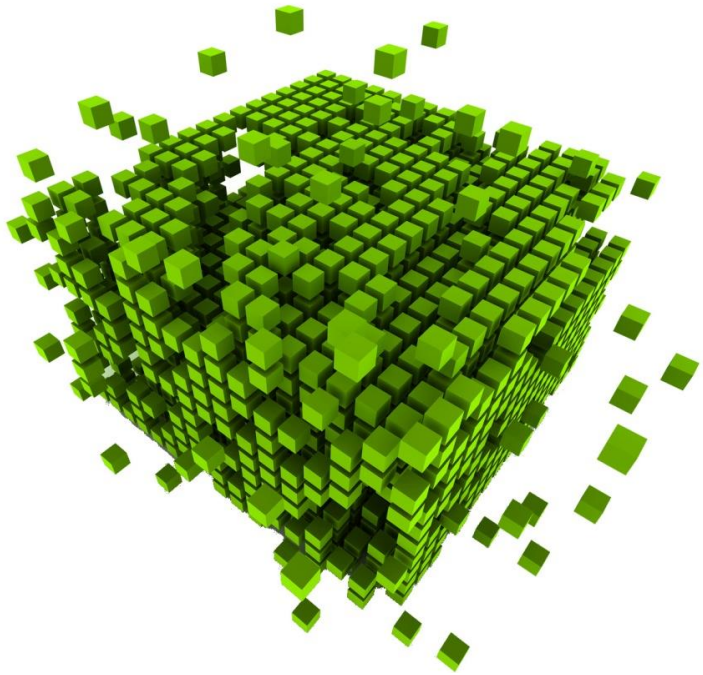
Apica
http://www.apicasystem.com

ZebraTester

Handling of "Dynamically-Exchanged Session Parameters"

English Edition



© 2011 -2019

Table of Contents

1 Overview....3
1.1 What are "dynamically-exchanged session parameters"?....3
1.2 Why "dynamically-exchanged session parameters" require manual configuration in the ZebraTester GUI?....3
1.3 How does the manual configuration work?....4
2 Illustrated Example....5
3 Other Ways to Extract and Assign Variables - Structured Data Access....24
4 Appendix A: Inner Working of the Text-Token-Based Algorithm (Var Extractor Wizard)....30
5 Manufacturer....33

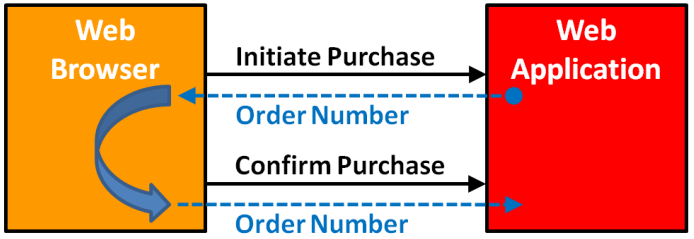
© 2011 - 2019 Apica All Rights Reserved

1 Overview

This document explains how the handling of dynamically-exchanged session parameters can be configured in the ZebraTester GUI.

1.1 What are "dynamically-exchanged session parameters"?

Some web applications use unique values to keep track of each individual session. These "session-context" values are usually dynamically generated CGI- or form-parameter that are exchanged between the web application and the web browser. If you would repeat the same web surfing session you would observe that the values of these session-tracking parameters are constantly updated by a more or less random algorithm. A good example could be a purchase order number in web shop application. If during a load test execution ZebraTester would simply echo the session-tracking values that were observed during the test recording stage, the load test run would fail.



The values of these dynamically-exchanged session parameters are always generated by the web server first and sent to the web browser. The web browser will then send these values back to the web server as part of future page requests. In other words: if a parameter value is not issued by the web server first, that value is NOT to be considered as a dynamically-exchanged session parameter for the scope of this documentation. For example, user names and passwords which are entered in login forms are NOT dynamically-exchanged session parameters.

In contrast to session cookies which are automatically handled by ZebraTester, "dynamically-exchanged session parameters" require some manual configuration to allow ZebraTester to locate and handle these parameters. ZebraTester's **Var Finder** menu will assist you with the task of **finding and handling these dynamically-exchanged session parameters in an intuitive and simple fashion.**

1.2 Why "dynamically-exchanged session parameters" require manual configuration in the ZebraTester GUI?

Correctly understanding and processing these dynamic session values is a requirement for ZebraTester to simulate hundreds or even thousands of web user sessions from only one computer system.

During the execution of a load test ZebraTester simulates only the strictly required functionality of a web browser. As an example the automated handling of session cookies occurs with no on-screen rendering and without relying on a JavaScript engine. In a nutshell ZebraTester simply simulates the network traffic that is usually generated by a web browser, without the need that for each simulated user to run a browser instance.

Doing so is **up to hundred times more efficient** from a CPU/memory requirement point of view then using real web browser instances. This allows ZebraTester to simulate hundreds or even thousands of web users from a single machine.

This is a technical solution that allows you the execution of high-performance load tests with minimal hardware resources.

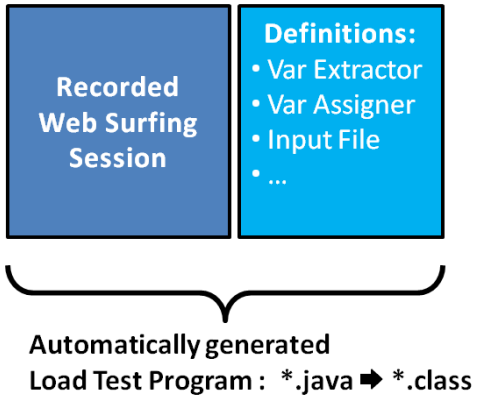
© 2011 - 2019 Apica All Rights Reserved

1.3 How does the manual configuration work?

The manual configuration of a "dynamically-exchanged session parameter" works in such a way that the parameter's value is extracted into a variable (Extract Var) when it is first received from the web application. The value of the variable is then sent back to the web application in subsequent requests (Assign Var).

During the manual configuration of this process - i.e. when extracting of a value into a variable, as well as when assigning a variable to a parameter - the ZebraTester GUI behaves as if the displayed value is already dynamically-exchanged. **In reality the GUI only creates a definition of how that specific value needs to be extracted to a variable and then assigned to subsequent requests.** These definitions are then incorporated in the automatically generated load test program and executed at run time during the load test for each simulated user. The definitions are also stored in the *.prxdat files as an integral addition to the recorded web surfing session. This means that the core of a recorded web surfing session is not modified by creating definitions and remains still "intact" in the original condition.

You can always delete any previously created definition, without the need to record the web surfing session again.



The usage "dynamically-exchanged session parameters" is heavily dependant on the technologies used by the application you are planning to test. Some web applications exclusively rely on cookies for session tracking purposes and don't require you to manually configure handling of session parameters. On the other hand there are web applications that heavily rely on these session parameters and even a simple application made up of a few web pages might use a number of "dynamically-exchanged session parameters". Most of the "dynamically-exchanged session parameters" are purely technical in nature and do not correspond directly to end-user data fields which are displayed in the web browser.

After recording a new web surfing session, you have always to check whether "dynamically-exchanged session parameters" are used or not used by the web application. This is can be done by calling the **Var Finder** menu.

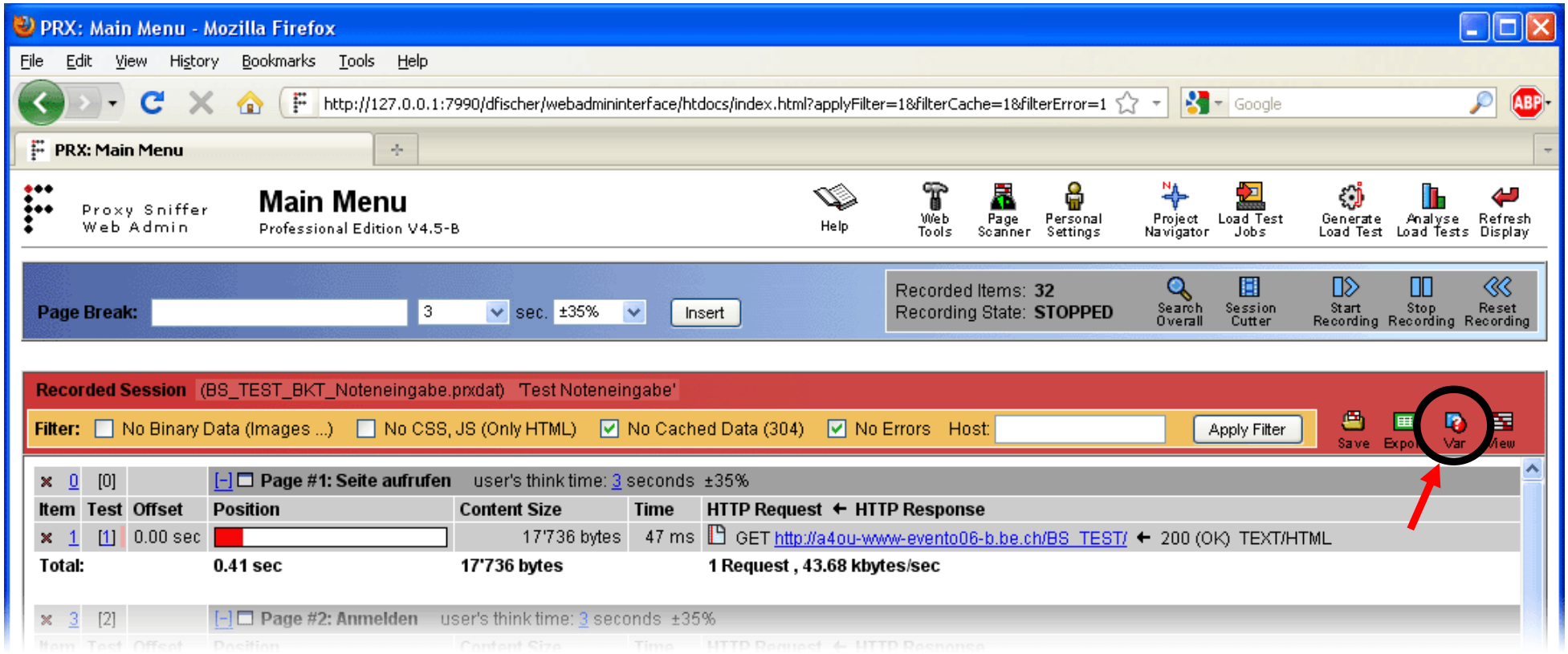
As you will see the following example, the ZebraTester GUI assists you with **powerful features** in order to quickly and easily handle "dynamically-exchanged session parameters".

© 2011 - 2019 Apica All Rights Reserved

2 Illustrated Example

Hint: Before you start the configuring dynamically-exchanged session parameters you should delete or filter-out all unwanted URL calls from your recorded web surfing session. For example, you should delete all calls to Google Analytics because you just want to stress your web application, not Google Analytics.

Click on the **Var Finder** menu after you have deleted or filtered-out all unwanted URL calls in Main Menu:

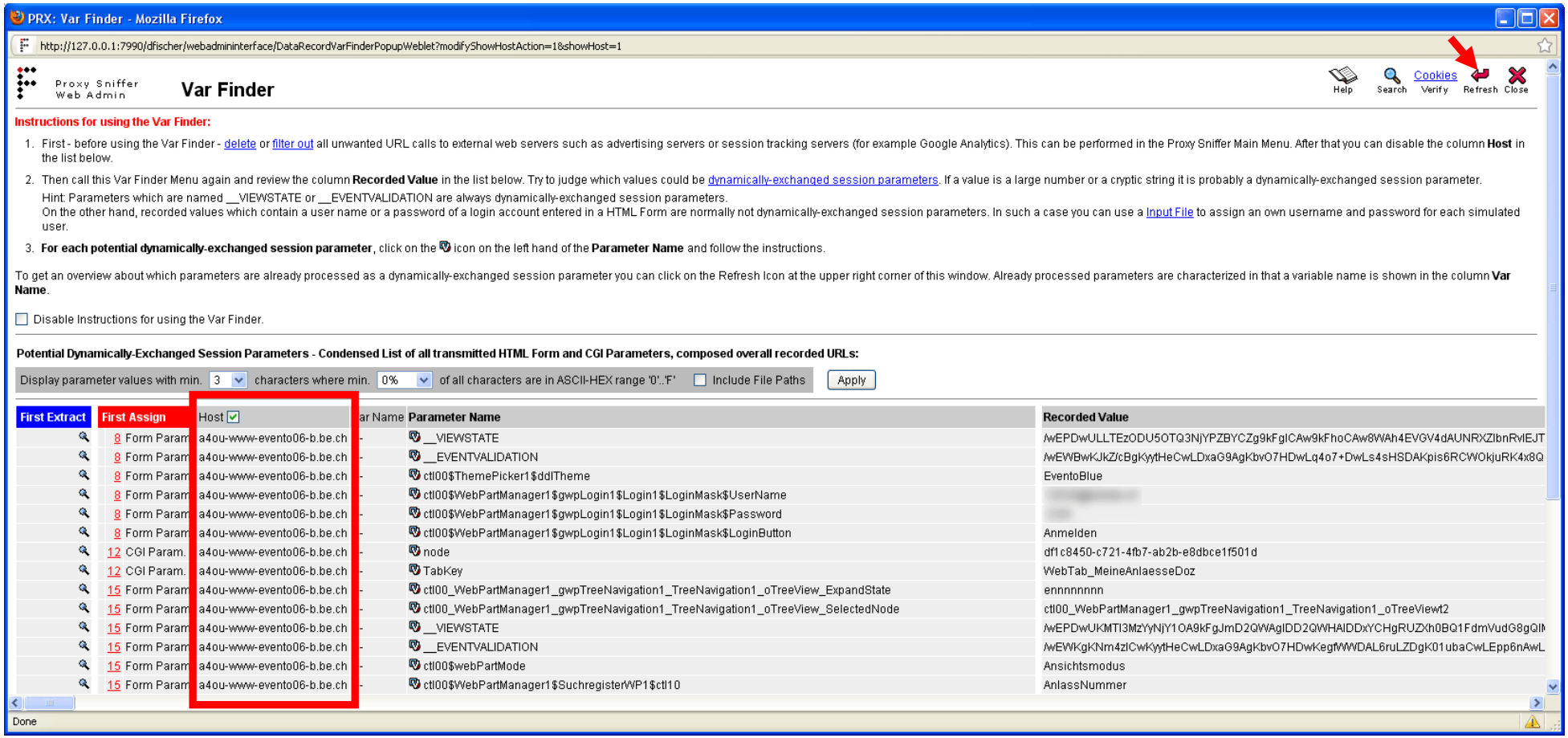


© 2011 - 2019 Apica All Rights Reserved

The **Var Finder** menu scans briefly the recorded web surfing session and as a result displays a list of all CGI parameters and all HTML form parameters that are sent via HTTP(S) requests to the web application. The list includes all the **parameter names** as well as their **parameter values** (column "Recorded Value").

