Next Page >> More Spider Script Examples

How To Write Spiders for Mine The Web

This document is an easy to follow tutorial that will teach you how to create spiders for Mine The Web. Spiders are what Mine The Web uses to retrieve content for you. Mine The Web has a very flexible spider scripting language that allows you to retrieve content from websites regardless of where the contents are placed within a webpage. By the end of this tutorial, you will realize that content you once thought was difficult or even impossible to structure and retrieve from a website can now be easily done using Mine The Web.

Your Very First Mine The Web Spider

The key to Mine The Web's flexibility and ease of use is its spiders, and the key to creating spiders for Mine The Web is in knowing the spider scripting language. You will see from this first example that creating a spider that does a whole lot is actually extremely easy and straightforward using the spider scripting language.

For our first spider, go ahead and point your browser to http://quote.yahoo.com. Take a look at the Market Summary column. Yup, you guessed it. In this example, we are going to retrieve the Dow, Nasdaq and S&P 500 quotes and store them in our own database. We are going to achieve this without using any XML at all, which is why Mine The Web is so powerful. It does not require data sources to be in XML format. Without further ado, here's the spider script that we'll need to retrieve those market quotes.

```
BEGIN HEADER
source:http://quote.yahoo.com/
CRto:3

BEGIN INFORMATION

BEGIN ACTION
startafter:
width=192 height=96></a></td></tr>|
endat:
<tr align=right><td align=left nowrap><a href="/q?d=t&s=^TNX">|
pattern:
<tr align=right><td align=left nowrap><a href="/q?d=t&s=^$1"><small><nobr>$2</nobr></sma
definition:
$1:SYMBOL:TEXT
$2:INDEX:TEXT
$3:CLOSING:FLOAT
$4:CHANGE:FLOAT:StripHTMLTags()
$5:CHANGEPCT:FLOAT:StripHTMLTags()

BEGIN DO
```

Looks too simple to be true? Indeed it is. But before we explain what's going on in the script, we'll show you how to input this spider into Mine The Web so that Mine The Web can run it. For this, go to our demo page if you aren't already there. Go to the Spiders page and click on Add Spider. Give your spider (actually, it's OUR spider since we came up with this example :) a name in the Spider Name field, and copy and paste the above script into the Spider Script field. For the Destination Table field, enter the name of the database table that you would like Mine The Web to store the retrieved data into. Make sure the table name is a valid database table name. Then click the Submit button. You will be returned to the main Spiders page.

Now click on the Execute link next to the spider you just created, and voila! You will see a page with the market quotes from Yahoo!. Want to store that data in a database table for future use? Just click on the "Store in table ..." button. Remember that you do not have to create a new spider every time you want to get market updates from Yahoo!. Just execute the same spider again tomorrow to get tomorrow's quotes. There's even a way to automate the process to perform the updates automatically! More on that later.

Holy Craps, That Was Easy!

That's the first time we heard that. Today! :) Now let's get to the down and dirty stuff, and the most exciting part of Mine The Web: the spider scripting language.

As you may have noticed from the script example above, a spider is made up of 4 sections: BEGIN HEADER, BEGIN INFORMATION, BEGIN ACTION, and BEGIN DO. Let's start with the first section, BEGIN HEADER.

All you really need to tell Mine The Web in the BEGIN HEADER section is the URL of the page or site that you want to retrieve content/data from. In the above example, we told Mine The Web that we want to retrieve content/data from http://quote.yahoo.com, which is Yahoo's Market Quotes page.

We'll leave the BEGIN INFORMATION section empty for the moment.

The BEGIN ACTION section is where all of the fun stuff happens. To understand this section, we'll need you to view the HTML source of the http://quote.yahoo.com page. You usually do this by clicking on View->Source from your browser. Go ahead and do it. The BEGIN ACTION section consists of several instructions, such as startafter:, endat:, pattern:, and definition:. The startafter: and endat: instructions basically tell Mine The Web where the data you are interested in is in the HTML code of the source page (the source page is the Yahoo Quotes page). For example, if your source page looks like this:

```
<HTML>
<BODY>
Data line 1
Data line 2
Data line 3
</BODY>
</HTML>
```

and you want to retrieve the data lines between the opening and closing tags, then your startafter: and endat: instructions should look like this:

```
startafter:
<BODY>|
endat:
</BODY>|
```

The '|' character is used to terminate an instruction.

