pos="54"
      show="www.eventhelix.com" value="037777770a6576656e7468656c697803636f6d00"/>
      <field name="dns.qry.type" showname="Type: A (Host address)" size="2" pos="74"
      show="0x0001" value="0001"/>
      <field name="dns.qry.class" showname="Class: IN (0x0001)" size="2" pos="76"
      show="0x0001" value="0001"/>
    </field>
  </field>
  <field show="Answers" size="16" pos="78" value="c00c000100010001518000043f63d153">
    <field show="www.eventhelix.com: type A, class IN, addr 63.99.209.83" size="16"
    pos="78" value="c00c000100010001518000043f63d153">
      <field name="dns.resp.name" showname="Name: www.eventhelix.com" size="2" pos="78"
      show="www.eventhelix.com" value="c00c"/>
      <field name="dns.resp.type" showname="Type: A (Host address)" size="2" pos="80"
      show="0x0001" value="0001"/>
      <field name="dns.resp.class" showname="Class: IN (0x0001)" size="2" pos="82"
      show="0x0001" value="0001"/>
      <field name="dns.resp.ttl" showname="Time to live: 1 day" size="4" pos="84"
      show="86400" value="00015180"/>
      <field name="dns.resp.len" showname="Data length: 4" size="2" pos="88" show="4"
      value="0004"/>
      <field show="Addr: 63.99.209.83" size="4" pos="90"
      value="3f63d153"/>
    </field>
  </field>
</proto>

```

Figure 2-10 PDML Fields Selected for Parameter Extraction

## 2.4 All.fxt.xml

VisualEther includes a predefined extraction file that supports a large number of protocols. The file can be found in the VisualEther installation directory.

("c:\Program Files\EventHelix.com\VisualEther Protocol Analyzer\All.fxt.xml" is the path for a default installation.)

The following protocols are covered in All.fxt.xml

SIP	Session Initiation Protocol
H.225	Narrowband visual telephone services
H.245	Negotiate channel use and capabilities
Q.931	Manage call setup and termination
RADIUS	Remote Authentication Dial In User Service
RTP	Real-time Protocol
RTCP	RTP Control Protocol
SNMP	Simple Network Management Protocol
NFS V3	Network File System (Version 3)
RPC	Remote Procedure Call
TCAP	Transaction Capabilities Application Part
SCTP	Stream Control Transport Protocol
POP3	Post Office Protocol (Version 3)
IGMP	Internet Group Management Protocol
ARP	Address Resolution Protocol
DNS	Domain Name System
FTP	File Transfer Protocol
HTTP	Hypertext Transfer Protocol
ICMP	Internet Control Message Protocol
NBNS	Net Bios Name Service
BOOTP	Bootstrap Protocol
OSPF	Open Shortest Path First Routing Protocol
BGP	Border Gateway Protocol
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
IP	Internetworking Protocol



## 3 FXT Reference

### 3.1 Message Templates

#### 3.1.1 <tcp-message>

The TCP message template is used to extract TCP messages from an Ethereal PDML file. You will be using this template to trap messages for protocols like HTTP, FTP, POP3 and Q.931.

The TCP message template can contain the following field templates:

<opcode>	Mandatory
<param>	Optional
<remark>	Optional

A TCP message template for extracting an HTTP message is shown below:

```
<tcp-message>
  <opcode>http.request.method</opcode>
  <param>http.request.uri</param>
  <param>http.request.version</param>
  <param>http.response.code</param>
  <param>http.If-Modified-Since</param>
</tcp-message>
```

Figure 3-1 TCP Message Template

**Note:** The TCP message template internally uses the source IP address and source TCP port number to identify the message source. The message destination is identified by the destination IP address and the destination TCP port.

### 3.1.2 <udp-message>

The UDP message template is used to extract UDP messages from an Ethereal PDML file. You will be using this template to trap messages from protocols like DNS, RTP, SIP and SNMP.

The UDP message template can contain the following field templates:

<opcode>	Mandatory
<param>	Optional
<remark>	Optional

A UDP message template for extracting an HTTP message is shown below:

```
<udp-message>
  <opcode display="brief">dns</opcode>
  <param>dns.flags</param>
  <param>dns.qry.name</param>
  <param>dns.qry.type</param>
  <param>dns.qry.class</param>
  <param>dns.resp.name</param>
  <param>dns.resp.type</param>
  <param>dns.resp.class</param>
  <param>dns.resp.ttl</param>
  <param display="brief">dns.Addr</param>
</udp-message>
```

Figure 3-2 UDP Message Template

**Note:** The UDP message template internally uses the source IP address and source UDP port number to identify the message source. The message destination is identified by the destination IP address and the destination UDP port.

### 3.1.3 <sctp-message>

VisualEther has direct support for SCTP. Stream Control Transmission Protocol (SCTP) is a new transport layer in the IP stack. SCTP is the preferred transport for signaling protocols over IP.

The SCTP message template can contain the following field templates:

<opcode>	Mandatory
<param>	Optional
<remark>	Optional

A TCAP message that uses the SCTP template is shown below:

