

Einführung in

StockTray-Datenquellen

Um die grundsätzliche Funktionsweise von Datenquellen zu verstehen, werfen Sie am besten einen Blick in die bereits unter STOCKTRAY\SOURCES mitgelieferten Quelldateien. Quickenfind.src (U.S. Aktien aus Quicken.com) oder dbkfind.src (Deutsche Aktien aus deutsche-bank.de) sind hier gute Punkte für den Einstieg.

Wenn Sie eigene Datenquellen erstellen möchten und trotz dieses Dokumentes noch Fragen haben (was alles andere als unwahrscheinlich ist), schreiben Sie uns einfach eine Email an m.schmidt@emtec.com.

StockTray Datenquellen

Einführung

StockTray bezieht Aktienkurse von Webseiten. Webseiten sind in einem speziellen Format, HTML genannt, gespeichert, welches sowohl die angezeigten Rohdaten als auch Steuerzeichen zum Formatieren der Anzeige enthält.

Da es unendlich viele Möglichkeiten gibt, Aktienkurse in HTML anzuzeigen, gibt es für StockTray keine "Standardlösung", um Aktien-Informationen aus Webseiten auszulesen. Stattdessen benötigt StockTray eine Anweisung, um schrittweise Daten mit dem gewünschten Inhalt (also die Aktienkurse) auszulesen, sowie Daten mit nicht erwünschtem Inhalt (die Steuerdaten für die Anzeige) zu überlesen. Derartige Anweisungen heißen Datenquellen.

Datenquellen werden als reine Textdateien im SOURCES-Verzeichnis von StockTray gespeichert.

Um die grundsätzliche Funktionsweise von Datenquellen zu verstehen, werfen Sie am besten einen Blick in die bereits unter STOCKTRAY\SOURCES mitgelieferten Quelldateien. `Quickenfind.src` (U.S. Aktien aus Quicken.com) oder `dbkfind.src` (Deutsche Aktien aus deutsche-bank.de) sind hier gute Punkte für den Einstieg.

Abschnitte einer Datenquelle

Eine Datenquelle besteht aus verschiedenen Abschnitten, die Informationen für verschiedene Verwendungszwecke enthalten. Hier ein Ausschnitt aus einer Datenquelle:

```
[Symbol]=yahcomfind
```

```
[Country]=1
```

```
[Name.German]=U.S. Kurse - Yahoo.Com
```

```
[Name.English]=U.S. Quotes via Yahoo.Com
```

```
[InfoShort.German]=U.S. Kurse in USD. Suche nach Ticker-Symbol. Kurse von Reuters. Biete gute Charts und Firmenprofile (engl.).
```

```
[InfoShort.English]=U.S. Quotes in USD. Search based on Ticker Symbol. Quotes provided by Reuters. Offers concise charts and company profile.
```

```
[InfoLong]=...
```

```
[SearchFor]=Ticker Symbol
```

```
[SearchTransform]=uppercase
```

```
[BasedOn]=http://quote.yahoo.com/
```

```
[Host]=quote.yahoo.com
```

```
[Doc]=q?s=$(SYMBOL) &d=t
```

```
[ParserScript]=...
set "decimal", "/"
set "thousand", ",",
set "currency", "USD"
...
```

Jeder Abschnitt hat eine spezielle Bedeutung, die nachfolgend erklärt ist:

Symbol: Ein eindeutiger Text, der die Datenquelle benennt.

Country: Der Ländercode (die nationale Telefonvorwahl) des Marktes, der von dieser Datenquelle abgedeckt wird, z.B. 49 für Deutschland oder 1 für USA.

Name: Der Name mit einer Kurzbeschreibung der Datenquelle. Dieser Name wird im Programm in der Liste aller möglichen Datenquellen angezeigt, aus der der Anwender wählen kann. Diese Sektion kann und sollten in deutscher und englischer Sprache eingetragen werden (Name.German und Name.English).