Now let's go back to the Yahoo! example. Look at the source of the http://quote.yahoo.com/ page. With a little bit of sleuthing, you'll discover that the data we are interested in starts right after

```
width=192 height=96></a></td></tr>
```

and ends right before

```
<tr align=right><td align=left nowrap><a href="/q?d=t&s=^TNX">
```

So to tell Mine The Web to look only at the HTML contents between those 2 strings and ignore everything else, we say

```
startafter:
width=192 height=96></a></td></tr>|
endat:
<tr align=right><td align=left nowrap><a href="/q?d=t&s=^TNX">|
```

Make sense? No rocket science involved here. Mine The Web now knows that it should only be concerned with the contents between those 2 strings, which is

```
<tr  align=right><td  align=left  nowrap><a  href="/q?d=t&s=^DJI"><small><nobr>Dow</nobr></sm
```

Notice that each <tr> holds one market quote. We want to extract the useful information within each <tr>...</tr> and discard the contents that we don't want such as HTML tags. The scripting language has a very flexible way of telling Mine The Web what it is that you want to keep and what you don't want to keep.

```
pattern:
<tr  align=right><td  align=left  nowrap><a  href="/q?d=t&s=^$1"><small><nobr>$2</nobr></smal
```

This tells Mine The Web to store the contents between the ^ and ", and between the <nobr> and <small> tags into the variables $1, $2, $3, $4 and $5 respectively. The definition: instruction is used to associate the variables $1 to $5 with useful names that will be used as field names when the data is stored in the database.

```
definition:
$1:SYMBOL:TEXT
$2:INDEX:TEXT
$3:CLOSING:FLOAT
$4:CHANGE:FLOAT:StripHTMLTags()
$5:CHANGEPCT:FLOAT:StripHTMLTags()
```

The definition instruction above is telling Mine The Web to store variable $1 in a table field named INDEX as a string data type, variable $2 in INDEX, variable $3 in CLOSING as a fractional number, and so on. The variable names themselves must be one character long, so $7 and $a for example are valid variable names, but $FOO is not.

StripHTMLTags() is a modifier function that is executed for the CHANGE and CHANGEPCT fields to remove any HTML tags that may be in those variables before they are stored in the database. More on that later.

You can store data as text or fractional numbers as shown above using the TEXT and FLOAT keywords, or as round numbers using the keyword INT. Blobs are also supported (using the keyword BLOB). TEXT fields are limited to 255 characters. If you want to store anything more than 255 characters, blobs can come in handy. Finally, the BEGIN DO line in the example simply tells Mine The Web to execute the script.

That's it. You've just created your very first spider script to retrieve specific content from within a chaos of HTML code. Play around with this example for a while and we'll move on to another example.

Next Page >>

[<< Previous Page](#) [Next Page >>](#) [More Spider Script Examples](#)

Cinema Listings, Anyone?

Here's our next example. Cable Car Cinema and Cafe is an artsy fartsy cinema. Not the sort of place you'd expect to find an XML data source. Which is exactly where Mine The Web makes life easier. In this example we are going to introduce the BEGIN INFORMATION section and the @n variables. Go grab a cup of coffee.

Cable Car Cinema and Cafe

Visit the Cable Car Cinema and Cafe website and view the HTML source. We are going to retrieve the list of shows playing at the cinema. To do this, we have to find out where in the HTML source the movie listings appear. This is almost always a recurring "pattern" of HTML code. As we saw in the previous Yahoo! Quotes example, the market quotes were encapsulated within a repeating pattern of <tr>'s. See if you can identify the pattern in this example. The pattern is:

```
<FONT SIZE="+2"> <b>
Movie Name

</b></FONT>
<blockquote>
Movie Image<P>
Movie Description

<p>
Showing Times
Running Time: Movie Length
</blockquote>
```

We used black to distinguish values that are different for each occurence of the pattern. The first pattern occurs right after

```
</FONT><p>
```

and the last pattern is immediately followed by

```
</tr></table>
```

So far, our spider script looks like this:

```
BEGIN HEADER
source:http://www.cablecarcinema.com/index.html
CRto:3

BEGIN INFORMATION

BEGIN ACTION
startafter:
</FONT><p>

|
endat:
</tr></table>
</pre></blockquote></font>
<br>|

BEGIN DO
```