```
<sctp-message>
  <opcode>tcap</opcode>
  <param>tcap.oid</param>
  <param>tcap.application_context_name</param>
  <param>tcap.otid</param>
  <param>tcap.msgtype</param>
  <param>tcap.len</param>
  <param>m2ua.message_type</param>
  <param>m2ua.message_class</param>
  <param>mtp3.network_indicator</param>
  <param>mtp3.service_indicator</param>
  <param>mtp3.dpc</param>
  <param>mtp3.opc</param>
  <param>mtp3.sls</param>
  <param>sccp.called.pc</param>
  <param>sccp.called.ssn</param>
  <param>sccp.calling.pc</param>
  <param>sccp.calling.ssn</param>
</sctp-message>
```

Figure 3-3 SCTP Message Template

**Note:** The SCTP message template internally uses the source IP address and source SCTP port number to identify the message source. The message destination is identified by the destination IP address and the destination SCTP port.

### 3.1.4 <ip-message>

The IP message template is used to extract IP messages from an Ethereal PDML file. Typically this template is used with protocols that do not use TCP, UDP or SCTP transport. ICMP is one such protocol.

The IP message template can contain the following field templates:

<opcode>	Mandatory
<param>	Optional
<remark>	Optional

The following example shows an IP message template being used to extract ICMP messages.

```
<ip-message>  
  <opcode>icmp.type</opcode>  
  <param>icmp.seq</param>  
</ip-message>
```

Figure 3-4 IP Message Template

**Note:** The IP message template internally uses the source IP address to identify the message source. The message destination is identified by the destination IP address.

### 3.1.5 <message>

The message template is used to extract messages where the source or destination of the message may not be represented by the IP address and port number in the message. This template is used to trap messages for protocols like ARP.

When the <message> template is used, the <source> and the <destination> field templates must be specified in the message. These field templates are required to identify the source and the destination of the message.

The message template can contain the following field templates:

<opcode>	Mandatory
<param>	Optional
<source>	Mandatory
<destination>	Mandatory
<remark>	Optional

An ARP message filter that uses the message template is shown below:

```
<message>
  <opcode display="brief">arp</opcode>

  <param display="brief">arp.src.hw_mac</param>
  <param display="brief">arp.src.proto_ipv4</param>
  <param display="brief">arp.dst.hw_mac</param>
  <param display="brief">arp.dst.proto_ipv4</param>
  <source>
    <ip>arp.src.proto_ipv4</ip>
    <port>arp.src.hw_mac</port>
  </source>
  <destination>
    <ip>arp.dst.proto_ipv4</ip>
    <port>arp.dst.hw_mac</port>
  </destination>
</message>
```

Figure 3-5 Message Template

### 3.2 Field Templates

#### 3.2.1 <opcode>

The opcode field defines the packet name in the generated sequence diagrams. The opcode extraction template definition for the DNS protocol is shown below.