InfoShort: Eine kurze (ca. 20-30 Worte) Beschreibung der Datenquelle. Diese Beschreibung wird angezeigt, wenn der Anwender die Datenquelle in StockTray selektiert. Die Beschreibung kann mehrsprachig angegeben werden (English/German).

InfoLong: Eine optionale längere Beschreibung oder weiterführende Informationen zur Datenquelle. Dieser Inhalt wird angezeigt, wenn der Anwender nach Auswahl der Datenquelle den Knopf 'Weitere Informationen' drückt. Auch diese Beschreibung kann mehrsprachig angegeben werden (English/German).

SearchFor: Eine Zeichenfolge, die den Kurswert beschreibt, der abgefragt werden kann. Diese Text wird als Überschrift für das Eingabefeld verwendet, in das der Anwender die Aktien-Nummer eingibt. In den USA wird üblicherweise der Text *Ticker Symbol* verwendet, in Deutschland der Text *WKN*.

SearchTransform: Eine optionale Umsetzung der Such-Eingabe des Anwenders in Groß- oder Kleinbuchstaben. Mögliche Wert sind die Worte *uppercase* und *lowercase*.

BasedOn: Eine komplette URL (beginnend mit "http://"), die auf die Webseite verweist, auf der die Datenquelle basiert. Hier wird üblicherweise die Hauptseite bzw. die Suchseite des jeweiligen Anbieters eingetragen. Die hier angegebene Webseite wird angezeigt, wenn der Anwender nach Auswahl der Datenquelle auf den Knopf "Zeige Webseite, auf der diese Quelle basiert" drückt.

Host: Der Internet-Host, der für die Kursabfrage der Datenquelle verwendet wird.

Doc: Der variable Teil der URL, der auf die entsprechende Seite mit den gesuchten Kursen verweist. Üblicherweise enthält dieser Teil den Text $\$(SYMBOL)$, der gegen die Aktien-Nummer ersetzt wird, die der Anwender für die Suche eingetragen hat. Im obigen Beispiel würde der variable URL-Teil bei Eingabe des Abfragetextes YHOO den die Webseite *q?s=YHOO&d=t* von *quote.yahoo.com* (also *http://quote.yahoo.com/q?s=YHOO&d=t*) abfragen.

ParserScript: Ein Script, das die empfangene Webseite analysiert und die von StockTray benötigten Informationen anhand spezieller Bezeichner zur Verfügung stellt. Die verwendete Script-Sprache wird später in diesem Dokument beschrieben (es handelt sich hierbei um die gleiche Sprache, die auch in den EmTec-Produkten InfoTray und Yonc verwendet wird).

StockTray-Erweiterungen der Script-Sprache

Wenn Sie mit StockTray Scripts arbeiten, ist es sehr empfehlenswert, die Option zum Debugging in den StockTray-Optionen zu aktivieren. Ein Blick in die von StockTray während einer Abfrage erzeugte Logdatei löst so manches Problem.

Sollten Sie trotz dieser Anleitung noch Fragen haben (was alles andere als unwahrscheinlich ist), dann schreiben Sie uns einfach eine Email an m.schmidt@emtec.com.

Vorab-Verarbeitung

Vor der Ausführung eines StockTray-Scripts werden die nachfolgenden Zeichenketten gegen die im dazugehörigen StockTray-Dialogs eingetragenen Werte ersetzt.

\$(SYMBOL)"	Wird mit den Inhalt des Wertpapier-Feldes ersetzt (üblicherweise gegen das Ticker-Symbol bzw. die WKN-Nummer des abzufragenden Wertes)
\$(SEARCHWHERE)	Wird durch den Inhalt des Feldes ersetzt, in dem der Benutzer den Börsenplatz angegeben hat.
\$(USERNAME)	Wird mit den Wert ersetzt, der vom Anwender unter dem Menüpunkt Datei->Zugangsdaten angegeben wurde bzw. der eingegeben werden muss, wenn die verwendete StockTray den Name/Passwort-Dialog öffnet.
\$(PASSWORD)	Angabe des Passworts für zugangsgeschützte Seiten (siehe \$(USERNAME)).