Adding in the pattern: and definition: fields will result in:

```
BEGIN HEADER
source:http://www.cablecarcinema.com/index.html
CRto:3

BEGIN INFORMATION

BEGIN ACTION
startafter:
</FONT><p>

|
endat:
</tr></table>
</pre></blockquote></font>
<br>|
pattern:
$1<FONT SIZE="+2"> <b>
$2</b></FONT>
<blockquote>
$3
<p>
$4
Running Time: $5
</$6>
|
definition:
$1:
$2:TITLE:TEXT
$3:DESCRIPTION:TEXT:StripHTMLTags():StripTrailingWhitespace()
$4:TIMES:TEXT:StripHTMLTags():StripTrailingWhitespace()
$5:RUNNINGTIME:TEXT
$6:

BEGIN DO
```

A little bit of explanation is necessary in the above listing. Why do the $1 and $6 lines not have field names? Well, if you specify a variable without a field name, Mine The Web will not store it in the database. This is useful when there is a piece of information in a pattern that isn't always the same but you're not interested in keeping. For example, the title of the movie isn't the same each time the pattern repeats, but you want to store that information in the database. Therefore, you give the $2 variable a field name (TITLE). If however you decided that you did not want the movie names to be stored in the database, then you would not give the $2 variable any name.

Look at each pattern in the Cable Car HTML source, and you may notice that there isn't always the same number of empty lines between each pattern. Putting an unnamed variable $1 at the start of each pattern will "eat up" those empty lines. Otherwise, Mine The Web won't know where to start each pattern.

Variables $3 and $4 have got additional things to talk about. After the name and data type field, you can optionally include up to 2 modifier functions to process the contents of that variable. What are modifier functions? Modifier functions can be used to process a variable's data before storing it in the database. Again looking at the Cable Car HTML source, you'll see that the description of each movie contains plenty of
 tags. We don't want these tags to be stored in our database, we just want the raw description text, so we use the StripHTMLTags() modifier function to strip off all HTML tags before storing it in the database. The StripTrailingWhitespace() tag strips off any trailing whitespace. Also notice that if the movie listing has an image, the variable $3 will contain an <IMG> and <P> tag which we don't want since $3 is supposed to be just the movie description field. The StripHTMLTags() modifier function takes care of getting rid of those tags.

Here's a list of all the modifier functions that are available in Mine The Web's spider scripting language:

- ConvertBRToNL() : Converts HTML
 tags into newline characters
- PrependWith(arg1) : Adds the string arg1 to the beginning of the variable. Example: $3:LINK:TEXT:PrependWith (http://www.site.com) will prepend LINK with http://www.site.com . This is useful if links in a HTML file are relative links and you want them to show up in your database as absolute links
- PrependWithIfNotEmpty(arg1) : Same as PrependWith but only performs the operation if the variable is not empty/blank
- StripHTMLTags() : Removes all HTML tags
- StripNbspAndWhitespace() : Removes all leading and trailing whitespace and all 's
- StripLeadingWhitespace() : Removes leading whitespace
- StripTrailingWhitespace() : Removes trailing whitespace
- StripWhitespace() : Removes leading and trailing whitespace
- SubStr(arg1,arg2) : Returns a portion of the variable value, starting from character position arg1, with length arg2. For example, if the data retrieved is "abcdef", SubStr(1,3) will convert it to "bcd". SubStr(0,8) will convert it to "abcdef". If arg1 is a negative value, the extraction will start at the arg1'th character from the end of the variable's value. That is, SubStr(-3,1) will convert "abcdef" to "d". Lastly, if arg2 is negative, then that many characters will be omitted from the end of the variable's value. For example, performing SubStr(0,-1) on "abcdef" converts it to "abcde"
- Replace(arg1,arg2) : Replaces all occurences of arg1 with arg2. For example, applying Replace("-",":") on a variable that contains "1022-1203-123" will convert it to "1022:1203:123". If either arg1 or arg2 themselves need to contain the "," character, escape it with a "\". For example, to remove all "," characters from your retrieved data, use Replace (\,)

