

# WebGrab+Plus

## Advanced XMLTV EPG Grabber

By Jan van Straaten (jan\_van\_straaten@hotmail.com)

(Document revision 2/12/2011, reflects WebGrab+Plus version 1.0.9)

Special thanks to Paul Weterings for his WebGrab from which it started  
and for his website ServerCare in which it finds a Web-home.  
<www.servercare.nl>

What's in this document:

For everyone new to this program: Read page 3 and 4 (upto chapter 4.2) , and Appendix B

The rest of this document is for everyone willing to develop a SiteIni file or simply want to know more than the basics.

## **Table of Content:**

### **1.Xmltv Elements supported**

- 1.1 *Single* - versus *multiple* - value xmltv elements

### **2.The grabbing, show update process and update modes**

- 2.1 The show update process
- 2.2 The update modes

### **3. Configuration file (WebGrab++.config.xml)**

### **4. SiteIni file**

- 4.1 SiteIni file Parts
- 4.2 The SiteIni file basics
  - 4.2.1 scrubstrings
  - 4.2.2 ElementNames
  - 4.2.3 Action specifiers
  - 4.2.4 Types
    - 4.2.4.1 Type *url*
    - 4.2.4.2 Types *single* and *multi*
  - 4.2.5 Arguments
    - 4.2.5.1 *includeblock* and *excludeblock*
    - 4.2.5.1 *separator*
    - 4.2.5.2 *max*
    - 4.2.5.3 *include* and *exclude*
    - 4.2.5.4 *debug*
    - 4.2.5.5 Dedicated Arguments *lang*, *force*, *sort*, *timespan*, *(section)separator*
- 4.3 General Site dependant data
- 4.4 Data that WebGrab+Plus needs to compose the url's to download pages
  - 4.4.1 *url\_index*
    - 4.4.1.1 *urldate* syntax
    - 4.4.1.2 *subpage* syntax
  - 4.4.2 other url elements
- 4.5 Data that WebGrab+Plus needs to scrub xmltv elements from the downloaded pages.
  - 4.5.1 The elements scrubbed from the index page
  - 4.5.2 The elements scrubbed from the show detail page
  - 4.5.3 Special elements
- 4.6 Operations : Optional data that allows modification of the elements.
  - 4.6.1 Notes and examples of the effects of *modify*
    - 4.6.1.1 The order of the actions
    - 4.6.1.2 Multiple value elements and modify
    - 4.6.1.3 Expression with indices
  - 4.6.2 Conditional arguments
    - 4.6.2.1 (pre) Conditional arguments
    - 4.6.2.2 (post) Conditional arguments
  - 4.6.3 Loops
  - 4.6.4The Modify Commands
    - 4.6.4.1 Replace
    - 4.6.4.2 Remove
    - 4.6.4.3 Substring
    - 4.6.4.4 Addstart and Addend
    - 4.6.4.5 Calculate
      - 4.6.4.5.1 # Count and Length
      - 4.6.4.5.2 @ Index-of
      - 4.6.4.5.3 Date and time calculations
    - 4.6.4.6 Cleanup
  - 4.6.5 Examples of operations

### **5. Tricks**

- APPENDIX A XMLTV Elements, sample xmltv file syntax
- APPENDIX B Example WebGrab++.config.xml file
- APPENDIX C Example SiteIni files
- APPENDIX D Supported Element names in a siteini file

# WebGrab+Plus , an advanced XMLTV EPG Grabber

The program grabs EPG data from TV Guide internet sites and

- can grab from multiple sites , programmable by user through a siteini file
- very fast through incremental grabbing (only changed and new shows grabbed)
- programmable through editing commands that enable changing, filtering, adding, moving, removing (parts) and calculating of the xmltv elements.