```
<udp-message>
  <!--Pattern Matching Template -->
  <opcode>dns</opcode>

  <!-- Other Extraction Templates -->
</udp-message>
```

Figure 3-6 Opcode Extraction Template Shown in a UDP Message Template

The figure below shows an example of an opcode extracted by the <opcode> extraction template.

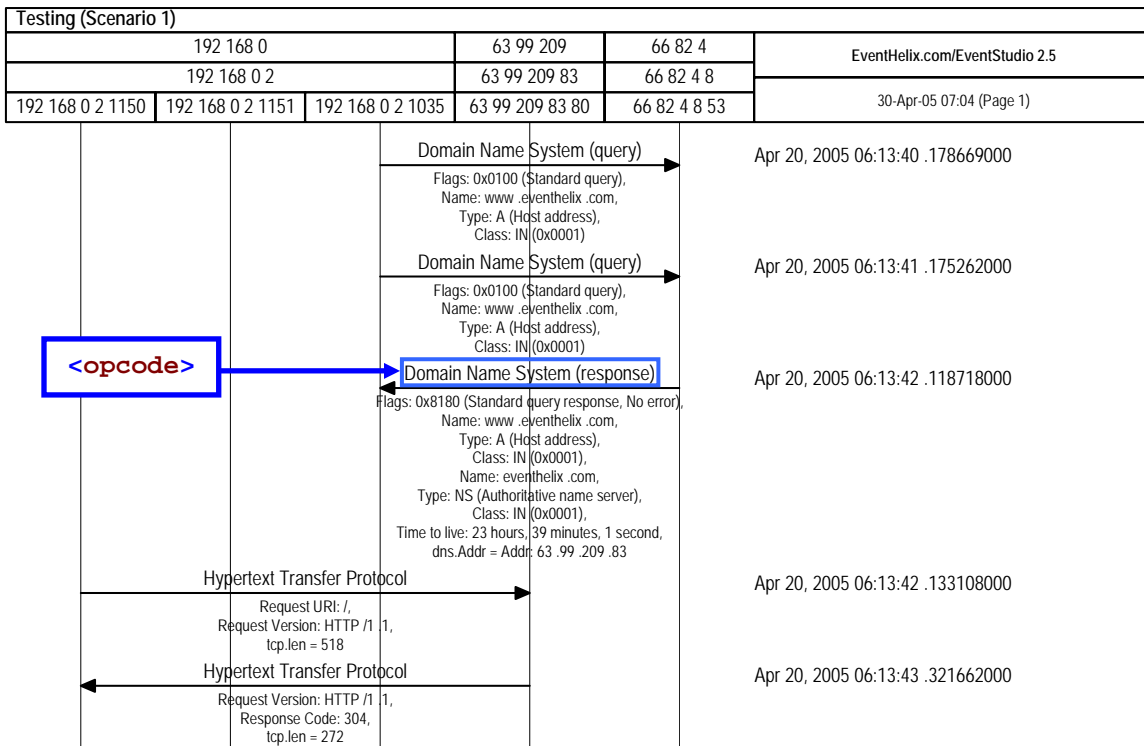


Figure 3-7 Extracted Opcode Shown in a Sequence Diagram

The selected field has been picked from the highlighted PDML entry shown below. The field name specified in the extraction template (<opcode>dns</opcode>) is matched with the name attribute specified in the PDML file (<proto name="dns"...).

Note that VisualEther picks the **showname** attribute's value for representing the opcode.<sup>2</sup>

<sup>2</sup> By default VisualEther picks the entry defined by the **showname** attribute. The **show** attribute can be selected by using the **display="brief"** attribute in the extraction template.

```
<packet>

  <proto name="geninfo" pos="0" showname="General information" size="143">
  </proto>

  <proto name="frame" showname="Frame 4 (143 bytes on wire, 143 bytes captured)"
  </proto>

  <proto name="ip" showname="Internet Protocol, Src Addr: 66.82.4.8 (66.82.4.8), Dst
  Addr: 192.168.0.2 (192.168.0.2)" size="20" pos="14">
  </proto>

  <proto name="udp" showname="User Datagram Protocol, Src Port: domain (53), Dst Port:
  1035 (1035)" size="8" pos="34">
  </proto>