BEGIN INFORMATION

What if for every database record that the spider creates, we also want to store the name of the cinema and its location? This information is not present within the pattern. No problem, we can use the BEGIN INFORMATION section to achieve this. Let's extend our spider script to include some spider script in the BEGIN INFORMATION section:

```
BEGIN HEADER
source:http://www.cablecarcinema.com/index.html
CRto:3

BEGIN INFORMATION
CINEMA:TEXT:Cable Car Cinema and Cafe
LOCATION:TEXT:Providence, RI

BEGIN ACTION
startafter:
</FONT><p>

|
endat:
</tr></table>
</pre></blockquote></font>
<br>|
pattern:
$1<FONT SIZE="+2"> <b>
$2</b></FONT>
<blockquote>
$3
<p>
$4
Running Time: $5
</$6>
|
definition:
$1:
$2:TITLE:TEXT
$3:DESCRIPTION:TEXT:StripHTMLTags():StripTrailingWhitespace()
$4:TIMES:TEXT:StripHTMLTags():StripTrailingWhitespace()
$5:RUNNINGTIME:TEXT
```

```
$6:

BEGIN DO
```

This tells Mine The Web to create 2 additional fields for every record stored, called CINEMA and LOCATION, which will both be stored as text. For each occurence of the pattern defined in the BEGIN ACTION section, Mine The Web also stores the values 'Cable Car Cinema and Cafe' and 'Providence, RI' in the fields CINEMA and LOCATION.

Now, what if we want to store the duration the movies are playing in each record? This information is shown at the top of the Cable Car main page next to the logo. This piece of information is the same for each repetition of the pattern, just like the BEGIN INFORMATION example above. However, the difference is that we do not know the value for this piece of information in advance. It can change from week to week. In the previous example, we knew in advance what the name of the cinema is and where it is located (there's only one Cable Car Cinema in the world). We don't have the same luxury in this case. So what do we do? Simple, use defpatterns.

Think of defpatterns as miniature BEGIN ACTION sections, or sub BEGIN ACTION sections. defpatterns are defined within the @ and ^@ characters. Take a look at this script:

```
BEGIN HEADER
source:http://www.cablecarcinema.com/index.html
CRto:3

BEGIN INFORMATION
CINEMA:TEXT:Cable Car Cinema and Cafe
LOCATION:TEXT:Providence, RI
SHOWINGUNTIL:TEXT:@1

BEGIN ACTION
startafter:
</FONT><p>

|
endat:
</tr></table>
</pre></blockquote></font>
<br>|
pattern:
$1<FONT SIZE="+2"> <b>
$2</b></FONT>
<blockquote>
$3
<p>
$4
Running Time: $5
</$6>
|
definition:
$1:
$2:TITLE
$3:DESCRIPTION:TEXT:StripHTMLTags():StripTrailingWhitespace()
$4:TIMES:TEXT:StripHTMLTags():StripTrailingWhitespace()
$5:RUNNINGTIME
$6:
@1:
defstartafter:
Showing through |
defendat:
</FONT><p>|
^@
```

```
BEGIN DO
```

The SHOWINGUNTIL line in the BEGIN INFORMATION section tells Mine The Web that the value for that field is determined by the @1 defpattern. The @1 defpattern itself is defined in the BEGIN ACTION section between the characters @1 and ^@1.

```
defstartafter:
Showing through |
defendat:
</FONT><p>|
```

Can you figure out what's going on there? Very simple. All the above 4 lines say is that the value for the SHOWINGUNTIL (@1) field is between the "Showing through " and "</FONT></p>" text in the HTML source, which is the showing duration.

That's all there is to it. You now have a very good understanding of Mine The Web's spider scripting language. The next page of this tutorial will cover the rest of Mine The Web's spider scripting language.

[<< Previous Page](#) [Next Page >>](#)

[<< Previous Page](#) [Next Page >>](#) [More Spider Script Examples](#)

Special characters

Certain characters have special meanings in the spider scripting language. We will describe these characters here.

The # character is used to include comments in a script. Example:

```
BEGIN HEADER
source:http://www.cablecarcinema.com/index.html # The data source

BEGIN INFORMATION

BEGIN ACTION
startafter: # startafter instruction
</FONT><p>

|
endat: # endat instruction
</tr></table>
</pre></blockquote></font>
<br>|

BEGIN DO
```

Everything that appears after the # character on the same line is ignored by Mine The Web. If the HTML code of the data source that you are retrieving contains the # character, you will have to escape it by prefixing it with a \. For example, the # character is used in HTML to denote a color. You'll have to escape the # character with a \, otherwise Mine The Web will ignore everything occuring after that character.

```
startafter:
</FONT color="\#00FF00">
```

Another special character that you already know about is the $ character. It is used to define variables. Again, if this character is present in the original HTML code of the data source, you'll have to escape it with a \ .