## 1.Xmltv Elements supported

For an overview of the xmltv elements supported see APPENDIX A and APPENDIX D

### 1.1 Single- versus multiple- value xmltv elements:

According to the xmltv specification, some elements can have more than one value in the xmltv file. We distinguish *single value* xmltv elements (e.g. description) and *multiple value xmltv* elements (e.g. category, actor). WebGrab+Plus treats them differently. (for examples see 4.2.4 Types)

Note that the element 'title' is a *single value element* but the program supports a second version of the same title (titleoriginal) with different 'lang=' attributes. (See 4.2.5.5 argument *Lang*)

## 2.The grabbing, show update process and update modes:

### 2.1 The show update process

Assuming a previous xmltv listing exist (e.g. of yesterday), the program reads this and stores it as a target for update and as reference of what shows have to be changed or added. If no xmltv listing exists, the program creates a new one. Before grabbing show details, the program determines if the existing show in the xmltv listing is still valid or needs an update. For that it connects to the TV Guide website and grabs the so called index pages (the html pages that contain an overview the scheduled shows per timespan (e.g. day or several days)). It then compares the shows listed there (channel, start and stop times and title) with shows in the existing xmltv listing. As a result of this comparison the following situations occur:

- **same (.)**, no update .The show in the index page is considered the same as the one in the existing xmltv listing.
- **changed (c)**, update. The index show is different from the xmltv show but they have overlapping or equal time span.
- **gab (g)**, insert . The index show fits in a time gab of the xmltv listing.
- **new (n)**, add . The index show is new, it will be added to the end (or to the beginning if that is the case) of the xmltv listing.
- **repair (r)**, update. This is a special situation that occurs if errors or overlapping shows are detected in the xmltv listing. The program will try to solve this by remove and update.

When the program runs, these resulting situations for each show are printed in the command window like this (the *iiii* indicates 4 days of index pages downloaded):

```
iiii.....g.....CCC.....C.C.....g....r.....nnnnnnnnnnnnnnnnnnnnnnnnnnnnnn
```

The comparison of the show title in the index page (*index\_title*) and the one in the xmltv file is rather complicated and tricky. This is due to the fact that the *index\_title* frequently differs from the one in the show detail page to a certain extend. Differences can be due to abbreviation of long titles, different use of punctuation characters and combination of title with other elements in the *index\_title* (like category and subtitle). The program deals with all those differences through a weighted comparison. The result of this comparison is a 'title match factor', which , roughly, is the biggest percentage of 'matching' words between the two titles in any of the elements of the *index\_title*. If this title match factor is less than the value for it in the siteini file (see 4.3) the show is considered - not same - and a show

update is started.

For that it will grab the show details from the show detail html page(s) of the TV Guide website if provided by it.

## 2.2 The update modes

The program supports a variety of update modes. The preferred and most efficient is 'incremental' (i) Works as described above for all shows in the index page. In this mode the download time is minimized to the minimum.

Other update modes are:

'light' (l) which is incremental but forces a re-grab of all shows for 'today' ,  
'smart' (s) is the same with a forced re-grab for today and tomorrow ,  
'full' (f) not incremental, forces a full re-grab of all days requested.

Index-only mode:

Besides and independent from the modes mentioned above is a special grabbing mode 'index-only' that is automatically selected by the program if no elements need to be scrubbed from the show detail page. (see also 4.5) This mode is 'superfast' but seldom useful because most sites provide very little show data on the index page. But if you are satisfied with just start and stop times and a title it's there. Occasionally there is a site with richer data on the index page (like tvguide.co.uk). Some sites list only details on the index page or provide only more detailed information for some shows on detail pages. The program automatically recognizes these cases.

## 3. Configuration file (WebGrab++.config.xml)

This file supplies all TV Guide website independent settings for WebGrab+Plus. Among them are - update mode - time span - and, most important, a list of channels to grab. Each channel for which epg data in the xmltv listing is requested needs to be added to this channel list. The channel data in this list consists of the update mode (see 2.2) , the site to get the data from (see 4) the site-id (the channel id of the site, see 4.4.1), the xmltv\_id (the id by which xmltv recognises the channel) and the channel display name.