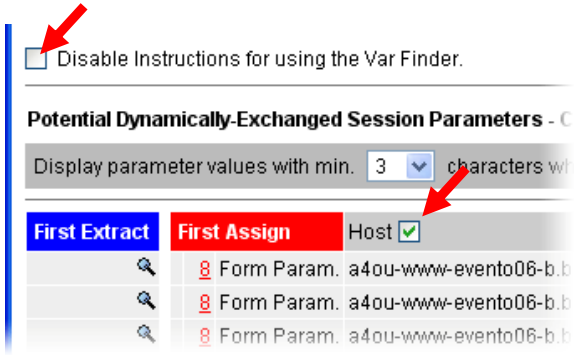
Note here that each **combination** of **parameter name AND parameter value** is shown **only once in the list**. The column "First Assign" shows in which URL this combination was sent for the first time to the web application.

Hint: review the column "**Host**" to verify that only the hostname of the server you want to load test is listed. If you find unwanted hostnames, return to the main menu and delete or filter-out the unwanted URL calls. Then click on the **Refresh Icon** in the Var Finder menu (top right corner).



© 2011 - 2019 Apica All Rights Reserved

In the Var Finder menu you can also **turn off** the option to the **displays the help text** and **hide the column "host**".



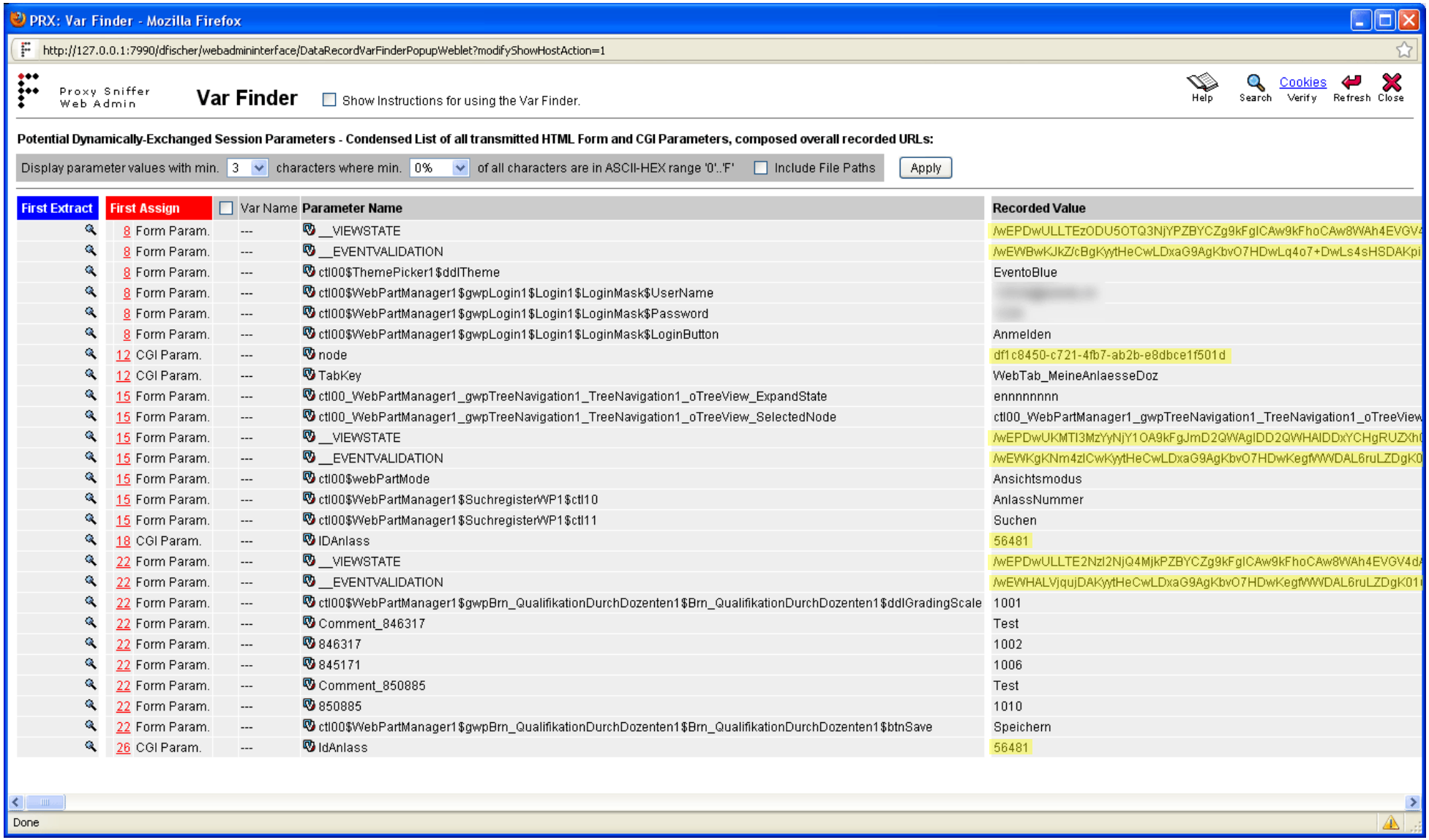
The next step is to identify which of the shown parameters are actually dynamically-exchanged. Usually they can be identified quite easily:

- If you see in the “Recorded Value” column a **parameter value** that you have **entered yourself in an HTML form** while recording the web surfing session, this is not a "dynamically-exchanged " parameter. For example, the data entered for a login account (username and password) are not "dynamically-exchanged session parameters".
- If the **parameter value** is a **cryptic string** or a **cryptic number**, this is almost always a "dynamically-exchanged session parameter". For example a value like "wEWBwKJkZ/cBgKyytHeCwLDxaG9AgKbvO7HDwLq4o7".
- If the **parameter value** is a **long number** and you have not entered this number into a HTML form, it is often a "dynamically-exchanged session parameter" (for example the number "56481"). Here you can also try to guess the meaning of the parameter name. For example, if the parameter has the name "EventID", this is probably a "dynamically-exchanged session parameter".
- If the **parameter value** is a **short number**, this is often not a "dynamically-exchanged session parameter" (for example the number "1001"). Note here that short simple numbers are often references (keys) of database records, and that their parameter names often resemble the name of the DB-Keys.

Hint: If you can contact a developer of the web application you can ask him or her to tell you if a parameter is "dynamically-exchanged". However, this is often not necessary because you can find out by yourself with a little bit of effort – as shown on the next pages in this document.

© 2011 - 2019 Apica All Rights Reserved

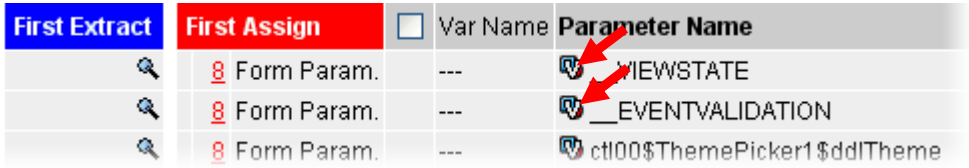
In this example, the values of a few candidate "dynamically-exchanged session parameters" are highlighted in yellow:



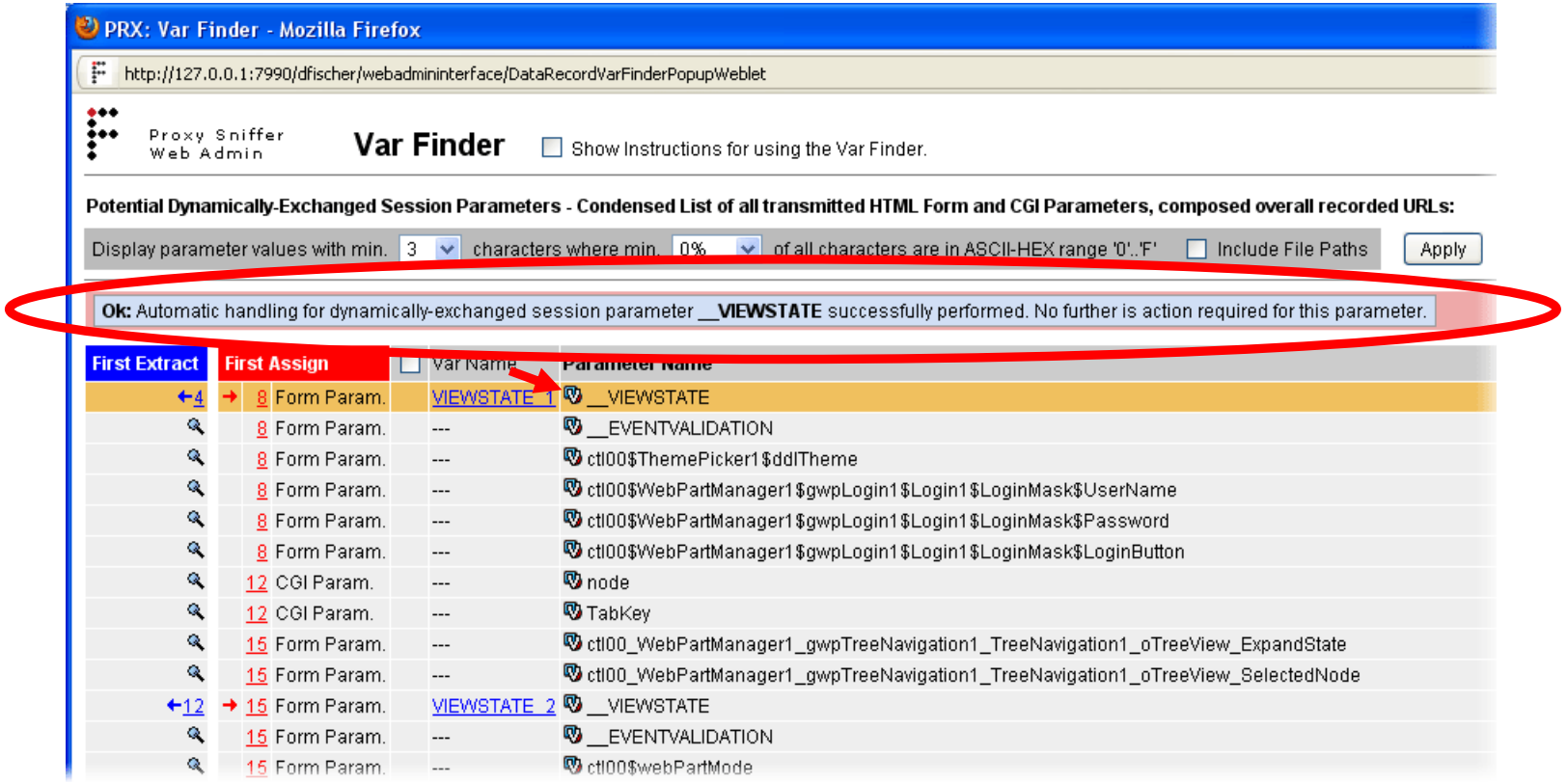
These are the parameters __VIEWSTATE, __EVENTVALIDATION, node, IDAnlass and IdAnlass.