  <proto name="dns" showname="Domain Name System (response)" size="101"
  pos="42">
    <field name="dns.id" showname="Transaction ID: 0x82a3" size="2" pos="42"
  show="0x82a3" value="82a3"/>
  </proto>
</packet>
```

Figure 3-8 PDML Field Selected for Opcode Extraction

### 3.2.2 <param>

The parameter extraction template is used to display additional attributes of the packet. Packet parameters are represented below the arrow in a sequence diagram. The parameter contents are extracted using the <param> extraction template. Extracted representation of two such parameters is shown in the following figure.

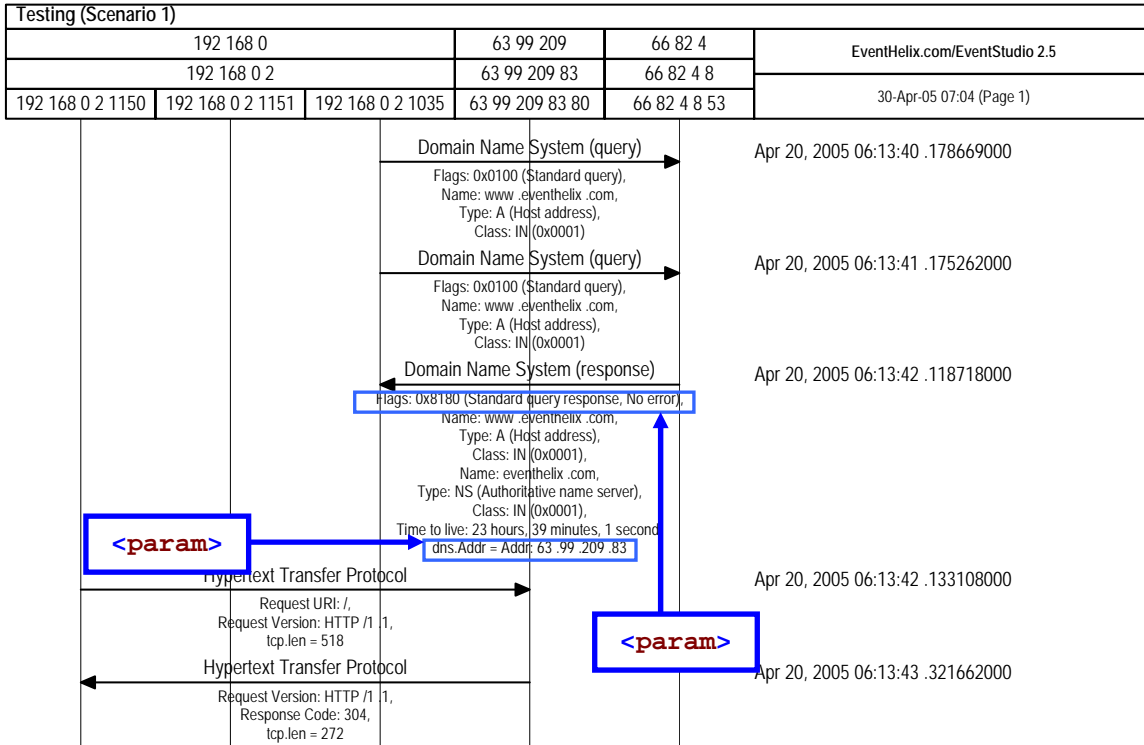


Figure 3-9 Some of the Extracted Parameters Shown in a Sequence Diagram

Here we will have a closer look at the extraction of the first and the last parameters of the "Domain Name System (response)" packet. The template for these parameters is shown below.

```

<udp-message>
  <!-- Opcode Extraction Template -->
  <opcode>dns</opcode>

  <!-- Parameter Extraction Template -->
  <param>dns.flags</param>

  <!-- More Parameters -->
  <param>dns.Addr</param>

  <!-- More Extraction Templates -->
</udp-message>
    
```

Figure 3-10 Parameter Extraction Template Shown in a Message Template

The first highlighted parameter in Figure 3-10 extracts the value of `dns.flags`. The `showname` attribute is extracted from a field with the name `"dns.flags"`. (See Figure 3-11)

The second highlighted parameter in Figure 3-10 really defines a special case. Here Ethereal has not included a name attribute in the field element (See the second highlighted field in Figure 3-11). Such fields are addressed by internally naming the field. In this case the field has been named `dns.Addr`. The name has been derived from the name of the enclosing `<proto>` attribute (`dns`) and the first part of the `show` string (`Addr` - The part before the colon).