Wenn also beispielsweise ein Script die Zeile

```
findline "$(SYMBOL)"
```

enthält, und der Benutzer dieses Script in einer Datenquelle verwendet, die für den entsprechenden Eintrag das Wertpapier YHOO abfragen soll, führt StockTray stattdessen den Befehl

```
findline "YHOO"
```

aus.

Set / Outputto, Say

Da StockTray aus einer Webseite gleich mehrere Informationen liefern muss (z.B. den aktuellen Kurs, die prozentuale Änderung zum Vortag usw.), wurde für diesen Zweck das Vorgehen gewählt, die entsprechenden Werte in Form von Ausgabe-Variablen an StockTray zu übermitteln.

Zum einen kann eine Variable mit dem Befehl `set` auf einen festen Wert gesetzt werden, z.B. `set "decimal", "."`

In der Mehrzahl der Fälle werden Variablen aber per Umleitung des SAY Befehls auf einen entsprechenden Wert gesetzt. Der Befehl OUTPUTTO legt fest, dass alle nachfolgenden SAY Befehle in die unter outputto angegebene Variable gespeichert werden. Nachfolgendes Beispiel füllt die Variable "date" mit den zwei darauffolgenden say-Befehlen:

```
outputto "date"
sayuntil "<"
say "|D.M.Y"
```

Hier eine Liste aller möglichen Variablen, die per SET bzw. OUTPUTTO gefüllt werden können. Für den Fall, dass diese Werte nicht gesetzt werden, zeigt StockTray in den korrespondierenden Spalten den Text "<n/a>" an.

decimal"	Das Zeichen,das bei Zahlen als Dezimaltrennzeichen verwendet (z.B. "," oder ".")
thousand"	Das Zeichen,das bei Zahlen als Tausendertrennzeichen verwendet (z.B. "," oder ".")
last	Aktueller Kurs
currency	Währung des Kurses.
date	Datum des Kurses in der Form <datum> <format>. Format kann D, M, Y or Z enthalten (diese stehen für Tag, Monat, Jahr bzw. 3stelliger Monat als Zeichen), z.B. "01.09.00 D.M.Y" or "01-Jan-2001 D-Z-Y"
time	Zeit des Kurses in der Form <zeit> <format>. Format kann h, m, s enthalten (diese stehen für Stunde, Minute, Sekunde), z.B. "12:04 h:m" oder "14.33.07 h.m.s"
bid	Akt. Geldkurs
ask	Akt. Briefkurs
kassa	Kassakurs heute
yesterday	Gestriger Schlußkurs
change	Änderung (prozentual) seit gestern
abschange	Änderung (absolut) seit gestern
volume	Anz. gehandelter Aktien heute
open	Eröffnungskurs heute
high	Heutiges Hoch
low	Heutiges Tief
high52weeks	Jahreshoch
low52weeks	Jahrestief
urlchart	URL der mit http:// beginnt und den Chart abruf
urldaychart	URL der mit http:// beginnt und einen Intraday Chart abruf
urlprofile	URL der mit http:// beginnt und Firmeninformationen liefert
market;	Börse von der der Kurs stammt, z.B. Frankfurt, NYSE, Nasdaq, London, etc.

Extern "geturl", "<variable>"

Dieser Befehl lädt eine HTML-Seite innerhalb eines Scripts. Diese Funktionalität wird

üblicherweise dazu verwendet, um einem in einer HTML-Seite angegebenen Link zu folgen.

Um eine derartige Seite zu laden, muß eine Variable mit der vollständigen URL gefüllt und der Name der Variable dem `extern-geturl` Befehl übergeben werden, z.B.

```
outputto "parm"  
findinline "Quote-Details:"  
findinline "href=\\"  
sayuntil "\\"  
extern "geturl", "parm"
```

Nach Verarbeitung von `extern-geturl` steht das Script auf der ersten Zeile des Ergebnisses (Webtext) dieses Befehls.