© 2011 - 2019 Apica All Rights Reserved

In many cases, "dynamically-exchanged session parameters" can be **automatically handled**. Now go through the potential candidates you have identified in the Var Finder and click on the V-icon next to the parameter name:

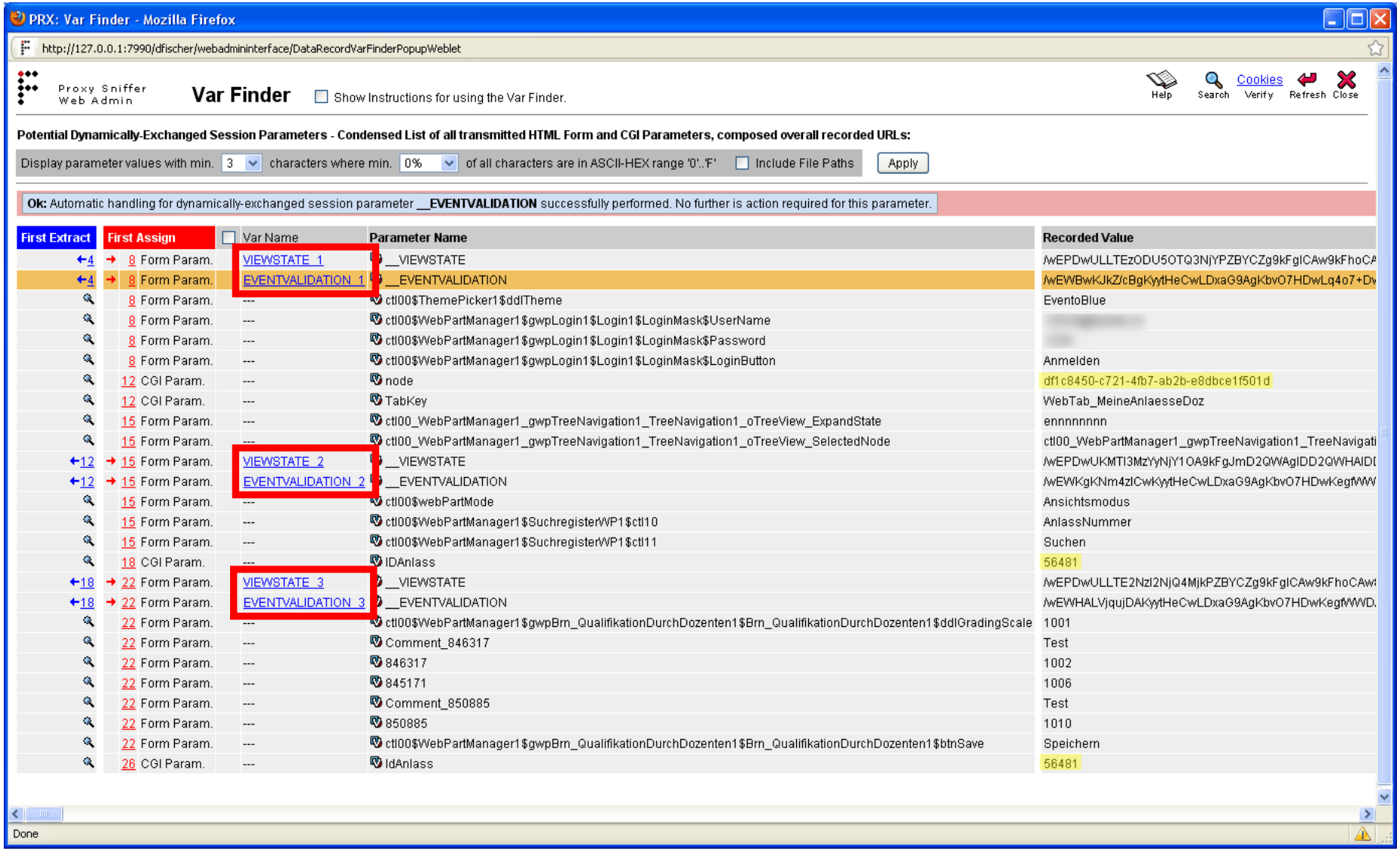


The parameters __VIEWSTATE and __EVENTVALIDATION are often used in Windows web server applications. After clicking on the V-icon you should see a success message indicating that ZebraTester was able to correctly handle the parameter. The highlighted success message is shown:



© 2011 - 2019 Apica All Rights Reserved

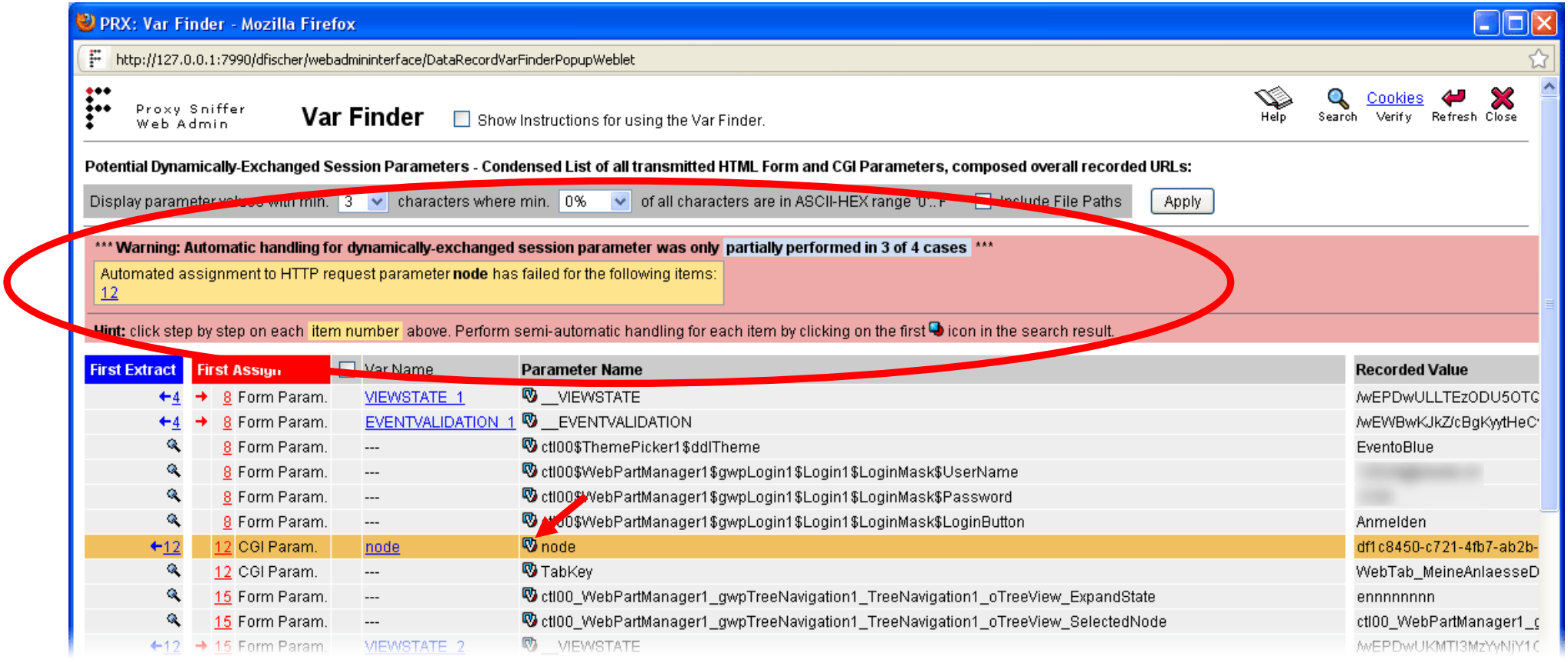
Once the __VIEWSTATE and __EVENTVALIDATION parameters have been configured for automatic handling the Var Finder menu will look as shown below. Please note that it only took you 2 mouse clicks in order to configure dynamic handling of your session parameters!



Left still remains the handling of the parameters node, IDAnlass and IdAnlass.

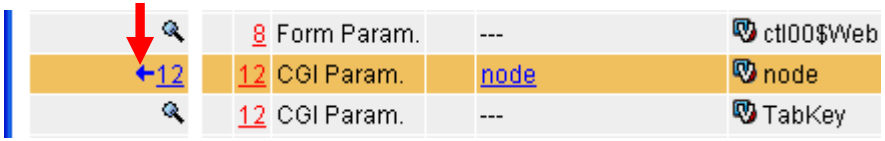
© 2011 - 2019 Apica All Rights Reserved

If the following warning message is displayed after clicking on the V-icon (as shown in the below example for the node parameter):



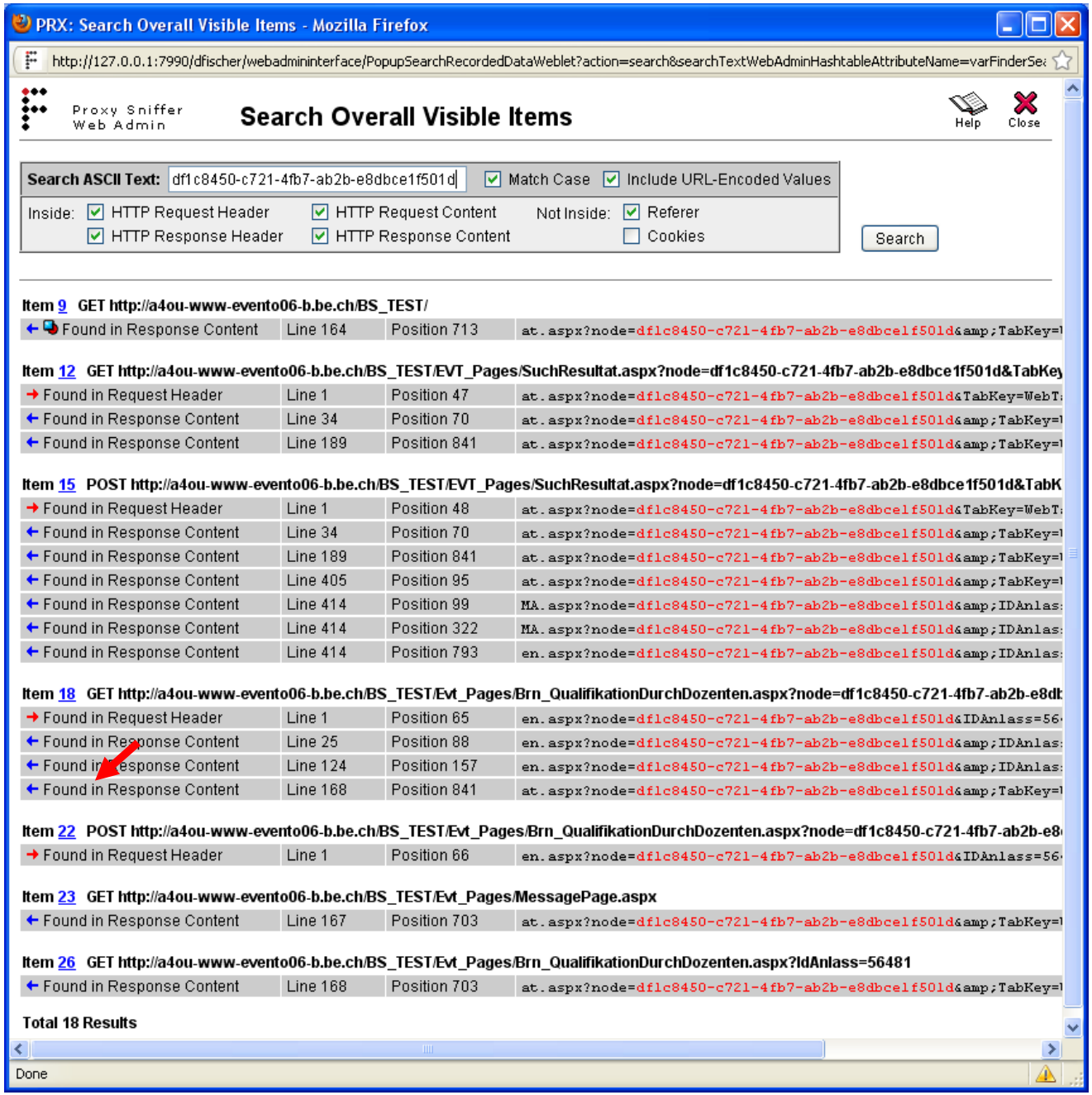
This indicates that the automatic handling of the parameter node partially failed. The text of the error message says that the value of the node parameter cannot be automatically assigned to the HTTP request of item number 12.

The best approach in such cases is to first get an **overview** of all URLs in where the **value** assigned to node occurs. For this purpose you can use the **Search Overall** menu which searches a text fragment over the entire recorded web surfing session.



With a mouse click on the blue arrow ← in the "First Extract" column, the **Search Overall** menu is directly invoked with the **value** of parameter node.

© 2011 - 2019 Apica All Rights Reserved



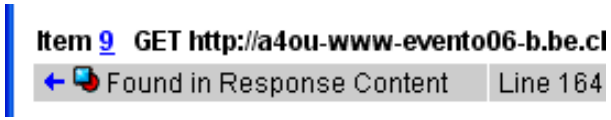
In the content of the **Search Overall** menu you can see that the value of the parameter node was **received** first from the web application in item number 9 (URL no. 9). This is represented by the blue arrow to the left ←.

We also see that the parameter value of node is **sent back** to the web application in the items 12, 15, 18, and 22. These transmissions are represented by the red arrow to the right →.