```

<proto name="dns" showname="Domain Name System (response)" size="101" pos="42">
  <field name="dns.id" showname="Transaction ID: 0x82a3" size="2" pos="42"
show="0x82a3" value="82a3"/>
  <field name="dns.flags" showname="Flags: 0x8180 (Standard query
response, No error)" size="2" pos="44" show="0x8180" value="8180">
  </field>
  <field show="Queries" size="24" pos="54" >
    <field show="www.eventhelix.com: type A, class IN" size="24" pos="54">
      <field name="dns.qry.name" showname="Name: www.eventhelix.com" size="20" pos="54"
show="www.eventhelix.com" value="037777770a6576656e7468656c697803636f6d00"/>
      <field name="dns.qry.type" showname="Type: A (Host address)" size="2" pos="74"
show="0x0001" value="0001"/>
      <field name="dns.qry.class" showname="Class: IN (0x0001)" size="2" pos="76"
show="0x0001" value="0001"/>
    </field>
  </field>
  <field show="Answers" size="16" pos="78" value="c00c000100010001518000043f63d153">
    <field show="www.eventhelix.com: type A, class IN, addr 63.99.209.83" size="16"
pos="78" value="c00c000100010001518000043f63d153">
      <field name="dns.resp.name" showname="Name: www.eventhelix.com" size="2" pos="78"
show="www.eventhelix.com" value="c00c"/>
      <field name="dns.resp.type" showname="Type: A (Host address)" size="2" pos="80"
show="0x0001" value="0001"/>
      <field name="dns.resp.class" showname="Class: IN (0x0001)" size="2" pos="82"
show="0x0001" value="0001"/>
      <field name="dns.resp.ttl" showname="Time to live: 1 day" size="4" pos="84"
show="86400" value="00015180"/>
      <field name="dns.resp.len" showname="Data length: 4" size="2" pos="88" show="4"
value="0004"/>
      <field show="Addr: 63.99.209.83" size="4" pos="90"
value="3f63d153"/>
    </field>
  </field>
</proto>

```

Figure 3-11 PDML Fields Selected for Parameter Extraction

**Note:** If a field with a specified name occurs multiple times in a packet, Visual Ether will include all occurrences in the generated sequence diagram.

### 3.2.3 <source>

Source extraction templates are used to identify the source of a message. The source of a message is represented as a column in the sequence diagrams. The source extraction template needs to be included only with the <message> template. The <tcp-message>, <udp-message>, <sctp-message> and <ip-message> templates already include an internally defined source extraction template.

VisualEther supports definition of two types of source extraction templates:

IP Address Extraction Templates	Only the source IP address of the packet is used to identify the source.
IP Address and Port Number Extraction Template	The source IP address and the port number are used to identify the source.

In this example we will be examining an “IP Address and Port Number Extraction Template”. We will be using the message template to identify the source of the message<sup>3</sup>.

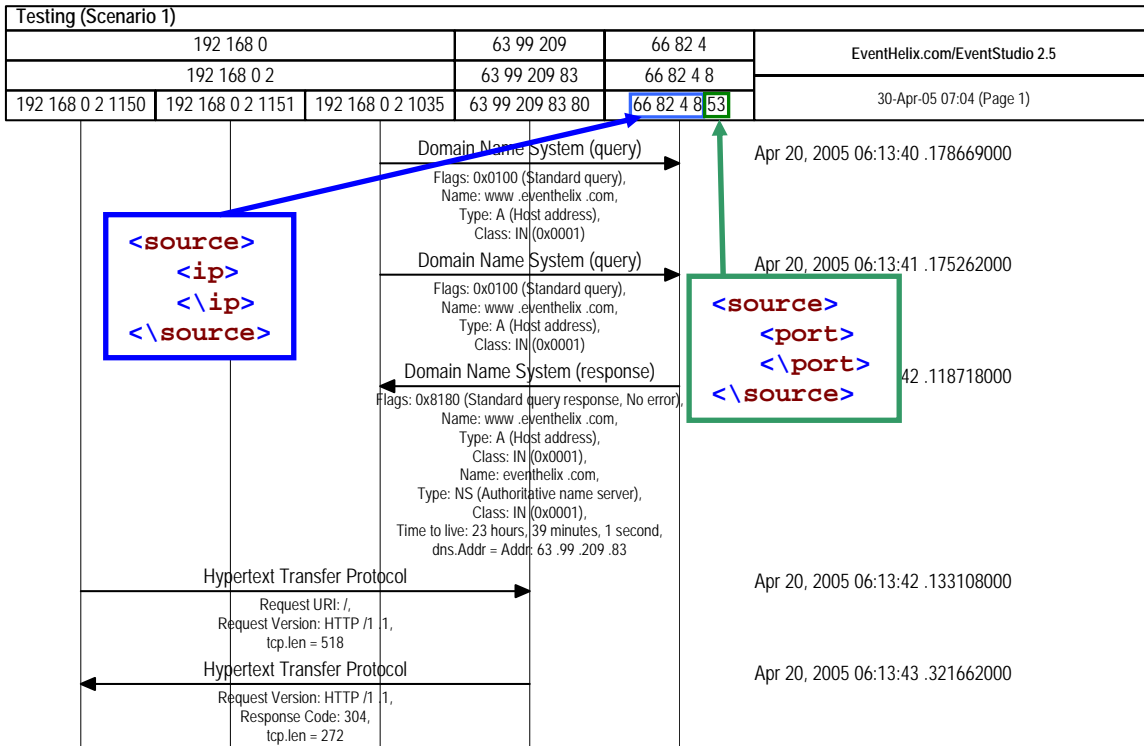


Figure 3-12 Source Extraction Template Shown in a Sequence Diagram

Now let's look into the template definition of these extraction templates.

<source>  
 <ip>  
 <\ip>  
 <\source>

The <ip> tag contained in the source extraction defines the field for extracting the source IP address of the packet. Figure 3-13 shows the source IP address for “Domain Name System (response)” packet.

<sup>3</sup> Typically you would use the UDP message extractor to extract UDP messages. The message template is used here just for illustration.

```

<source>
  <port>
    <\port>
<\source>

```

The `<port>` tag contained in the source extraction template defines the field for extracting the source UDP/TCP port number of the packet. Figure 3-13 shows the source UDP port number for "Domain Name System (response)" packet.