Extern "post", "<variable>"

Auf manchen Seiten ist es notwendig Daten per Html-POST an den Server zu übergeben, um Eingaben in einem Formular zu simulieren.

Mit *extern post* werden die Daten aus einer Variable (die vorher mit `set` oder `outputto` gefüllt werden muß) an einen Webserver gesandt. Dem `extern post` Kommando wird dabei als zweiter Parameter der Name der Variable übergeben.

Die Variable muß Daten in der Form "`<url>|<data>`" enthalten, z.B.
"`http://qoteserv.com/getsym.asp|symbol=CSCO`" (hiermit würde der Wert `symbol=CSCO` an den Webserver `http://qoteserv.com/getsym.asp` geschickt.)

Dies könnte in einem Script etwa so aussehen:

```
set "parm", "http://qoteserv.com/getsym.asp|symbol=$(SYMBOL)"  
extern "post", "parm"
```

Nach Verarbeitung von `extern-post` steht das Script auf der ersten Zeile des Ergebnisses (Webtext) dieses Befehls.

Script Language to Analyze Web-Pages (Parser)

This part of the document is taken from EmTec's Yonc product.. A German version for the document is available when you download Yonc from www.emtec.com

Introduction

The parser is a tool that is designed to take instructions (which are called a "parser script") to retrieve a few words from a text that contains multiple lines. Using this tool you can work your way through html code (hypertext markup language -- the code that makes up WWW documents) to extract a few words or numbers that make up stock quotes.

Basics:

Before we begin with a sample from the world wide web, let us look at the basic principles of the parser by looking at an example that is not related to the WWW but which is easier to start with and with which it is easier to demonstrate things.

So just take the following text, which is the output from an hard disk checking program, as an abstract example. Let us assume that it were listed on a WWW page and that we would need something of it for a stock quote.

```
The type of the file system is FAT.
Volume DUKE_DRIVE_C created 10/16/96 7:44 PM
Volume Serial Number is A2FA-3C14
CHKDSK is verifying files and directories...
File and directory verification completed.

361684992 bytes total disk space.
    114688 bytes in 6 hidden files.
    368640 bytes in 44 directories.
286965760 bytes in 1157 user files.
    1335296 bytes in extended attributes.
 72900608 bytes available on disk.

    8192 bytes in each allocation unit.
 44151 total allocation units on disk.
   8899 allocation units available on disk.
```

Now -- for the sake of making our first example simply (which actual web pages are not) -- let us demonstrate how to retrieve the number of hidden files in for a stock quote. This means that you need a way to tell StockTray which parts to pick from this raw data.

How to Start

First of all you need to find a way to identify the lines which contain the interesting bits of information. The StockTray parser sees output as lines and characters and you tell the parser how to move from the start to the places you want to display. StockTray uses a pointer which is positioned at the beginning of the file and which can be moved with several commands. So how do we move this pointer to the number of hidden files?

If it starts at the top we can either move it down seven lines or we can tell it to move down until it finds the text "hidden files". To do the first we would either use the command `MoveLine 7` or `GotoLine 8` -- this either means go down seven lines from where you are or go to the eighth line from the top of the text. To do the search, the command would be `FindLine "hidden files"`, which – well – finds the next line from where you are that contains the text "hidden files". In all cases the pointer would be moved to the first character of the target line. From there we could tell the pointer to move 19 steps to the right (`MoveChar 19`) or to move to the 20th character in the line (`GotoChar 20`) or to position the pointer after the text "in " (`FindInLine "in "`). All of these commands would result in the pointer being moved to the first digit of the number of hidden files. Please note the difference between the `FindLine` and `FindInLine` commands: The earlier goes through lines from where you are to find a text and places the pointer at the beginning of the line, while the latter looks within the line where the pointer is and positions it after the found text.

Now that we are where we want, we need tell StockTray to send data to our stock quote. To do this, a `Say` command is used, but we have to find a way to tell it what to say and what not. In this case it is relatively easy, because we want to output a number, and so we use the `SayNextNumber` command. An alternative would be the `SayUntil " "` command, which would output everything until a space character is found.