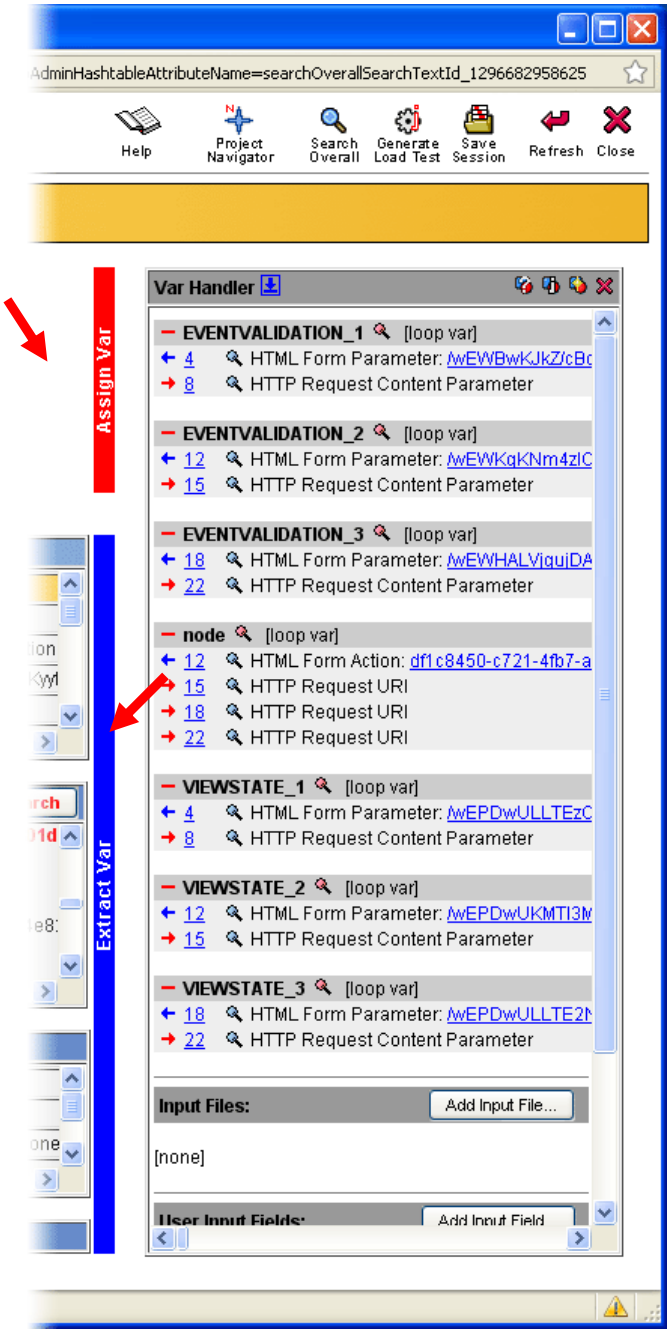
Because the error message in the Var Finder menu indicated that the automatic assignment did fail for item 12, the value of the parameter was probably extracted too late by the automatic algorithm (after, or in the response of item 12). But the value needs to be extracted in the response of item number 9.

By clicking on the **URL Details / Var Handler** link you can check in the "**Var Handler**" **area** which definitions have been automatically created:

© 2011 - 2019 Apica All Rights Reserved

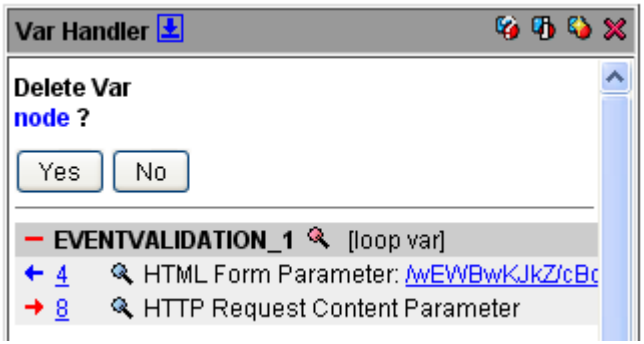


© 2011 - 2019 Apica All Rights Reserved



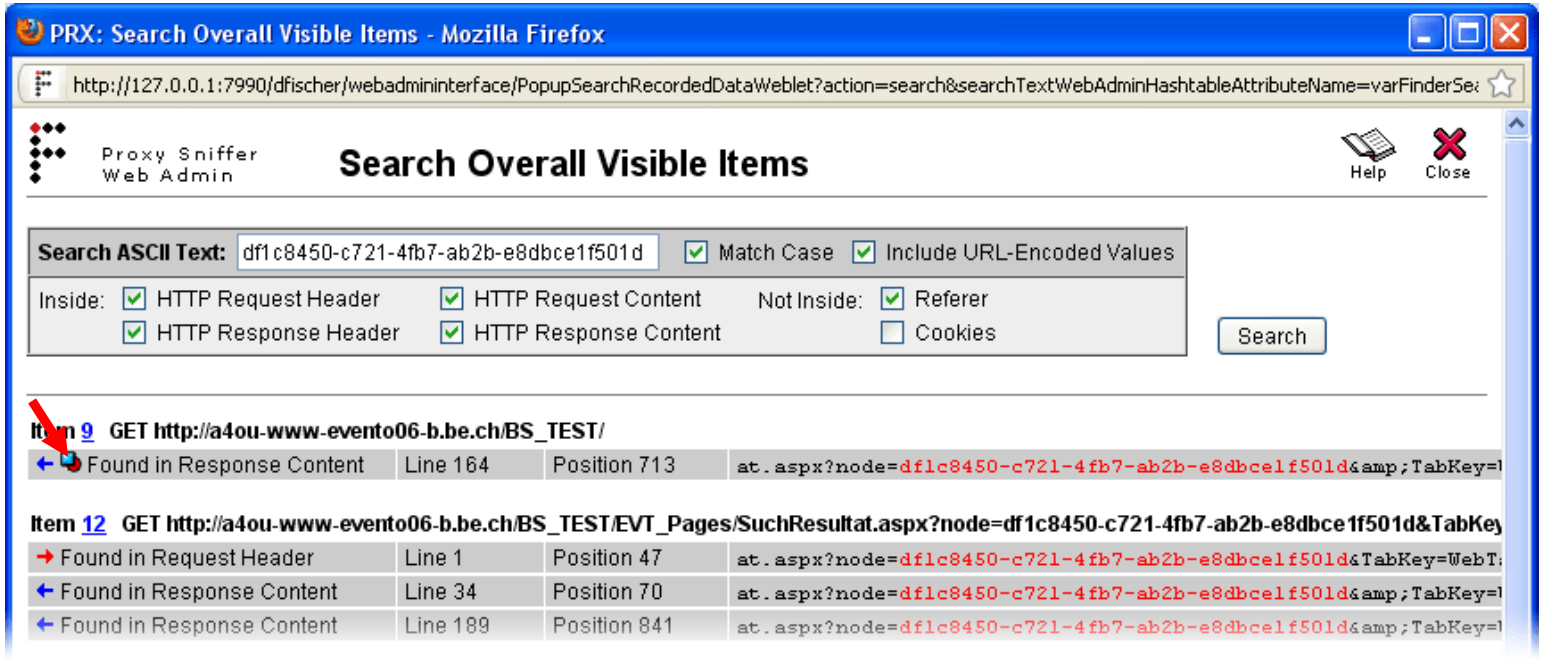
As the extraction of the value of the node parameter by the automatic algorithm occurred too late it has to be manually extracted in item 9. After that the value should also be manually assigned to the corresponding HTTP(S) request-parameter in item number 12.

As you can see from the screenshot below a number of steps that assign the node variable in various requests are already present. It is probably more efficient to first **delete** all of the automatically generated handlings for node in the **Var Handler** menu and then reconfigure everything:



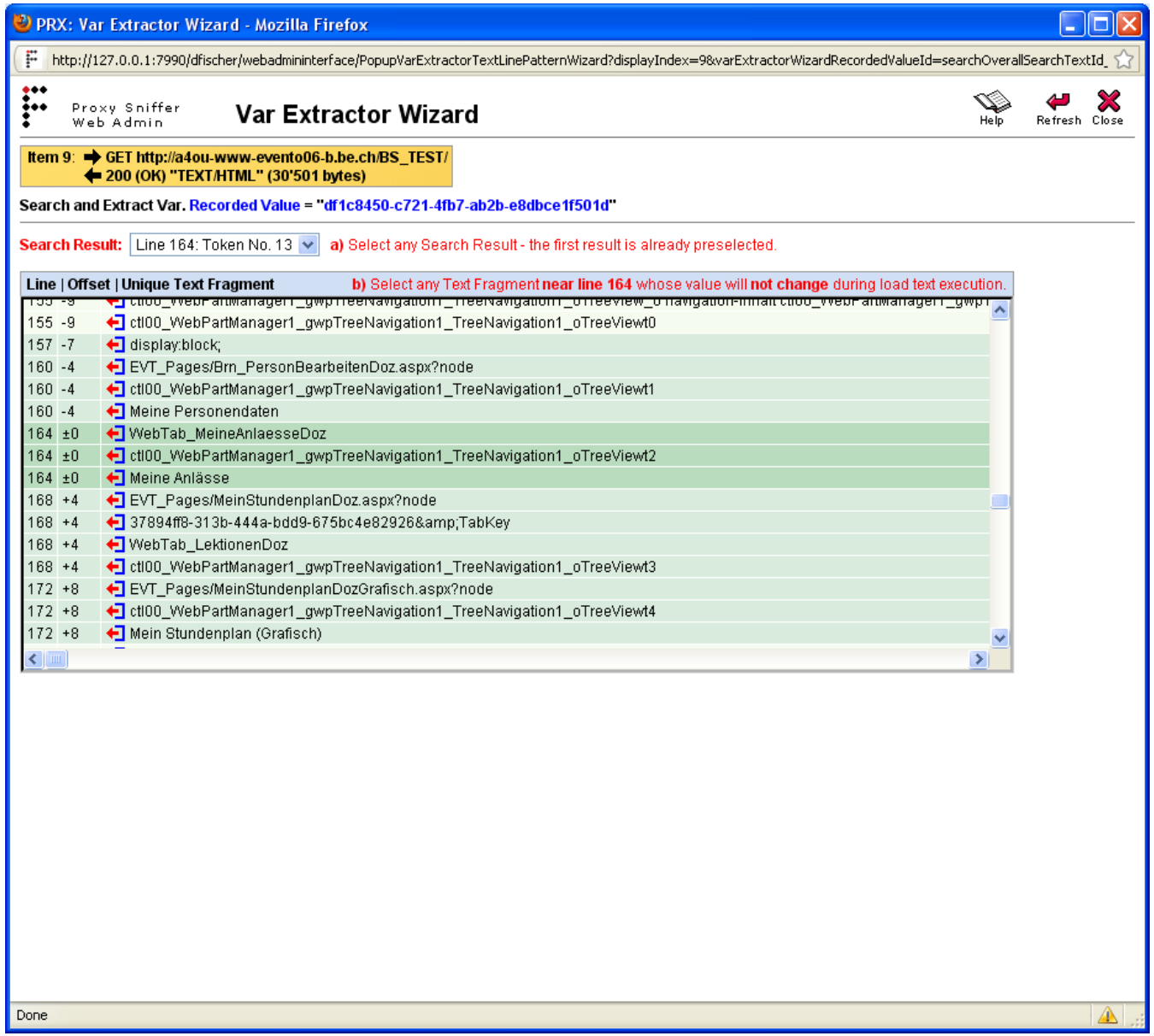
© 2011 - 2019 Apica All Rights Reserved

Then configure the extraction for the value of node from the **Search Overall** menu - with a click on the "Var Extract" icon (item 9):



By clicking on the "Var Extract" icon the "**Var Extractor Wizard**" is invoked:

© 2011 - 2019 Apica All Rights Reserved



The **Var Extractor Wizard** first searches the **value** of node in the response of item 9 (**Recorded Value**, displayed in **blue color**)

After the parameter value was found by the wizard the entire data content of the HTTP response of Item 9 is automatically divided into "unique text fragments", which are presented in a list.

The only thing you need to do is to **select one of the "unique text fragments"**. The selected text fragment will later be used as a "relative anchor" or as a "search pattern" to determine a line-offset to the parameter's value that needs to be extracted.

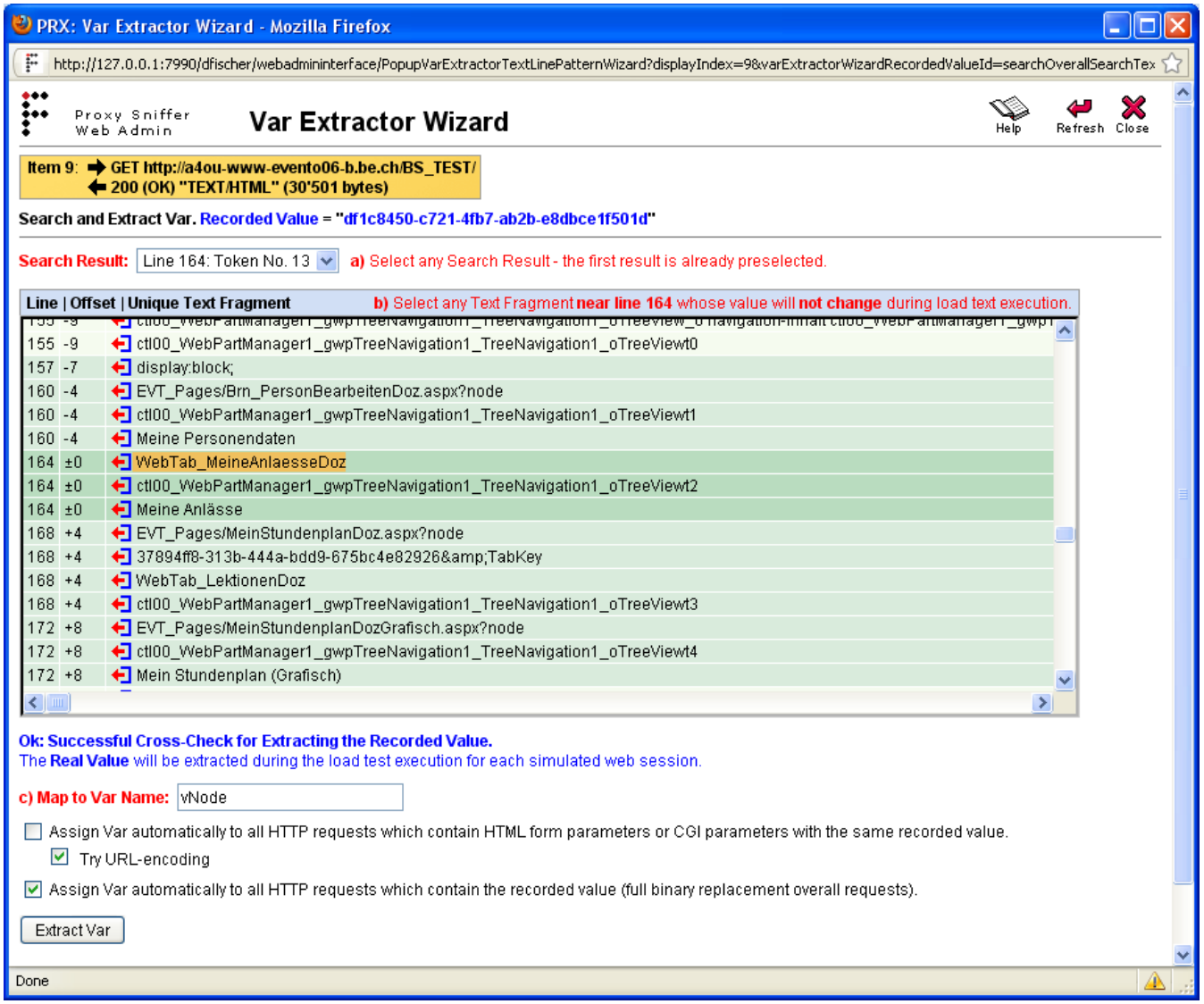
You can basically select any text fragment. However, you should consider the following two conditions:

1. **The selected text fragment should preferably be located close to the extracted value**. In the column "Offset" you see how close a text fragment is. You should select a text fragment which has only a small negative or positive "offset ".