```

<message>
  <!-- Other Extractors -->

  <!-- Packet Source Extraction Templates -->
  <source>
    <ip>ip.src</ip>
    <port>udp.srcport</port>
  </source>

  <!-- Other Extraction Templates -->
</message>

```

Figure 3-13 Source Extraction Template Shown in a Message Template

The relevant sections of the PDML file are shown below:

```

<packet>

  <proto name="ip" showname="Internet Protocol, Src Addr: 66.82.4.8 (66.82.4.8), Dst
  Addr: 192.168.0.2 (192.168.0.2)" size="20" pos="14">
    <field name="ip.src" showname="Source: 66.82.4.8 (66.82.4.8)"
    size="4" pos="26" show="66.82.4.8" value="42520408"/>
    <field name="ip.dst" showname="Destination: 192.168.0.2 (192.168.0.2)" size="4"
    pos="30" show="192.168.0.2" value="c0a80002"/>
  </proto>

  <proto name="udp" showname="User Datagram Protocol, Src Port: domain (53), Dst Port:
  1035 (1035)" size="8" pos="34">
    <field name="udp.srcport" showname="Source port: domain (53)"
    size="2" pos="34" show="53" value="0035"/>
    <field name="udp.dstport" showname="Destination port: 1035 (1035)" size="2" pos="36"
    show="1035" value="040b"/>
  </proto>