A typical WebGrab++.config.xml file is listed in APPENDIX B. It also provides the explanation of all the settings. The file is self explanatory.

## 4. SiteIni file

For each TV Guide website that is entered in the channel list of the config file (see above) a SiteIni file is required to supply WebGrab+Plus with site dependant settings. The name of this file is directly related to the value of the site attribute in the channel list through the addition of .ini to this value. (e.g. channel list site attribute : tvuids.nl .. SiteIni file name : tvuids.nl.ini)

### 4.1 SiteIni file Parts

The data in this file consists of the following parts:

- General Site dependant data (see 4.3)
- Data that WebGrab+Plus needs to compose the url's to download pages (see 4.4)
- Data that WebGrab+Plus needs to scrub xmltv elements from the downloaded pages (see 4.5)
- Optional data that allows post modification of the scrubbed xmltv elements (see 4.6)

### 4.2 The SiteIni file basics

#### 4.2.1 scrubstrings

Most of the settings in this file relate to how WebGrab+Plus extracts ("scrubs") xmltv elements from the TV Guide website html pages. For that it uses (up to) 4 strings that should point to the beginning and the end of the element to scrub. Obvious are the 'element start' **es** and the 'element end' **ee** string. They represent the unique strings (e.g. html tags or parts of it) between which required the element is always located on the html page. In most cases such unique **es** and **ee** are unavailable because somewhere else in the html page the same strings

exist enclosing other data. In that case we need to separate the right **es** and **ee** pairs from the unwanted pairs. For that we use the block separators **bs**, block start **bs** and block end **be**. These should enclose a html region (block) in which **es** and **ee** enclose our wanted element and nothing else.

Consider the following sample html:

```
<div id = "detail-page">
  <div id = "program-content">
    <div id = "program-info">
      
      <h3>Basilisk: Serpent King</h3>
      <a class="channel-title" href="/TV/Guide/Channel/RTL+7/Today/">RTL 7</a>
      <div id = "program-desc-text">
        Amerikaanse actiefilm. Een team van archeologen ontwaakt een mythische slang die vernieling zaait. De enige manier om het wezen te stoppen is door een magische scepter te vinden.
      </div>
      <!-- Genre Subgenre Data -->
      <dl>
        <dt>Genre:</dt><dd>speelfilm</dd>
        <dt>Genre:</dt><dd>sequel</dd>
        <dt>Subgenre:</dt><dd>avontuur</dd>
        <dt>Duur:</dt><dd>90 min</dd>
        <dt>Regie:</dt><dd>Louie Myman</dd>
        <dt>Met:</dt><dd>Jeremy London, Wendy Carter, Griff Furst, Cleavant Derricks, Daniel Ponsky, Bashar Rahal</dd>
      </dl>
```

To scrub the title - *Basilisk: Serpent King*- we need **es**= `<h3>` and **ee**= `</h3>`. In fact if it is sure that `<h3>` tag is uniquely used to enclose the title we wouldn't need more than that. However even if that is the case on this (part of) html page, simple html tags like `<h3>` are seldom unique and thus it is more secure to use the block separators **bs**= `<div id = "program-info">` and **be** = `<a class`

It is a little different with the description, here **es**= `<div id = "program-desc-text">` and **ee**= `</div>`. Very likely this **es** is unique for the description, so we wouldn't need block separators.

Strings like **bs**, **es**, **ee** and **be** will be called scrubstrings in the remainder of this document.