2. **The selected text fragment should be "stable" during the load test execution (static text).** The text fragment itself should not represent a value of another "dynamically-exchanged session parameter".

For example, it would be **wrong** to select the text fragment "**37894ff8-313b-444a-bdd9-675bc4e82926&TabKey**". Instead it is **correct** to select the text fragment "**WebTab_MeineAnlaesseDoz**" (as shown in this example).

© 2011 - 2019 Apica All Rights Reserved



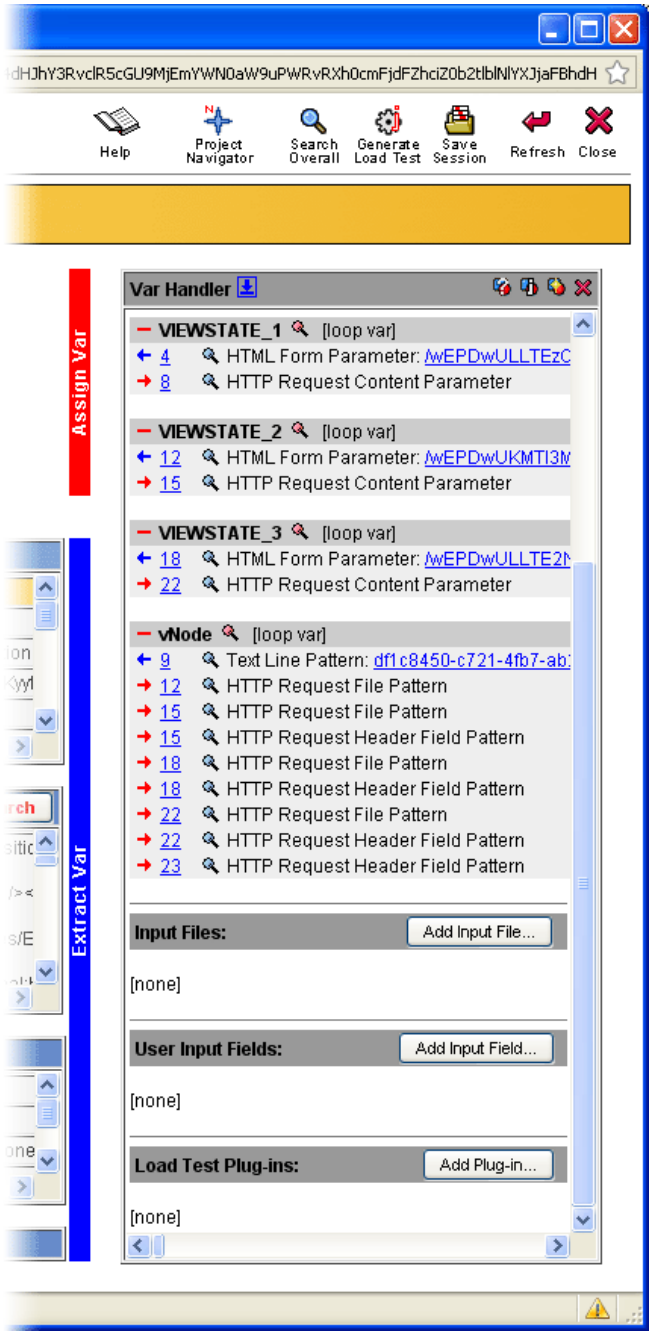
After you have selected the text fragment the **Var Extractor Wizard** displays a **success message**. You then have to **enter a name for the variable** in which the **value** of node will be extracted to.

You should use a unique and meaningful name for the variable. The definition of the new variable is then automatically created by the wizard.

At the bottom of the window you'll see **two checkboxes**. Their default values should be left unchanged as a rule. This means that the extracted value is automatically assigned to all relevant HTTP requests.

After clicking on the "**Extract Var**" button the corresponding variable definitions will be created by the wizard and shown in the "**URL Details / Var Handler**" menu.

© 2011 - 2019 Apica All Rights Reserved



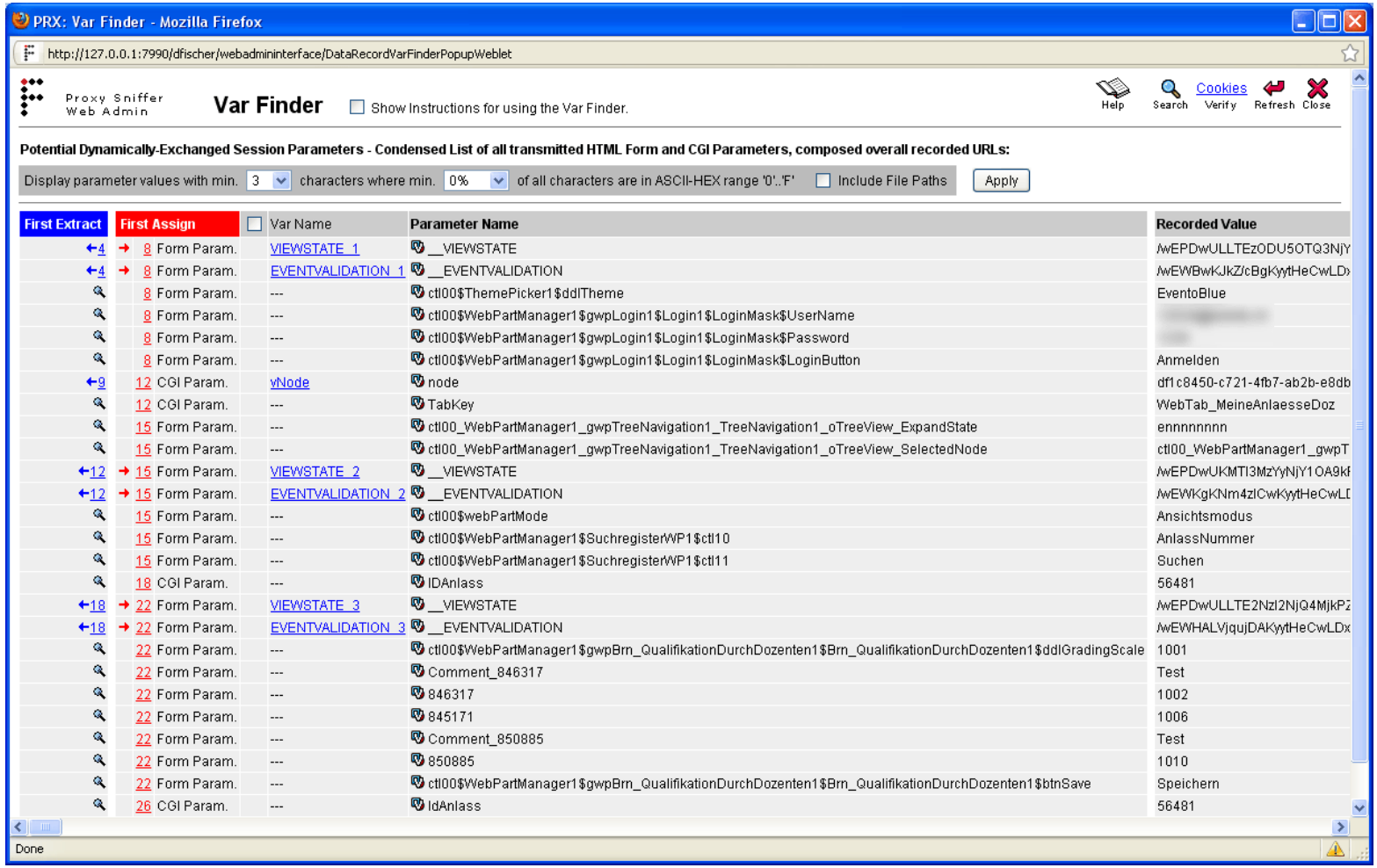
The manual configuration of the "dynamically-exchanged session parameter" node is now complete.

Only the handling of the parameters IDAnlass and IdAnlass needs to be configured.

Call the **Var Finder** menu again or click on the **Refresh Icon** at the top right corner of the **Var Finder** window.

© 2011 - 2019 Apica

All Rights Reserved



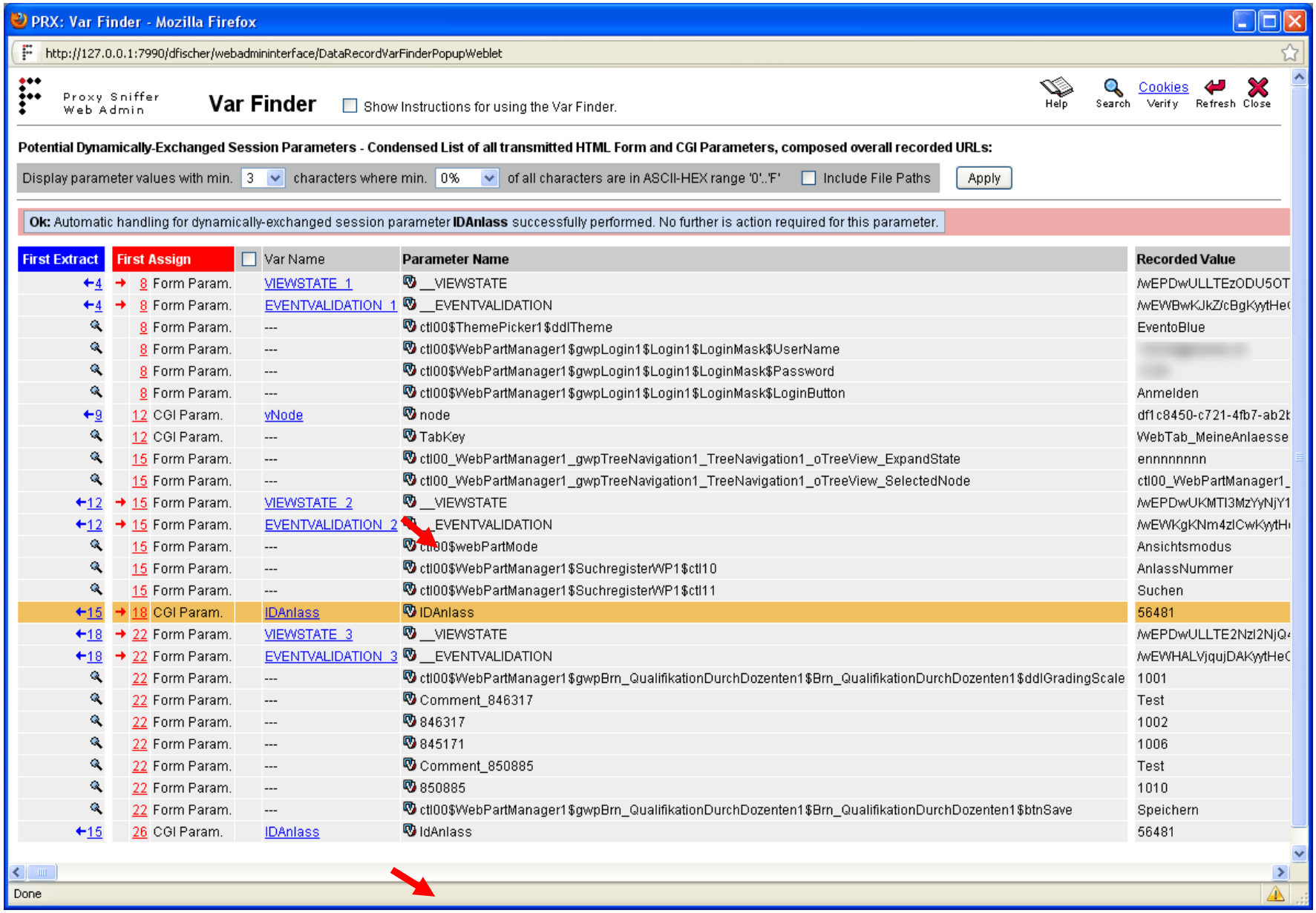
As you can see in this example, the parameter IDAnlass has the same value as IdAnlass (56481). The only reason why the parameter is shown twice is that the developers of the web application have used two different parameter names for the same thing.