So, a script to display the number of hidden files from the above text would look like this:

```
FindLine "hidden files"  
FindInLine " in "  
SayNextNumber
```

You will notice that the example used the Find-Commands rather than the Move- or Goto-Commands. Whenever you have a chance to use a Find command, please do so, because WWW pages tend to be changed. Using a script that relies on Find-Commands is more likely to survive a change in the raw data than one that relies on absolute positions.

Debugging

Doing all this only theoretically can be a bit tricky and if you make an error counting lines or characters you might end up with quite unexpected results. To check what the parser is doing, you can set the debug option in the program's main option

dialog. This will give you an output file which will show you step by step what the parser is doing and why you end up with a given output.

Many times you will even want to start your script just with a Debug command, to see what data you actually get for parsing and before you build your script step by step.

More Samples

Let us take this a bit further. Maybe you also want to see the total number of files together with some decoration, like `6 hidden / 1157 total`. To send the decoration to the output, you just use a `Say` command to print the text “ hidden / ” and “total”. Retrieval of the total number of files would work in similar to the way we found the number of hidden files. The complete script could look like this:

```
FindLine "hidden files"  
GotoChar 20  
SayNumber  
Say "hidden / "  
MoveLine 2  
GoToChar 0  
SayNumber  
Say " total"
```

Please note that, for the sake of demonstration, this script uses `Move` and `Goto` commands, rather using `Find` wherever it would be possible.

So much for the first steps. Let us now look at a real WWW example.

Retrieving Data from the World-Wide Web

To display data from the WWW you will have to look at the way web pages are stored (the so called HTML format) and, as said before, you will have to parse the HTML data with StockTray 's parser to retrieve the few words or numbers that you actually want to see.

A HTML file is a text document, that describes web pages. Most of the time it contains far more control data (enclosed in angle brackets, e.g. <TABLE>, <TD>, ...) than actual data (at first you will probably doubt if what you see is really the source of what your web browser displays so nicely, especially in large and complex pages).

Anyway, let's start with something simple. On

<http://language.perl.com/info/software.html> there is a web page that lets you download the current version of the popular PERL interpreter. At the time when this manual was written, it looked like

Acquiring Perl Software

Downloading the Current Version of the Interpreter

Just click to [download the latest Perl source](#) (currently Perl version 5.004_04) from a fast link. This version is a stable, production release (not beta) that compiles out of the box for virtually all flavors of UNIX (its native environment), plus VMS, OS/2, and 32-bit Windows platforms as well. Check out its [installation notes](#) for details (or see the INSTALL file in the src directory.) Read [the beta release notes](#) and/or [documentation](#).

...

Looking at the document source (select View, Source in your browser) will reveal something – err -- slightly different:

```
<HTML>
<HEAD><TITLE>Acquiring Perl Software</TITLE></HEAD>
<!-- begin header -->
<A HREF="http://perl-ora.songline.com/universal/header.map"><IMG SRC="http://perl-
ora.songline.com/graphics/header-nav.gif" HEIGHT="18" WIDTH="515" ALT="Nav bar"
BORDER="0" usemap="#header-nav"></A>

<!-- end header -->
<BODY>

<CENTER><H1>Acquiring Perl Software</h1></CENTER>

<!--
<H1><BLINK><FONT COLOR="RED">[FILE BEING EDITED]</FONT></BLINK></H1>
-->

<DL>
```

<P><DT><BIG>Downloading the Current Version of the Interpreter</BIG><DD>
Just click to
download the latest Perl source
(currently Perl version 5.004_04)
from a fast link.
This version is a
stable, production release (not beta) that
compiles out of the
box for virtually all flavors of U<small>NIX</small> (its native environment), plus
<small>VMS</small>, <small>OS/2</small>,
and 32-bit Windows platforms as well.
Check out its installation notes
for details (or see the INSTALL file in the src directory.)
Read the beta release notes and/or
documentation.