The syntax in which the SiteIni file expects them is :

```
{type(optional arguments)|bs|optional es|ee|optional be}
```

or:

```
{type(optional arguments)|bs|optional es|optional ee|be}
```

- scrubstrings must be enclosed between curly brackets {}
- the individual entries are separated by a vertical line | character
- the first entry is a type specification, optionally followed by a list of arguments separated by spaces and enclosed between parenthesis (). Explained later in 4.2.4 & 4.2.5.
- !!! notice that **es** is optional! If **es** is left blank than **bs** takes its place!!
- !!! also either **ee** or **be** is optional. If **be** is left blank there is simply no other limit to the block other than end of html page or a next block. Leaving **ee** blank is a special case, most useful with `index_showsplit`, see 4.5.1, it will cut the block between **bs** and **be** in slices at **es**. The values of **ee** and **be** are allowed to be equal (pointing to the same string). This is sometimes helpful if no **be** is available following **ee**. It is advised to specify the value of **ee** for **be**, rather than to leave it blank. It results in a more efficient and secure scrub-

bing.

- !!! every character (with the sole exception of the vertical line |) in the **bs**, **es**, **ee** and **be** value is valid! Including spaces! So | | specifies a space and is different from || which is empty.

- **bs** and/or **es** may be left empty, like ||, in that case the block starts at the beginning of the html page and/or the element starts at the beginning of the block.

To complete a SiteIni scrub specification we need to add the xmltv element name and an action specifier :

**ElementName.ActionSpecifier{ScrubString}**

The scrub specifications for two scrubstrings from above for description and title respectively:

```
description.scrub {single|<div id = "program-desc-text">||</div>}
title.scrub {single|<div id = "program-info">|<h3>|</h3>|<a class>}
```

#### 4.2.2 ElementNames :

Element names in the siteini file have fixed values. Not all however are obligatory. A complete list of supported elements can be found in APPENDIX D.

Elements with a prefix *index\_* are scrubbed from the index page, the ones without a prefix are scrubbed from the show detail page. Notice that most elements can be scrubbed from either the index or the detail page. This depends on the actual content of these pages. It is allowed to have an element scrubbed from both pages, in that case the scrubbed values will be added in a way which depends on if it is a *multiple value* xmltv element or not. (see 1. and 4.2.4). In the case of a *multiple value element* they will be listed as separate elements, while when it concerns a *single value element* the values are merged. (for more explanation see 4.6.2.1)

The obligatory elements are either required for proper functioning of the program (*url\_index* and *urldate* to connect to the site, *index\_showsplit* to separate the show index parts, *index\_start* and *index\_title* for update decision making) or as a minimum for a meaningful xmltv output (*index\_start* and *title* or *index\_title*).

#### 4.2.3 Action specifiers :

Action specifiers are either *format* , *scrub* or *modify*. They specify what kind of action the program has to perform. Some elements, of type *url* (see 4.2.4), don't need one. See APPENDIX D for an overview.

#### 4.2.4 Types :

Type is *url*, *single* or *multi*

##### 4.2.4.1 type *url* :

Scrubstring specifications for this type have varying formats to build the url's to connect to the various site pages. (see 4.4)

##### 4.2.4.2 types *single* and *multi* :

Very often elements in html pages are divided in several paragraphs or otherwise split into parts such that no *single* pair of element separators (**es** and **ee**) enclose the element. Suppose the description in the html page looks like this:

```
<div id = "program-desc-text">
  <p>Amerikaanse actiefilm.</p>
  <p>Een team van archeologen ontwaakt een mythische slang die vernieling zaait. De enige manier om het wezen te stoppen is door een magische scepter te vinden.</p>
  <p>Geproduceerd in 1998</p>
</div>
```

In such a case we use type *multi* to instruct WebGrab+Plus to scrub all the elements within

the block with the specified element separators, like here `es = <p>` and `ee = </p>`

To illustrate the scrub results from this html with type `single`:

```
description.scrub {single|<div id = "program-desc-text">|<p>|</p>|</div>} will result in :  
  <desc lang="xx">Amerikaanse actiefilm.</desc>
```

While the same with type `multi` :

```
description.scrub {multi|<div id = "program-desc-text">|<p>|</p>|</div>} will result in :  
  <desc lang="xx">Amerikaanse actiefilm. Een team van archeologen ontwaakt een mythische  
  slang die vernieling zaait. De enige manier om het wezen te stoppen is door een magische scep-  
  ter te vinden. Geproduceerd in 1998</desc>
```

Notice that WebGrab+Plus adds the three description paragraphs together. This is due to the fact that the element description is a *single value* xmltv element. (see 1 and 4.2.2)

To illustrate what happens with a *multiple value* xmltv elements, consider the category.