Other special characters that have to be escaped if they are present in the HTML code are: @, |, ^, and \ . As you already know, The @ character is the defpattern character. The | character is the terminator character. The \ character is the escape character itself, so if a \ appears in the HTML code, you'll have to escape the \ with a \. For example, if your source HTML looks something like this:

```
<p>My favourite ASCII characters are #, $, &, @, |, ^, and \</p>
```

then your script should escape the special characters:

```
<p>My favourite ASCII characters are \#, \$, &, \@, \|, \^, and \\</p>
```

What is the CRto: instruction anyway?

You may have noticed that in most of the previous examples, we used a CRto: instruction in the BEGIN HEADER section. What is CRto:? CRto: tells Mine The Web to convert carriage returns in a spider script to other characters. This is useful when each line in the HTML code that a web server outputs does not end with a carriage return character. Some web servers terminate lines with newlines, or carriage-return followed by a newline. In these cases, use the CRto: instruction to make Mine The Web match the web server's output format. The syntax for using CRto: is

```
CRto:n
```

where n is either 1, 2 or 3. A value of 1 tells Mine The Web to convert the carriage returns in your spider script into carriage-return followed by a newline. The value 3 tells Mine The Web not to perform any conversion. Finally, the value 2 tells Mine The Web to convert the carriage-returns to newlines. So for example, if the web server you are retrieving data from ends each line in the HTML source with a newline, you would use:

```
CRto:2
```

There is a useful utility included with Mine The Web that will determine what the appropriate CRto: value of a HTML source should be. This utility is appropriately called the Smart CRto Utility and is accessible from the Add Spider page. The utility will ask you for the URL of a HTML source and tell you what the CRto: value of that HTML source should be.

Did you know?

Did you know that you can use more than one set of BEGIN HEADER, INFORMATION, ACTION and DO's in a single script?

```
# Retrieve and store data from a webpage
BEGIN HEADER
source:http://www.test.com/example-page-1.php
BEGIN INFORMATION

BEGIN ACTION
...
BEGIN DO

# Retrieve and store data from another webpage
BEGIN HEADER
source:http://www.test.com/example-page-2.php
BEGIN INFORMATION

BEGIN ACTION
...
BEGIN DO
```

The script is executed sequentially, meaning that Mine The Web will retrieve data from example-page-1.php first, followed by example-page-2.php.

<< Previous Page Next Page >>

[<< Previous Page](#) [More Spider Script Examples](#)

Following links

Mine The Web has the powerful ability to follow links. For example, say if the fictional site http://www.mynewssource.com/index.html contains a list of news headlines, but the full story can only be accessed by clicking on each headline to follow the link. With Mine The Web, you can achieve link following using the <FOLLOWLINK MERGE> directive in the definition: section of BEGIN ACTION. Here's an example:

```
...

BEGIN ACTION

...

pattern:
<a href=$1>$2</a><br>|
definition:
$1:<FOLLOWLINK MERGE>LINK:262_TEXT
$2:HEADLINE:TEXT

...
```

In the $1 variable line, the field name LINK is prepended with <FOLLOWLINK MERGE>. This tells Mine The Web that apart from storing the LINK data in the database, you also want the spider to follow the link. After loading the page that the link points to, Mine The Web, will then execute spider 262. Each spider has a unique spider ID, you can find out the ID of a spider by hovering your mouse over the Execute link in the main spider listing page, and copying down the value of 'filter_id=' in the status bar of your browser.

In the above example, there would need to be a separate spider with ID 262 to parse the data when Mine The Web fetches the page pointed to by the link. Here's an example of that spider:

```
BEGIN HEADER
source:
CRto:2

BEGIN INFORMATION

BEGIN ACTION
startafter:
<table>|
endat:
</table>|
pattern:
<tr>
<td>$1</td>
</tr>
|
definition:
$1:STORY:TEXT

BEGIN DO
```

The spider looks just like a regular spider, except that the source: line is empty. This makes sense as the spider already knows where to retrieve data, from the link provided by the previous spider.