Now let us assume that you always wanted to see the version number of the current PERL release. When giving the HTML code a second look you will eventually see the line "(currently Perl version 5.004_04)". Now all you have to do is to write a short parser script that skips to that line and retrieves the number.

Here is the whole item:

```
FindLine "currently Perl version"  
FindInLine "<b>"  
SayUntil "<"
```

Finding the right URL

Finding the URL that contains the page that you want to get data from may be tricky as well. Sometimes when you type a URL to get to a page, the page contains so called frames which means that you actually see two or more Web-Documents merged into one (most pages with a navigation bar on the top or left side are frames).

Here are some tips on how to find what you are looking for:

- On non-framed pages, just look at the Location field in your Web Browser.
- Try to right click somewhere near the text you want to get (but not on a graphic). Most Web-Browsers will offer you to create a bookmark/favourite or an desktop-icon linking the page. After you created a bookmark/icon, edit it's properties. Normally you see the URL there.
Netscape 4.0: Use "Create Shortcut" from the right button menu within a page and just use the URL that's shown in the dialog that pops up.
Internet-Explorer 4.0: Use "Create Shortcut" from the right button menu and then right click the icon that was created on your desktop and select *Properties*.
- Try a right-click on a link that brings you to that page. You should also see an option to create a bookmark/favourite/shortcut. Do that and check it's properties.
- Before trying to access such a URL from Yonc, test if you get the expected

results by typing it directly into your Web-Browser or by just clicking your bookmark/favourite/desktop-shortcut. If you are locating data on framed Web pages, you should see a page without the navigation bar.

Limitations

Due to the ever increasing complexity of the WWW, sooner or later you will come across cases where data is graphical or part of a temporary frame or an Java- or Active-X control. In that case there will be no way to get it into Yonc. Most of the time this is true if you do not find the piece of data in the window that pops up when you select View-> Source in your web browser and/or if you cannot mark the text with the mouse to copy it to the clipboard.

Also, if the webmaster of the site from which you get your data is rearranging things your parser script will break. In the example above, this might happen, if the Perl site is reorganized (e.g. by renaming documents) or even if the text is slightly changed, so that Yonc does not find a line containing "currently Perl version" (because after a redesign it might just say "PERL V5.004_04"). So, if your data suddenly disappears or if you find a parser error message instead of the data that was still there yesterday, please first check if your data source is still there (and if it is still the same).

Another Web Example (Stock Prices)

The Deutsche Bank in Frankfurt maintains a nice Web-Site with Stock data. The site uses frames. If you follow their "Marktinformationen" (Stock-Market-Info) link, you get a navigation bar on the left that allows you to view German stock lists or to search current stock prices for most stocks dealt in Frankfurt. Assuming you want to see the daily stock price of the Porsche shares, click on the left on "Einzelkurse", enter `Porsche` and click "Suchen" (Search). A table will appear. Fortunately, the text in the table can be marked with the mouse which indicates that we are dealing with HTML pages and not with an applet.

Rightclick somewhere on the page, create a bookmark/shortcut and edit the properties. You should see the following (or similar) URL (which can be used as the Internet data source in an InfoTray item):

```
http://db-tc.teledata.de:9002/db/searchres.html?searchfor=porsche&Search=Suchen&searchforb=2&searchforicat=5
```

Type that into a Web browser, to verify that it is the correct link. The stock table will appear without the navbar, so we are on the right track. Save the page or use View→Source in your Web browser to see what the raw data for the page looks like. It is a plain table with a lot <TD> entries and somewhere in the middle you find a line with the name of the shares and in the lines below you see the corresponding data. A script that will parse the document to retrieve the current stock price and the change in percent (compared to the day before) looks like this:

```
FindLine "#DDDDDD"  
MoveLine 3
```

```

FindInLine "SIZE=2>"
SayUntil "<"
say " ("
FindLine "color="
FindInLine ">", 4
Sayuntil "<"
Say " at "
Moveline 3
FindInLine ">", 3
SayUntil "<")
Say ")" "

```

Note that the findline command does not actually try to find the name PORSCHE but some other significant text near that. The idea behind it was to create a script that could be used for other stocks as well (i.e. to use the same script for all your stocks).