One possibility would be to extract the value of this parameter by hand - as shown before – by using the Var Extractor Wizard. This would also assign the value of the parameter to all relevant URLs in one step.

To minimize effort you can first try automatic variable handling for both parameters. On success two new variables should be defined (IDAnlass and IdAnlass).

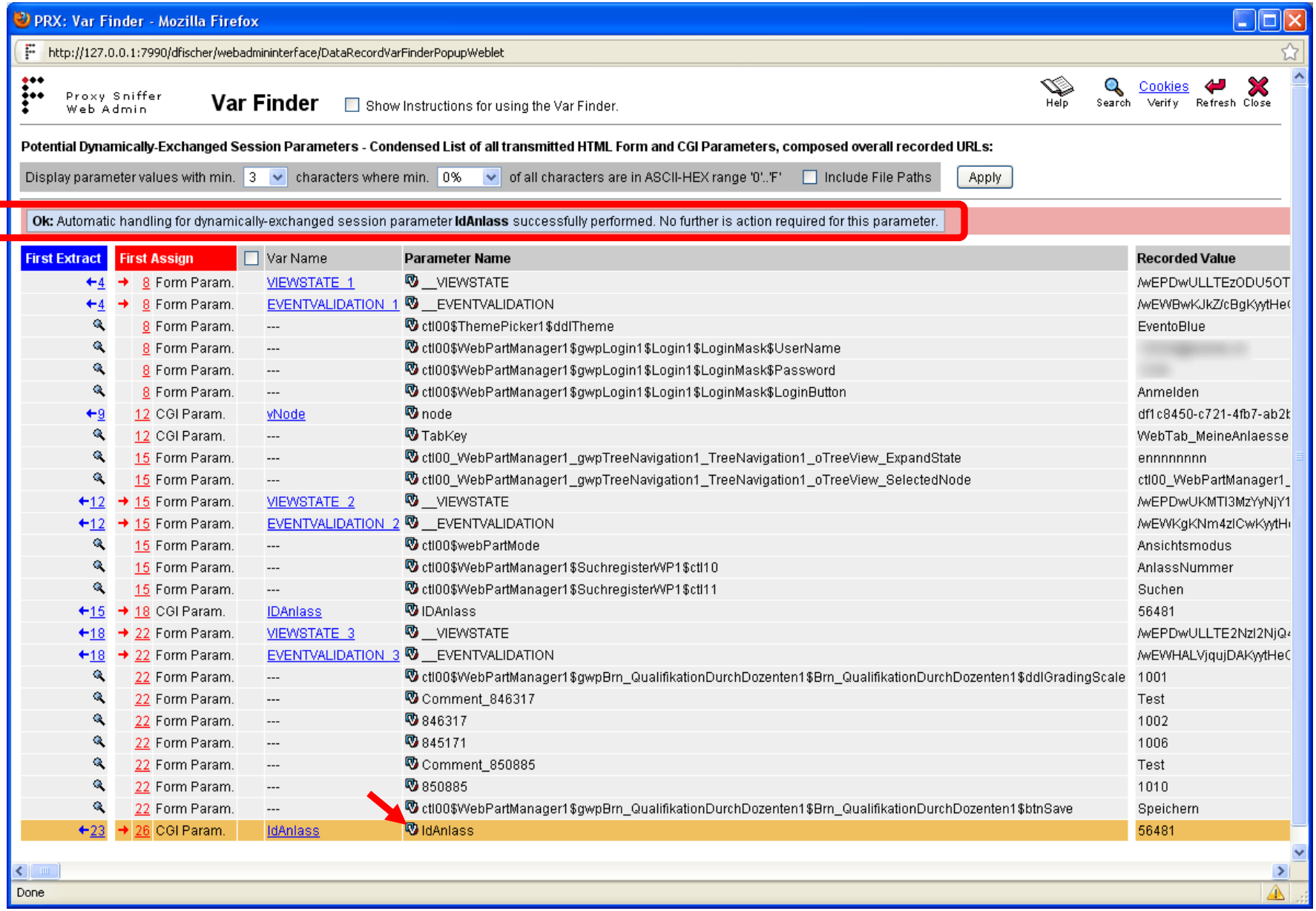
© 2011 - 2019 Apica All Rights Reserved

This screenshot shows that the automatic handling of IDAnlass was successful:

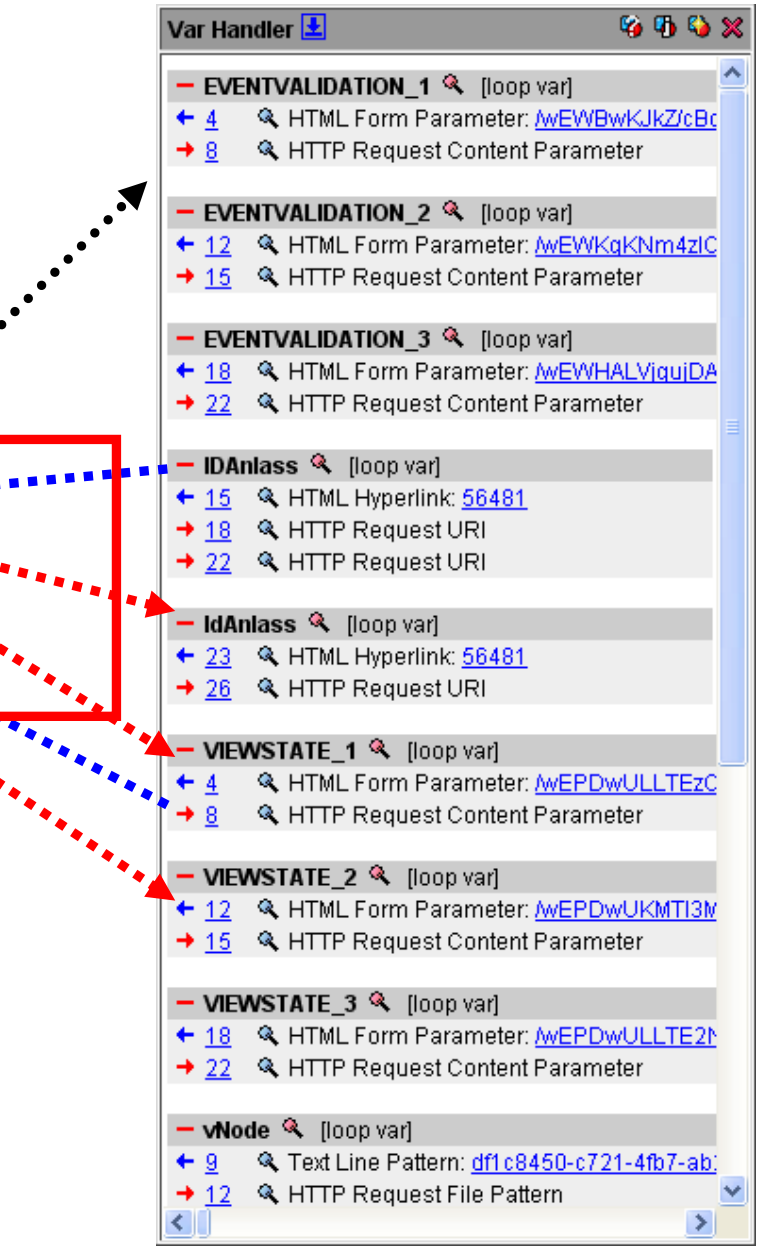


© 2011 - 2019 Apica All Rights Reserved

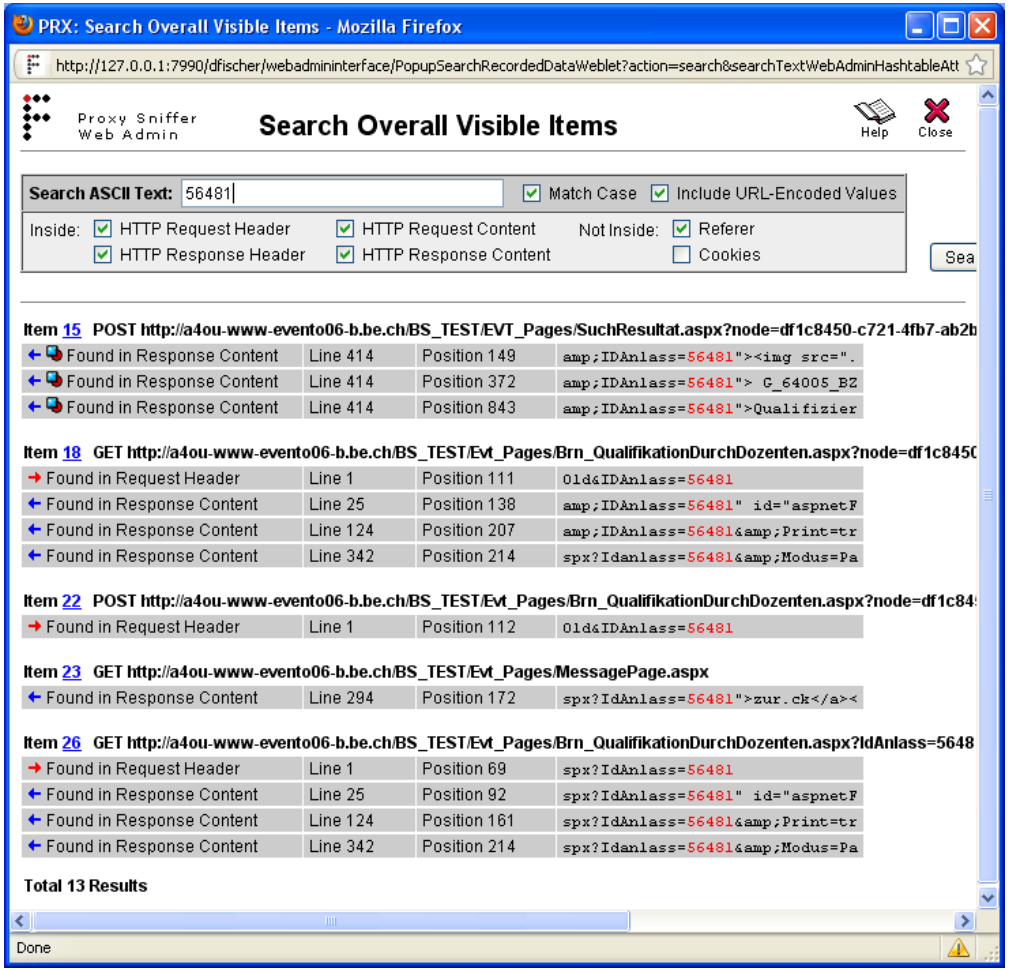
Similarly the automatic handling of IdAnlass succeeded:



The **Var Handler Definitions** look now like this:



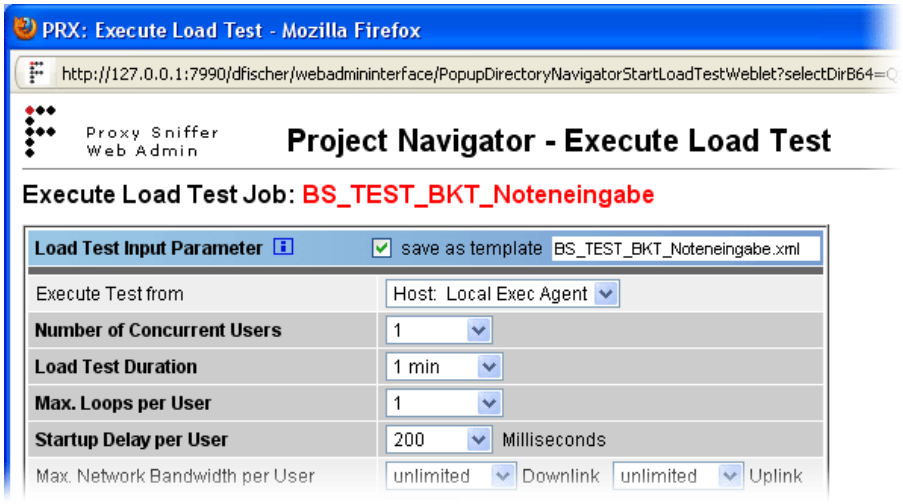
Review of the result: By clicking on the **parameter value** in **Var Handler** menu the **Search Overall** menu is opened. In there you can check that all the extractions/assignments for IDAnlass and IdAnlass are correct:



All looks good in this example; the definitions for the assignment of both variables have been created (item 18, 22 and 26). The **Search Overall** menu does not show any additional red arrows pointing right ➔, which are not covered by a corresponding definition in the Var Handler.

© 2011 - 2019 Apica All Rights Reserved

After this step the configuration of all "dynamically-exchanged session parameters" is complete. The parameters __VIEWSTATE, __EVENTVALIDATION, IDAnlass and IdAnlass were handled automatically, but the parameter node had to be manually configured.



We then strongly suggest to perform a functional test by performing a **trial run of the load test** with **only one simulated user** that executes **only one loop**.

The functional check should successfully complete, i.e. without any errors.

You are now ready to perform a load test simulating a larger number of users.