Sending a form POST

You can simulate a form POST by including a POSTdata: line in the BEGIN HEADER section. The POSTdata line should come

before the source: line. For example:

```
BEGIN HEADER
POSTdata:username=myname&password=password
source:http://www.mysite.com/login.asp
CRto:2

BEGIN INFORMATION
...
```

This will simulate sending the POST variables username and password to http://www.mysite.com/login.asp

Retrieving content from a series of URLs

Sometimes it can be handy to be able to retrieve content from a series of URLs without having to write a series of spider scripts. For example, if you would like to retrieve a complete list of products from a website, one way to do it is to retrieve each product detail page:

```
http://www.timetraveldevices.com/products/detail.php?product_id=1
http://www.timetraveldevices.com/products/detail.php?product_id=2
http://www.timetraveldevices.com/products/detail.php?product_id=3
...
http://www.timetraveldevices.com/products/detail.php?product_id=10000
```

Obviously, the downside of this is that you'd have to define a separate spider for each of those URLS, resulting in 10,000 spiders! If the website had a product index page that listed and linked to each and every one of their products, you could use the link following technique described earlier. But if not, all hope is not lost. Mine The Web gives you the ability to grab content from all those 10,000 pages using only one spider! Here's what you'd need to use in your spider's source field in order to achieve this:

```
BEGIN HEADER
source:http://www.timetraveldevices.com/products/detail.php?product_id=<intrange:1:10000>
...
```

The above tells Mine The Web to apply the spider for 10,000 URLs, starting from http://www.timetraveldevices.com/products/detail.php?product_id=1 until http://www.timetraveldevices.com/products/detail.php?product_id=10000

Running your spider from the command-line

You can run your spiders from the command-line. This is useful if you want to have a spider run automatically (e.g. as a cronjob). To run your spider from the command-line, you need to know the spider ID of your spider. You can find this out easily by hovering your mouse over the Execute or Edit links in the main spider page. The URL that displays in the status bar of your browser should show you what the value for filter_id is. The filter_id is the spider ID.

Then, all you have to do at the command-line is change to the directory where you installed Mine The Web, and execute:

```
php cr_engine.php '&filter_id=377&nohtml=1&storenow=1&ignoreduplicatewarnings=0'
```

Substitute the value 377 above for whatever the value of your spider ID is. If your PHP installation has the command line interface (CLI) installed, you can use this syntax instead:

```
php cr_engine.php 377 1 1 0
```

The first argument after cr_engine.php is the spider ID, the second is the nohtml setting which should be 1, the third is the storenow setting which should also be 1, and the last argument is the ignoreduplicatewarnings setting which can be either 0 or 1 according to your preferences.

Congratulations!

Congratulations! You have successfully completed the Spider Script Tutorial for Mine The Web. You are now ready to fully utilize Mine The Web to satisfy your content retrieval needs. If you would like to see more spider scripting examples, check out our Spider Script examples page. Otherwise, go ahead and play around with our demo site to test your Spider Scripting skills. And don't forget, if you have any further questions, feel free to contact us at support@ohcraps.com.

<< Previous Page

To try out the scripts below and view the results for yourself, copy the script and input it into our demo site.

Most of the examples below use real websites, thus if the actual website goes down, the script will not work. To report such a problem, please send an e-mail to support@shuetech.com.

- Download.com Download Monitor Script

  This script demonstrates how a software publishing company can use Mine The Web to monitor the popularity of their software products on download.com, a popular software download site. Using Mine The Web gives them instant insight on how their products are performing for the week and even in comparison to competitor's products.

  In this example, CRT and SecureCRT are products belonging to a company called Van Dyke, and EMU is a product from a competing company. The script will show Van Dyke managers how many downloads they are getting (thus how popular they are) compares to EMU. Note that this is a LIVE example, meaning everything that you see when you run this script is REAL data, not make believe. You'll actually see how popular Van Dyke's products are in real time.

- Soccer Betting Odds in Singapore

  This script collects betting odds for soccer fans from a Singaporean website. Note that we hold no responsibility for the data provided, it is provided solely as a convenience.