  <proto name="dns" showname="Domain Name System (response)" size="101" pos="42">
  </proto>
</packet>

```

Figure 3-14 PDML Fields Selected by Source Extraction Template

### 3.2.4 <destination>

A destination extraction template is used to extract the destination address of a message. Destination extraction templates are very similar in usage to the source extraction templates.

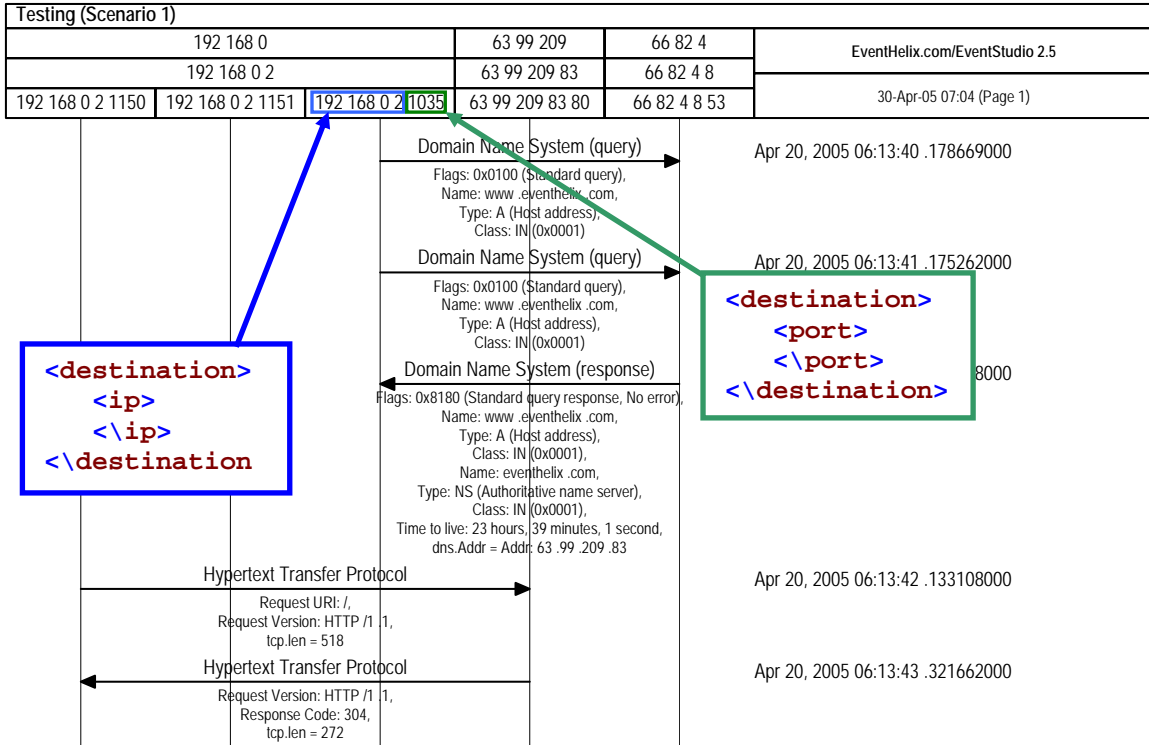


Figure 3-15 Destination Extraction Template Shown in a Sequence Diagram

### 3.2.5 <remark>

The remark extraction template specifies the field to be used for the remark used on the right side of the sequence diagrams. The figure below shows an example of a remark extracted using the remark extraction template.

The remark extraction template is optional. The template defaults to the packet timestamp if the message template has not been specified.

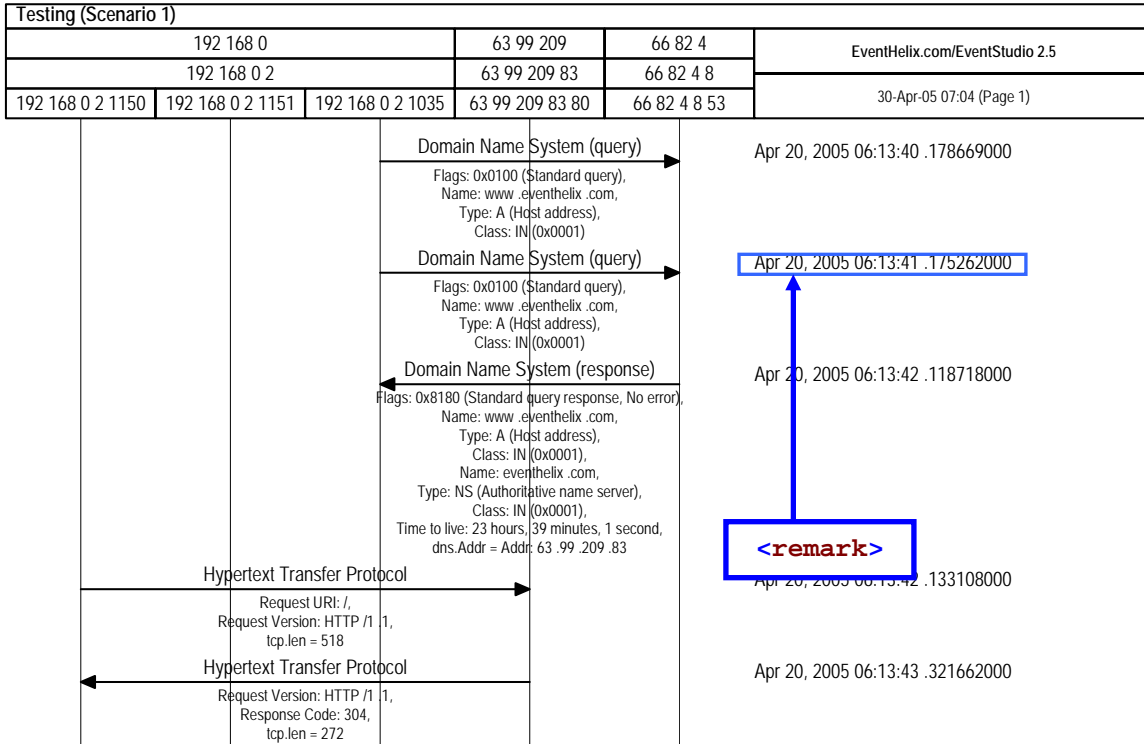


Figure 3-16 Remark Extraction Template Shown in a Sequence Diagram

Figure 3-17 shows the definition of a remark extraction template. In this case, the remark extraction template has been defined to extract the timestamp field from the PDML file.<sup>4</sup>