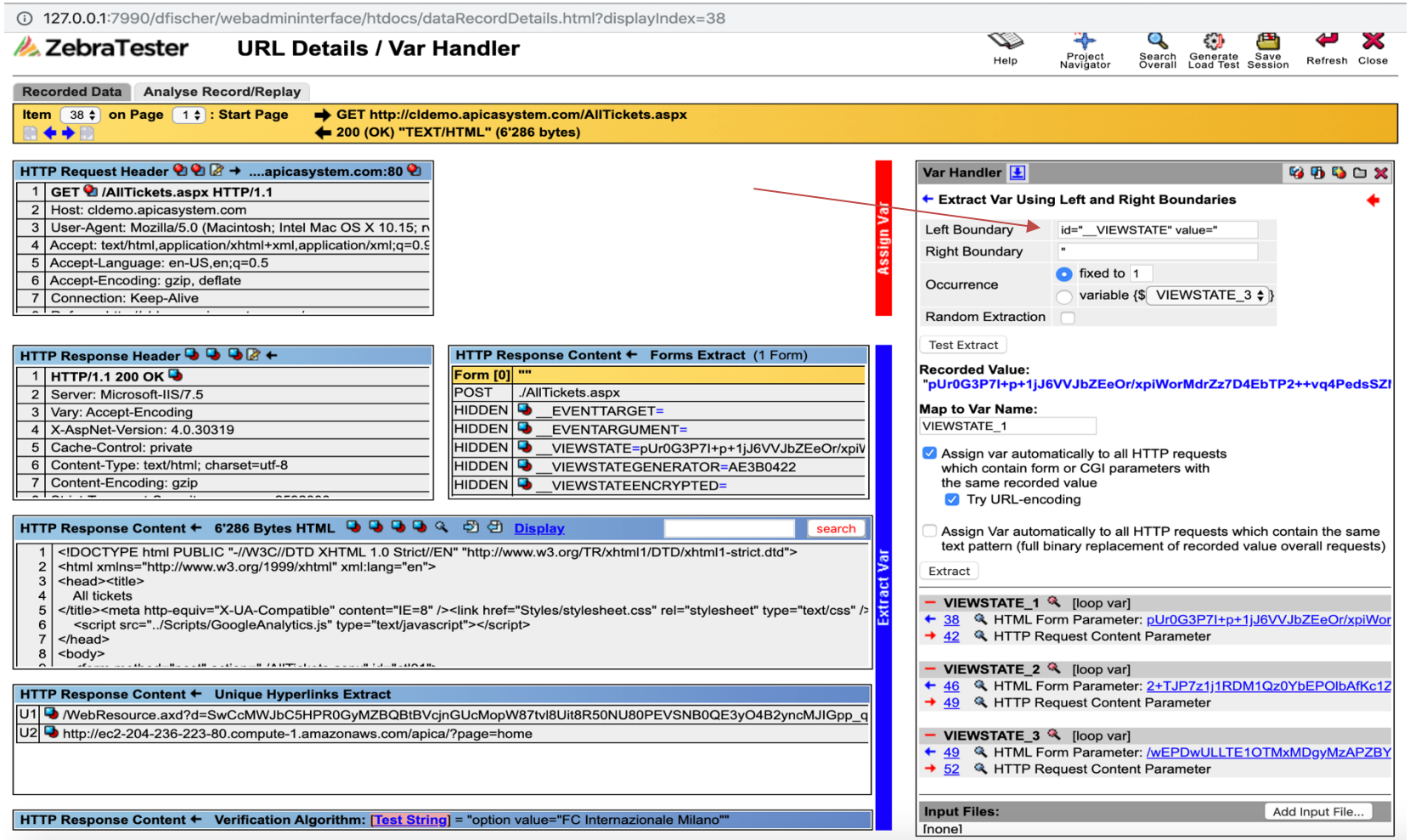
© 2011 - 2019 Apica All Rights Reserved

3 Other Ways to Extract and Assign Variables - Structured Data Access

The **Var Extractor Wizard** shown in the previous example extracts the variables using a **text-token-based algorithm**. This algorithm is technology-independent and works for all data that is NOT received in a raw binary format.

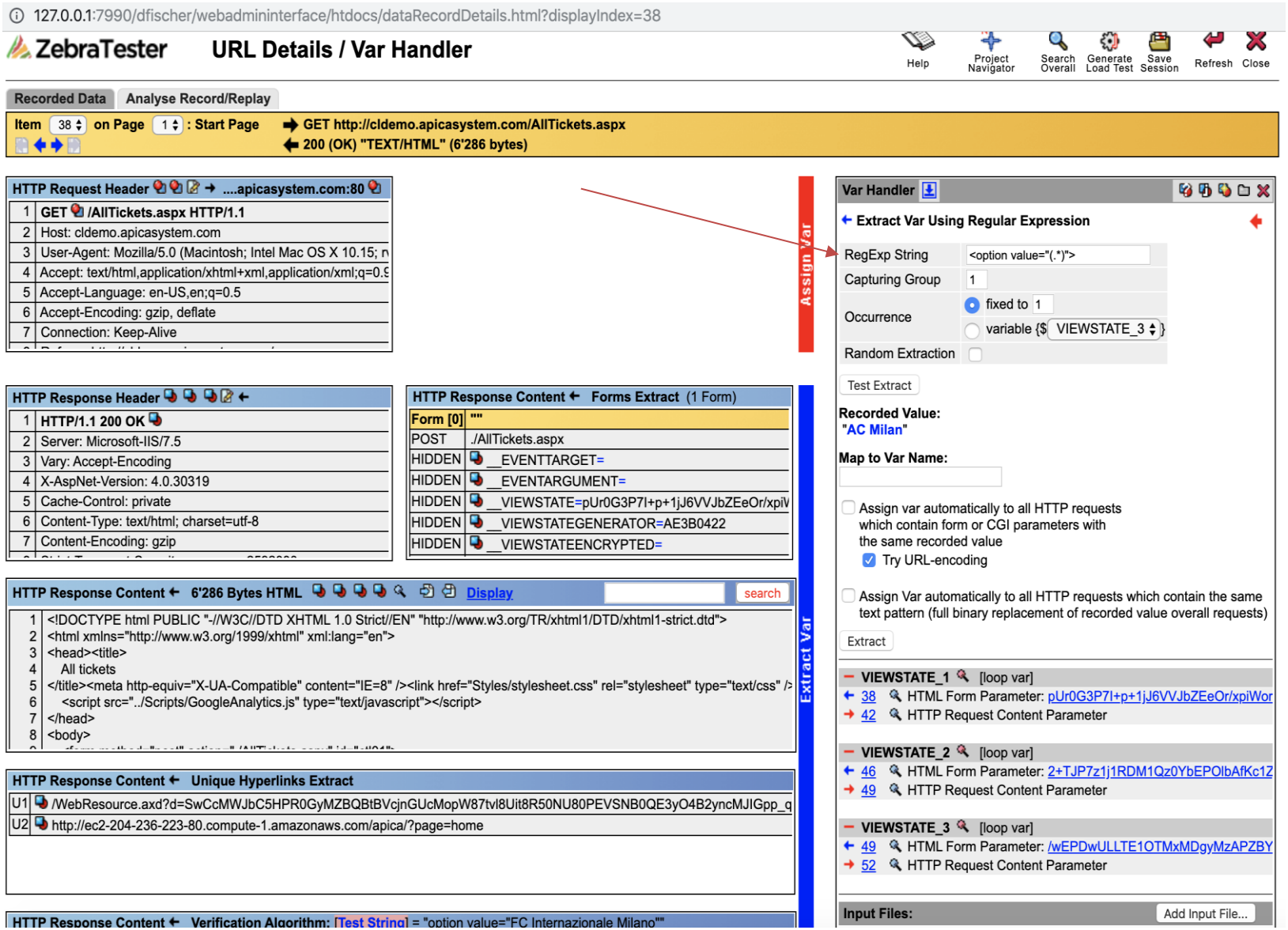
Apart from the text-token based extractor ZebraTester also supports the following extractor patterns