In the html the genre and subgenre are the obvious choice for that. Xmltv doesn't specify a subgenre element, so we take them all together as category

```
  <!-- Genre Subgenre Data -->  
  <dl>  
    <dt>Genre:</dt><dd>speelfilm</dd>  
    <dt>Genre:</dt><dd>sequel</dd>  
    <dt>Subgenre:</dt><dd>avontuur</dd>  
<dt>Duur:</dt><dd>90 min</dd> </dl>
```

There are two genre entries in the html, with the same element separators, so we use type `multi` to grab them both.

```
category.scrub {multi|<!-- Genre Subgenre Data -->|<dt>Genre:</dt><dd>|</dd>|</dl>}
```

The result will be the following xmltv listing for category:

```
  <category lang="xx">speelfilm</category>  
  <category lang="xx">sequel</category>
```

Because category is a *multiple value* xmltv element the two are not joined to one xmltv element but listed as separate category elements.

To add the third category element , the Subgenre in the html, we use another feature of the SiteIni specification : For most siteini elements it is allowed to use more than just one scrub specification for the same xmltv element! (see APPENDIX D column `-multiple scrub-` which)

So we add:

```
category.scrub {single|<!-- Genre Subgenre Data -->|<dt>Subgenre:</dt><dd>|</dd>|</dl>}
```

The final result:

```
  <category lang="xx">speelfilm</category>  
  <category lang="xx">sequel</category>  
  <category lang="xx">avontuur</category>
```

#### 4.2.5 Arguments:

Arguments can be either/and `includeblock`, `excludeblock`, `separator`, `max`, `include`, `exclude`, `debug` and dedicated arguments `lang`, `force`, `sort`, `timespan`

##### 4.2.5.1 Argument `includeblock` and `excludeblock` :

If it is only possible to find blocks that, apart from the required information, contain unwanted information with the same element separators `es` and `ee` , these arguments can be used to select the correct blocks. The syntax:

```
  includeblock=bn1,bn2, .. ,bnn/tn -or- "string-1""string-2" .. "string-n"  
  excludeblock=bn1,bn2, .. ,bnn/tn -or- "string-1""string-2" .. "string-n"
```

- bn , the block number to include or exclude, starting with 1
- tn , the number of blocks for which the block numbers bn repeat

- "string" , include or exclude only the blocks that contain the "string". When more than one "string" is entered, the block selection is done by an 'or' function of the strings.  
Example : `includeblock="abc""def"` , the blocks included contain the string "abc" or "def" .  
When more than one "string" is entered separated by the char & , the block selection is done by an 'and' function.  
Example : `includeblock="abc"&"def"` , the blocks included contain the string "abc" and "def".
- All characters are allowed.
- The characters " ' { and ) need to be preceded by \ . So the string ("O'Neil {superhero}") must be entered as `"\(\\"O\'Neil \{superhero}\\""`

#### 4.2.5.2 Argument *separator* :

As example take a look at the actors :

```
<dt>Regie:</dt><dd>Louie Myman</dd>
<dt>Met:</dt><dd>Jeremy London, Wendy Carter, Griff Furst, Cleavant Derricks, Daniel Ponsky,
Bashar Rahal</dd>
</dl>
```

If we use : `actor.scrub {single|<dt>Met:</dt>|<dd>|</dd>|</dl>}` the xmltv listing of actor will be  
`<actor>Jeremy London, Wendy Carter, Griff Furst, Cleavant Derricks, Daniel Ponsky, Bashar Rahal</actor>`