```

<udp-message>
  <!-- Other Field Extraction Templates -->

  <!-- Remark Extraction Template -->
  <remark>timestamp</remark>
</udp-message>
    
```

Figure 3-17 Remark Extraction Template Shown in a Packet Template

<sup>4</sup> The remark template specifying the timestamp is used here for illustration. If just the timestamp is desired, there is no need to include the remark definition template as it is included by default.

The relevant part of the PDML file is shown below:

```
<packet>
  <proto name="geninfo" pos="0" showname="General information" size="143">
    <field name="timestamp" pos="0" show="Apr 20, 2005
06:13:42.118718000"
      showname="Captured Time" value="1113992022.118718000"
size="143"/>
  </proto>
</packet>
```

Figure 3-18 PDML Field Used by the Remark Extraction Template



## 4 VisualEther Options

### 4.1 VisualEther Options Dialog Box

All configurable options in VisualEther are accessible from the VisualEther options dialog box. The dialog box is shown below.

This dialog lets you specify the:

- EventStudio path
- XML Editor path
- Action to be taken after VisualEther has completed diagram generation.
- The level of detail in the generated documents.
- Right hand side remark contents

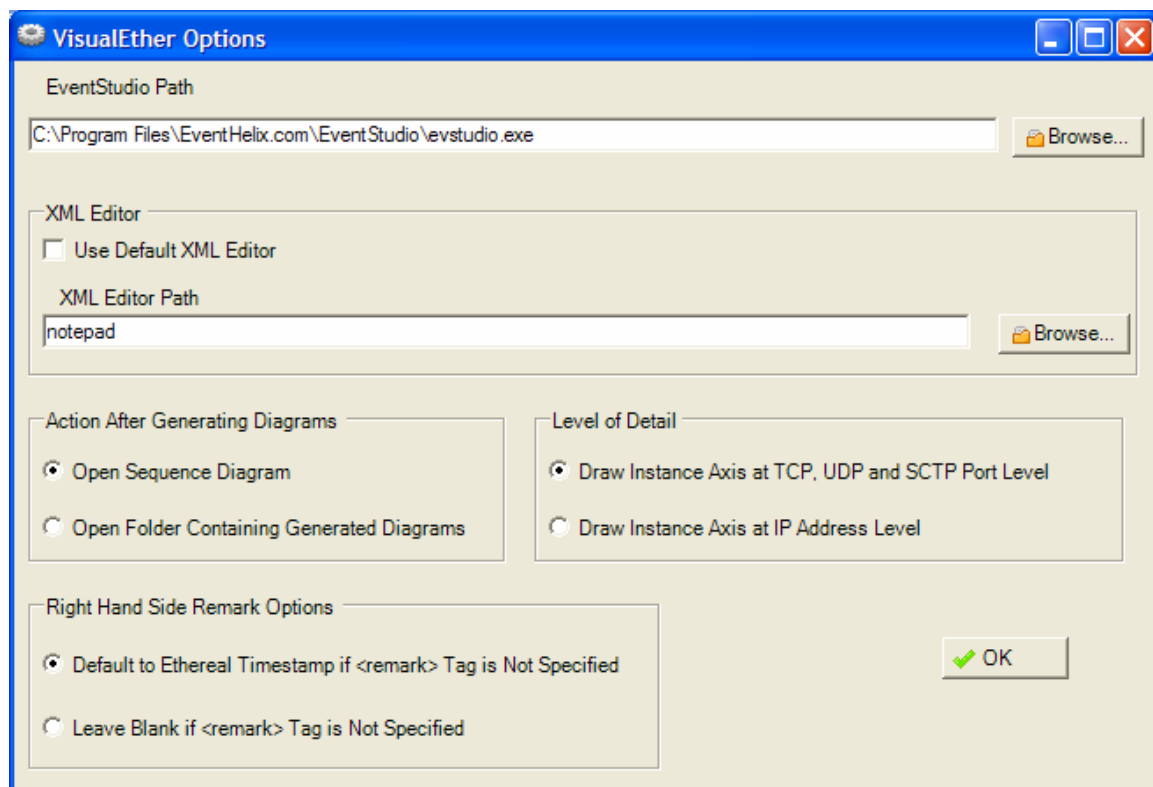


Figure 4-1 VisualEther Options Dialog Box

## 4.2 VisualEther Options

<b>EventStudio Path</b>	<p>VisualEther generates protocol sequence diagrams using the EventStudio System Designer's diagram rendering engine.</p> <p>On installation, VisualEther sets up the default EventStudio path to: C:\Program Files\EventHelix.com\EventStudio.</p> <p>The path needs to be changed if EventStudio was installed in a different directory.</p>
<b>XML Editor</b>	<p>VisualEther deals with XML files with .pdml.xml and .fxt.xml file extensions. The XML editor needed for editing these files can be configured here. The editor choices are:</p> <ul style="list-style-type: none"> <li>• Check the "Default XML Editor" checkbox if you wish to use the XML editor that Windows uses by when you double click on an XML file.</li> <li>• If the "Default XML Editor" checkbox is left unchecked, the path of the XML editor may be specified here.</li> </ul> <p>Note that VisualEther defaults to notepad as the editor of XML files. Use the options specified here to change that.</p>
<b>Action After Generating Diagrams</b>	<p>Once VisualEther finishes document generation it can be configured to either:</p> <ul style="list-style-type: none"> <li>• Directly open the generated sequence diagram; or</li> <li>• Open the folder containing the generated sequence diagrams.</li> </ul>
<b>Level of Detail</b>	<p>Use this option to control the level of detail in the generated documents. You can choose between:</p> <ul style="list-style-type: none"> <li>• Generating the diagram at TCP, UDP and SCTP port level. Each port is represented by a separate instance axis.</li> <li>• Generate the diagram at IP address level, regardless of the port being used.</li> </ul>
<b>Right Hand Side Remark Options</b>	<p>The &lt;remark&gt; tag is an optional tag in the FXT file. This option specifies the action to be taken if a &lt;remark&gt; tag is not specified in the matching template. The choices are:</p> <ul style="list-style-type: none"> <li>• Include the Ethernet timestamp</li> <li>• Leave blank</li> </ul>