- Left and Right Boundary Based Variable Extractor



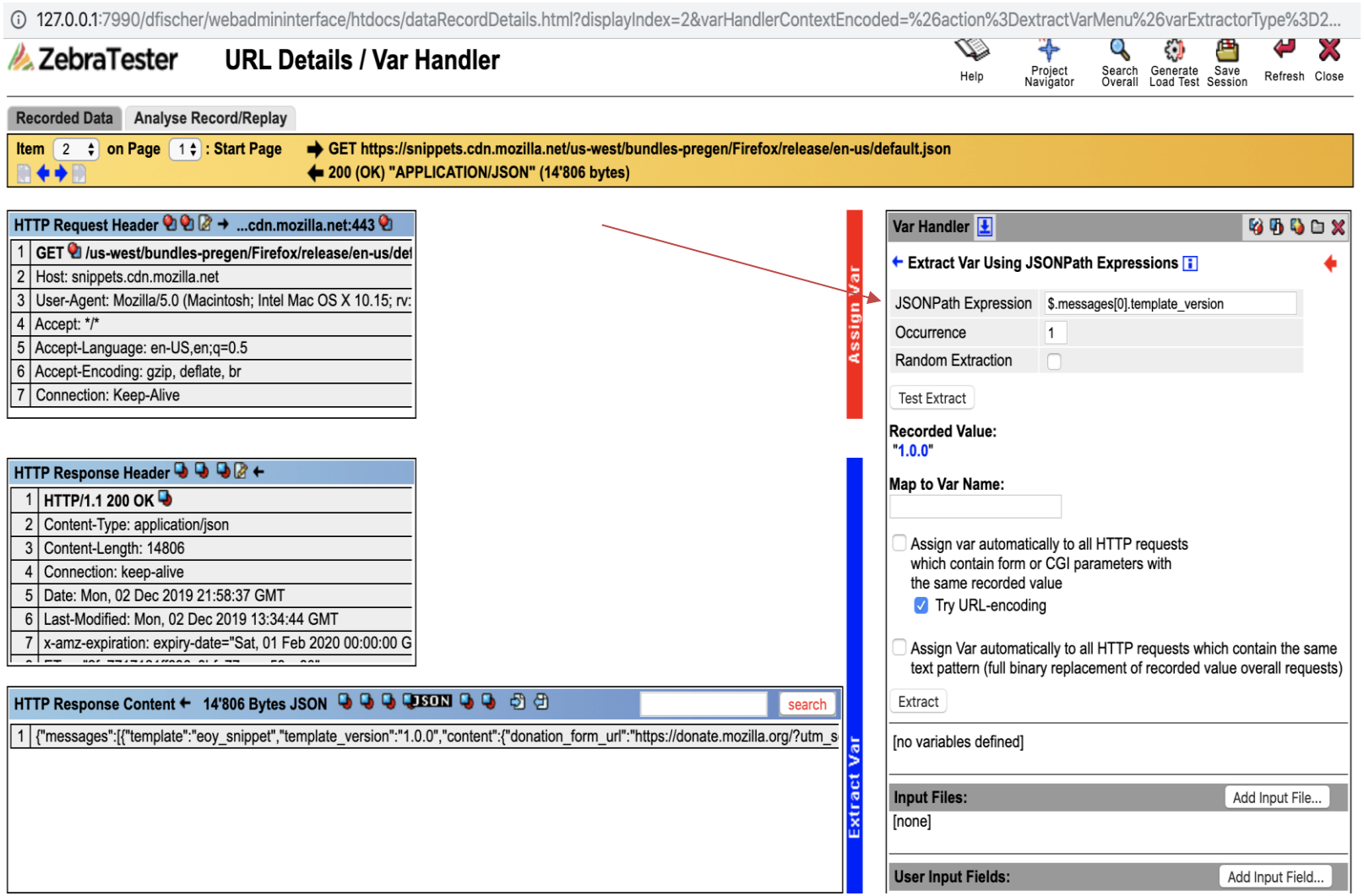
© 2011 - 2019 Apica All Rights Reserved

- RegExp Based Variable Extractor



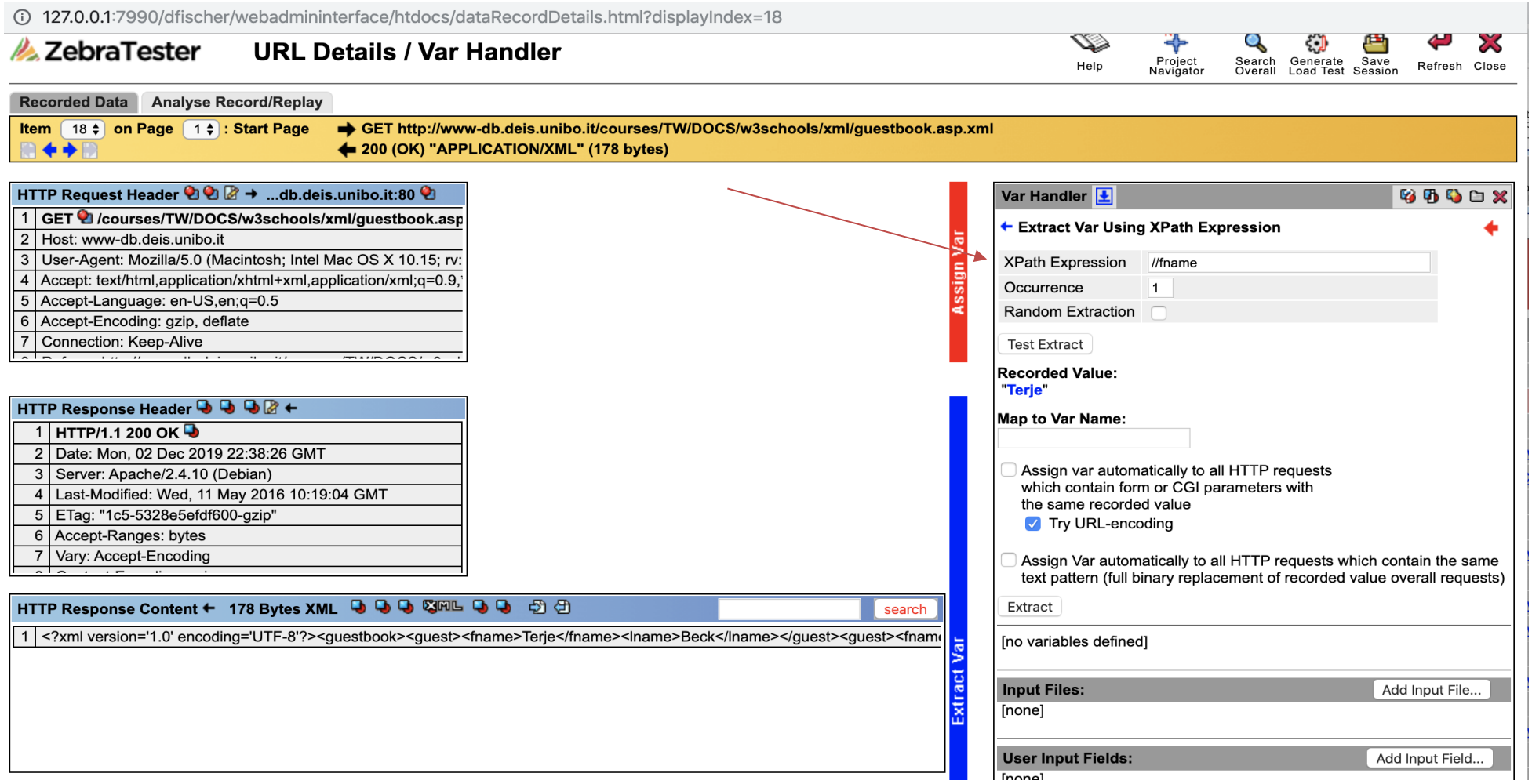
© 2011 - 2019 Apica All Rights Reserved

- JsonPath Based Variable Extractor



© 2011 - 2019 Apica All Rights Reserved

- XPath Based Variable Extractor



The Var Finder in ZT only displays the values of data contained in CGI parameters and in transmitted HTML forms.

You can also extract variables from other parts of the HTTP(S) responses by using the **structured data access** functions. Structured data access is supported for:

- HTTP host names, HTTP(S) TCP/IP ports, and transfer protocols ("HTTP" or "HTTPS")
- HTTP Request Paths
- HTTP CGI Parameters
- HTTP Redirects
- HTTP Header Fields
- HTML Form Data (inclusive "hidden" form fields)
- HTML Hyperlinks
- JSON Data
- XML and SOAP Data
- WebDAV Protocol Data
- Google Protobuf Data

The corresponding functions for extracting and assigning dynamically-exchanged values are accessible via the **URL Details / Var Handler** menu. Depending on the data format, the following icons are shown in **URL Details / Var Handler** menu:

for HTTP and HTML Data, and for the WebDAV Protocol Data

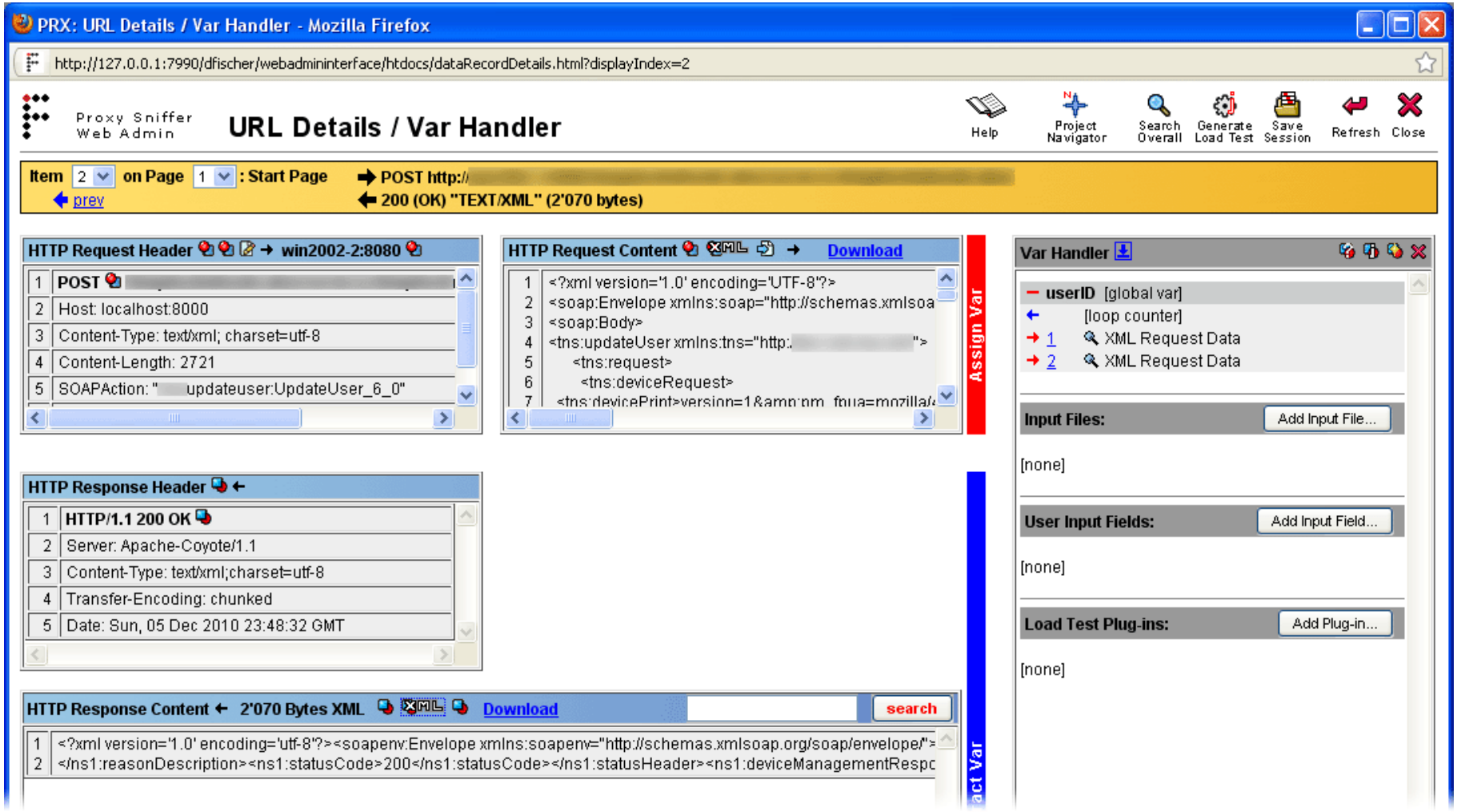
for XML and SOAP Data

for JSON Data

for Google Protobuf Data

© 2011 - 2019 Apica All Rights Reserved

Example: **structured data access** to **XML data**:



Even in this case the process of defining value extractions is straightforward. The extracted values are then assigned to all relevant HTTP/S requests of the web surfing session. **Further information about structured data access** is provided in the **ZebraTester User's Guide**.

© 2011 - 2019 Apica All Rights Reserved

4 Appendix A: Inner Working of the Text-Token-Based Algorithm (Var Extractor Wizard)

The **text-token-based algorithm** which is used in the **Var Extractor Wizard** works as follows:

1. First, the content of the URL response is automatically divided into lines (separated by <CR> and/or <LF> characters).

2. The value that has to be extracted will be searched automatically in the lines. The line numbers which contain the value to be extracted are memorized and used for further calculations.

3. All lines of the content are then automatically divided into text in fragments (called tokens). The separator characters, which stand between the tokens, are automatically determined (so-called token-delimiter characters).

4. All text fragments which occur only once in the URL response are displayed in the GUI. This is the list of the "unique text fragments" from which **a text fragment can be selected in the GUI by a mouse click**. Note that text fragments that contain the value to be extracted are not shown in the list. Furthermore, text fragments which are a subset of any other text fragment are also not shown in the list.

5. The selected text fragment is now used as an "anchor" to extract the value. A positive or negative line offset is calculated between the anchor and the value to be extracted.

6. Within the line which contains the value to be extracted, the token number of the value to be extracted is calculated.

7. All information which is required to extract the value during the load test execution is now available:
    a. The Text fragment of the anchor.
    b. The line offset between anchor and the value to be extracted (this can be a negative value, or zero, or a positive value)
    c. The token-delimiter characters which are applied in the line of the value to be extracted
    d. The token number of the value to be extracted

8. Finally, an automatic cross-check is made which simulates the runtime behavior to verify that the information used to extract the value is correct. **The result of the cross-check is displayed in the GUI**.

9. **The name of the variable in which the extracted value will be stored can now be entered in the GUI**. Additionally, you can also choose whether the extracted value (or rather its variable) should be automatically assigned to all relevant HTTP/S requests in the web-surfing session.

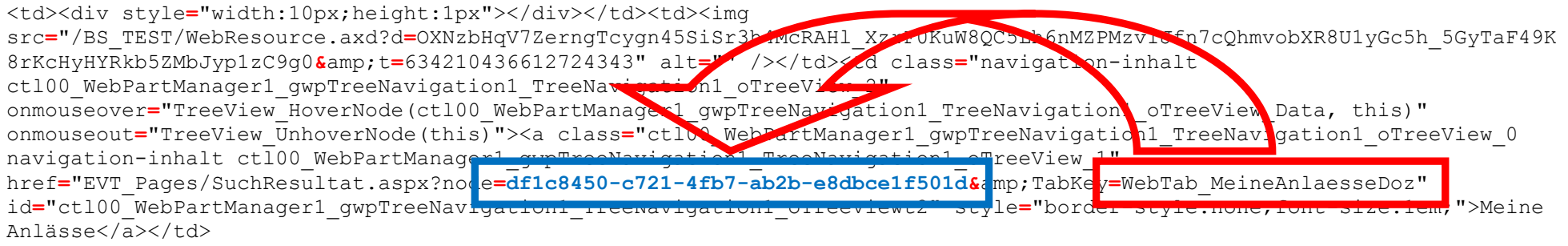
10. **With a simple mouse-click on the "Extract Var" button the corresponding definition is now generated**.

© 2011 - 2019 Apica All Rights Reserved

Note: in the **Var Handler** menu you can view the "internal configuration data" of the **variable extract definition** which was generated by the **Var Extractor Wizard.** This can be done by clicking at the corresponding magnifying glass icon 🔍:

© 2011 - 2019 Apica All Rights Reserved

Example: content of line 166 of a URL response:



```
<td><div style="width:10px;height:1px"></div></td><td><img
src="/BS_TEST/WebResource.axd?d=OXNzbHqV7ZerngTcygn45SiSr3b4McRAHl_XzrF0KuW8QC5Lh6nMZPMzvlJfn7cQhmvobXR8U1yGc5h_5GyTaF49K
8rKcHyHYRkb5ZMbJyp1zC9g0&amp;t=634210436612724343" alt="" /></td><td class="navigation-inhalt
ctl00_WebPartManager1_gwpTreeNavigation1_TreeNavigation1_oTreeView_2"
onmouseover="TreeView_HoverNode(ctl00_WebPartManager1_gwpTreeNavigation1_TreeNavigation1_oTreeView_Data, this)"
onmouseout="TreeView_UnhoverNode(this)"><a class="ctl00_WebPartManager1_gwpTreeNavigation1_TreeNavigation1_oTreeView_0
navigation-inhalt ctl00_WebPartManager1_gwpTreeNavigation1_TreeNavigation1_oTreeView_1"
href="EVT_Pages/SuchResultat.aspx?node=df1c8450-c721-4fb7-ab2b-e8dbce1f501d&amp;TabKey=WebTab_MeineAnlaesseDoz"
id="ctl00_WebPartManager1_gwpTreeNavigation1_TreeNavigation1_oTreeViewt2" style="border-style:none;font-size:1em;">Meine
Anlässe</a></td>
```

Note: During the execution of a load test ZebraTester does not search for line 166. Instead the line that contains the anchor is searched again at runtime. After that a relative line jump is made to the line which contains the value (line offset). This means that the algorithm works correctly even if the anchor is found at runtime in a different line than where it was when recording the web surfing session. However the extraction of the value will fail, if the line offset between the anchor and the value to be extracted is at runtime different than it was when recording the web surfing session, or if the token number of the value to be extracted is not the same as when recording the web surfing session. Therefore we recommend that you always use **structured data access** to extract values when possible. On the other hand, based on the experience of many successful performed load tests of our customers, we can conclude that the **text-token-based algorithm works surprisingly reliably** in almost all cases.

Additional note: When the **Var Finder** menu displays a success message for extracting and assigning a "dynamically-exchanged session parameter" this indicates that the parameter is handled by using **structured data access**.

© 2011 - 2019 Apica All Rights Reserved

5 Manufacturer

Apica

Manufacturer's Web Site: http://www.zebratester.com/
Apica Support: support@apicasystem.com
Apica Sales: sales@apicasystem.com

© 2011 - 2019 Apica All Rights Reserved