That is clearly not what we want. To separate them we use the separator argument. It specifies which string or strings separates the elements. Its syntax is:

`separator="string-1" "string-2" .. "string-n"`

- Between the separator strings a space is allowed but not required.
- All characters are allowed with the exception of | (vertical line). This is no limitation of this function because the program will automatically replace all | characters in the html page into the character combination `!?!!`, this to avoid problems with the special function of this character.
- The characters " ' { and ) need to be preceded by \ So the string ("O'Neil") must be entered as `separator="\(\\"O\'Neil\""`

The scrub specification for actor then becomes:

`actor.scrub {single(separator=", ")|<dt>Met:</dt>|<dd>|</dd>|</dl>}` and the resulting xmltv listing:

```
<actor>Jeremy London</actor>
<actor>Wendy Carter</actor>
<actor>Griff Furst</actor>
<actor>Cleavant Derricks</actor>
<actor>Daniel Ponsky</actor>
<actor>Bashar Rahal</actor>
```

Suppose the html line with the actors looked like this:

```
<dt>Met:</dt><dd>Jeremy London, Wendy Carter, Griff Furst, Cleavant Derricks, Daniel Ponsky
and Bashar Rahal</dd>
```

(The last two actors separated by the word - and - ) We then can use `separator=", " " and "` for the same result.

#### 4.2.5.3 Argument *max* :

To limit the number of elements (either added together in the case of *single value* xmltv elements or listed separately in the case of *multiple value* xmltv elements) we can use the argument max. Its syntax:

`max=n` in which n=positive integer

`actor.scrub {single(separator=", " max=3)|<dt>Met:</dt>|<dd>|</dd>|</dl>}` will result in:

```
<actor>Jeremy London</actor>
```

```
<actor>Wendy Carter</actor>
```

```
<actor>Griff Furst</actor>
```

#### 4.2.5.4 Arguments *include* and *exclude* :

These allow further control over which of the scrubbed elements will be passed to the final result. It is important to realise that both *include* and *exclude* can be used together in one scrub specification. The program will execute these in the order in which they occur in this specification. See for an example of the effect of this in 5.1

Its syntax:

```
include=n -or- first -or- firstn -or- last -or- lastn -or- "string"
```

```
exclude=n -or- first -or- firstn -or- last -or- lastn -or- "string"
```

- n the element number to include or exclude, starting with 1
- first or firstn (like first2) , the first or the first n elements to include or exclude
- last or lastn (like last2) , the last or the last n elements to include or exclude
- "string" , like "met o.m.", include or exclude only elements containing the "string"
- All characters are allowed with the exception of | (vertical line). This is no limitation of this function because the program will automatically replace all | characters in the html page into the character combination !??! , this to avoid problems with the special function of this character.
- The characters " ' { and ) need to be preceded by \ So the string ("O'Neil") must be entered as "\(\\"O'Neil\)"

. As with the argument *separator* (see 4.2.5.1) a list of strings is allowed like:

```
include="string-1" "string-2" .. "string-n"
```

The effect of these arguments differs depending on whether it is entered in - combination and after the argument *separator* – (case A) or not (case B).

#### Case A (after the argument *separator*) :

In this case it allows to make a selection of the elements we want after they are separated.

As example we use the following html for a title and sub-title combination that occurs frequently:

```
<div class="intro-datasheet">
  <div class="img">
    
    <p>Motociclismo: Cto. del Mundo</p>
  </div>
```

Here, the title *Motociclismo*, is separated from the sub-title *Cto. del Mundo* with a : character.

So we can use the arguments *separator=": "* to separate them , we then use *include=first* for the title and *exclude=first* for the sub-title, like this:

```
title.scrub {single(separator=": " include=first)|<div class="intro-datasheet">|<p>|</p>|</div>}
subtitle.scrub {single(separator=": " exclude=first)|<div class="intro-datasheet">|<p>|</p>|</div>}
```

The xmltv result :

```
<title lang="es">Motociclismo</title>
<sub-title lang="es">Cto. del Mundo</sub-title>
```

#### Case B (not after the argument *separator*) :

The program will evaluate all the scrubbed elements (single or multi) on the conditions speci-

fied by the include and/or exclude values.