Back to Examples Index

```
################
# CRT
################
BEGIN HEADER
source:http://download.com.com/3000-2155-10132564.html?tag=lst-0-1
CRto:2

BEGIN INFORMATION
Product:TEXT:CRT
Publisher:TEXT:Van Dyke Software

BEGIN ACTION
startafter:
    <td valign="middle" class="a2"><b class="dg">Downloads:</b></td>
    <td valign="middle" class="a2">|
endat:
</td>|
pattern:
</b><$1>$2</b>|
definition:
$1:
$2:Downloads:INT

BEGIN DO

################
# SecureCRT
################
BEGIN HEADER
source:http://download.com.com/3000-2155-10132571.html?tag=lst-0-4
CRto:2

BEGIN INFORMATION
Product:TEXT:SecureCRT
Publisher:TEXT:Van Dyke Software

BEGIN ACTION
startafter:
    <td valign="middle" class="a2"><b class="dg">Downloads:</b></td>
    <td valign="middle" class="a2">|
endat:
</td>|
pattern:
</b><$1>$2</b>|
definition:
$1:
$2:Downloads:INT

BEGIN DO

################
# EMU
################
BEGIN HEADER
source:http://download.com.com/3000-2155-5068665.html?tag=lst-0-2
CRto:2

BEGIN INFORMATION
Product:TEXT:EMU Terminal Emulator
Publisher:TEXT:Standard Networks Inc.
```

```
BEGIN ACTION
startafter:
    <td valign="middle" class="a2"><b class="dg">Downloads:</b></td>
    <td valign="middle" class="a2">|
endat:
</td>|
pattern:
</b><$1>$2</b>|
definition:
$1:
$2:Downloads:INT

BEGIN DO
```

Back to Examples Index

```
################
# SLeague
################
BEGIN HEADER
source:http://www.singaporepools.com.sg/webodds.htm
CRto:2

BEGIN INFORMATION

BEGIN ACTION
startafter:
<tr valign=top><td align=left bgcolor=\#CCCCCC width=500 colspan=3><img src=images/tran.g
<tr valign=top>|
endat:
</tr>
</td></tr>|
pattern:
$a<td align=left width=220>
<center><table border=0 cellspacing=0 cellpadding=1>
<tr valign=top><td colspan=3 align=center bgcolor=$a>$aMatch $1<br>$2</a><br>$3</b></font
<tr><td colspan=3 height=5></td></tr>
<tr>
<td bgcolor=$a height=12 colspan=3 align=center><font color=$a face=arial size=-1>1X2
</tr>

<tr>
<td$a/td>
<td width=3>$a</td><td align=right width=50>$4</font></td>
</tr>

<tr>
<td width=200 align=left bgcolor=$a>DRAW                                </td>
<td width=3 bgcolor=$a>$a</td><td align=right width=50 bgcolor=$a>$5</font></td>
</tr>

<tr>
<td$a/td>
<td width=3>$a</td><td align=right width=50>$6</font></td>
</tr>

<tr><td colspan=3 height=5></td></tr>
<tr>
<td bgcolor=$a height=12 colspan=3 align=center><font color=$a face=arial size=-1>1/2 GOA
</tr>

<tr>
<td$a/td>
<td width=3>$a</td><td align=right width=50>$7</font></td>
</tr>

<tr>
<td$a/td>
<td width=3 bgcolor=$a>$a</td><td align=right width=50 bgcolor=$a>$8</font></td>
</tr>

<tr><td colspan=3 height=5></td></tr>
<tr>
<td bgcolor=$a height=12 colspan=3 align=center><font color=$a face=arial size=-1>TOTAL G
</tr>
```

```
<tr>
<td width=200 align=left>0  GOAL                          </td>
<td  width=3>$a</td><td  align=right  width=50>$9</font></td>
</tr>

<tr>
<td width=200 align=left bgcolor=$a>1  GOAL                            </td>
<td  width=3  bgcolor=$a>$a</td><td  align=right  width=50  bgcolor=$a>$a</font></td>
</tr>

<tr>
<td width=200 align=left>2  GOALS                          </td>
<td  width=3>$a</td><td  align=right  width=50>$a</font></td>
</tr>
$a</table></td>|

definition:
$1:MatchNum:INT:StripHTMLTags()
$2:Match:TEXT
$3:Date:TEXT:StripWhitespace()
$4:Odd1:FLOAT:StripWhitespace()
$5:Odd2:FLOAT:StripWhitespace()
$6:Odd3:FLOAT:StripWhitespace()
$7:Odd4:FLOAT:StripWhitespace()
$8:Odd5:FLOAT:StripWhitespace()
$9:Odd6:FLOAT:StripWhitespace()
$t:Odd7:FLOAT:StripWhitespace()
$u:Odd8:FLOAT:StripWhitespace()
$a:

BEGIN DO
```