Yet Another Web Example (EmTec Products)

Here is a part of EmTec's main web page (www.emtec.com/index.html)

```

<!-- :set ts=2 -->
<html>
  <head>
    <meta name="description" content="EmTec Innovative Software">
    <meta name="keywords" content="">
    <meta name="author" content="Helmut Heidenreich">
    <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
    <title>
      EmTec Innovative Software
    </title>
  </head>
  <body background="images/wall.gif" bgcolor="#FFFFFF" text="#000000"
    alink="#001080" link="#001080" vlink="#801080">

```

[some lines deleted to save space]

```

  <table border=0 cellpadding=3 cellspacing=3>
    <tr>
      <td width=100>
        <br>
      </td>
      <td width=120 bgcolor="e0e0e0">
        <font size=-1>
          <a href="zoc/index.htm">
            </a>
            <a href="zoc/index.htm">ZOC V3.11</a><br>
        </td>
      <td bgcolor="e0e0e0">
        <font size=-1>
          <a href="stocktray/index.htm">
            </a>
            <a href="stocktray/index.htm">StockTray V0.92 (Beta!)</a><br>
        </td>
    </tr>
  </table>

```

```

<td width=10>
<br>
</td>

<td bgcolor="e0e0e0">
<font size=-1>
<a href="moony/index.htm">
</a>
<a href="moony/index.htm">Moony V1.05</a><br>
</td>
<td bgcolor="e0e0e0">
<font size=-1>
<a href="yonc/index.htm">
</a>
<a href="yonc/index.htm">Yonc V0.90 (Beta)</a><br>
</td>
</tr>
<tr>
<td width=10>
<br>
</td>

<td bgcolor="e0e0e0">
<font size=-1>
<a href="infotray/index.htm">
</a>
<a href="infotray/index.htm">InfoTray V1.01</a>
</td>
<td bgcolor="e0e0e0">
<font size=-1>
<a href="download.html">
</a>
<a href="download.html">Direkt-Download Seite</a><br>
</td>
</tr>
</table>

```

[more lines deleted to save space]

```

</body>
</html>

```

The part that is shown here contains the links to the product's sections together with the current version numbers. Here is a script that displays the version numbers of all EmTec products:

```

FindLine "zoc/index.htm"
FindInLine ">"
SayUntil "<"
say " / "
FindLine "moony/index.htm"
FindInLine ">"
SayUntil "<"
say " / "
FindLine "yonc/index.htm"
FindInLine ">"
SayUntil "<"

```

List of Parser Commands

Command	Parameter(s)	Description
GotoLine	N	Go to N'th line (counting from top)
MoveLine	N	Move down/up N lines (starting from current position)
FindLine	Sn	Find line with first or N'th occurrence of S (starting from the current position)
GotoChar	N	Skip to the N'th character in the current line
MoveChar	N	Move right/left N characters
FindInLine	Sn	Find the next/N'th occurrence of S within the current line
SkipChars	S	Move pointer forward and skip all characters which are listed in S.
Say	S	Send S to output
SayUntil	S	Send everything until S to output
SayNextWord	-	Send next word (all alphabetic characters) to output
SayNextNumber	-	Send next numeric to output
SayNChars	N	Send next N characters to output
If	S	Check for occurrence of S on current position
IfNot	S	Check for absence of S on current position
Else	-	Else branch of an If operation
Endif	-	End of an If operation
Debug	Ss	Debug output, S= "on" or "off", s is an optional file name.
OnError	Ss	Send S to output, if an error occurs, s is an optional file name

N	Required numeric parameter
S	Required string parameter (in quotes)
n	Optional numeric parameter
s	Optional string parameter (in quotes)

Details about what the commands are doing exactly can be found in Yonc's online help file.