As example we use the description again:

```
<div id = "program-desc-text">
  <p>Amerikaanse actiefilm.</p>
  <p>Een team van archeologen ontwaakt een mythische slang die vernieling zaait. De enige manier om het wezen te stoppen is door een magische scepter te vinden.</p>
  <p>Geproduceerd in 1998</p>
</div>
```

Remember the original scrub specification:

```
description.scrub {multi|<div id = "program-desc-text">|<p>|</p>|</div>} resulted in :
```

```
<desc lang="xx">Amerikaanse actiefilm. Een team van archeologen ontwaakt een mythische slang die vernieling zaait. De enige manier om het wezen te stoppen is door een magische scepter te vinden. Geproduceerd in 1998</desc>
```

But, the last element - Geproduceerd in 1998 - actually belongs to another xmltv element - date - which is meant to contain the date of production. So in fact it shouldn't be part of the description. We can use the following to exclude it from the description:

```
description.scrub {multi(exclude="Geproduceerd")|<div id = "program-desc-text">|<p>|</p>|</div>}
```

or if we are sure that it is always the last element that contains the production date:

```
description.scrub {multi(exclude=last)|<div id = "program-desc-text">|<p>|</p>|</div>}
```

or if it is always the third:

```
description.scrub {multi(exclude=3)|<div id = "program-desc-text">|<p>|</p>|</div>}
```

or

```
description.scrub {multi(include=first2)|<div id = "program-desc-text">|<p>|</p>|</div>}
```

Even this works!

```
date.scrub {multi(include="Geproduceerd")|<div id = "program-desc-text">|<p>|</p>|</div>} or
date.scrub {single|<div id = "program-desc-text">|Geproduceerd|</p>|</div>} both will result in:
<date>1998</date>
```

(this works because WebGrab+Plus finds any year value inside an element for the date xmltv element, see 4.5.2)

#### 4.2.5.5 Argument *debug* :

Adding the word `-debug-` as argument will start logging of the scrubbing process for the element in the `WebGrab++.log.txt` file. The html page from which the scrubbing is attempted is written to a separate file `html.source.htm` One should use this argument (preferably) for one element and one show at the time, otherwise the results could be confusing. The config file allows to Grab only one show with the `-timespan-` setting.

#### 4.2.5.6 Dedicated Arguments:

The following arguments are dedicated to the use with a certain element

- *lang* :

This argument only works for the element `titleoriginal`. See 4.5.2, `titleoriginal`.

- *force* :

This is a special argument to change the effect of scrubbing the element `index_date` (see 4.5.1)

- *sort* and *timespan* :

Arguments to be used together with `index_showsplit` in case of fragmented multiday indexpages (see 4.5.1 )

### 4.3 General Site dependant data:

One or more lines with the following syntax:

```
site {url=x.x|timezone=UTC+nn:00|maxdays=n.p|cultureinfo=xx-XX|
charset=xxx,yyy|titlematchfactor=nn}
```

and the following are optional:

```
site {ratingsystem=xxx|episodesystem=xxx|grabengine=wget|
firstshow=n|firstday=nnnnnn|subtitlestyle=xxx|retry=xxx}
```

- Site dependant data can be entered on one or more lines starting with the word 'site'

- *url*, e.g. url=tvguids.nl , the url of the site e.g. url=tvguids.nl

- *timezone*, e.g. UTC+01:00 ,the timezone for which the TV guide data is given.

- *maxdays*, specifies the number of days n for which TV guide data is provided by the site, followed by how many index pages p are used for it. If n and p are equal, e.g. 7 days on 7 pages, you can either specify 7.7 or just 7. However if the site has a multiday e.g. a weekly indexpage 7.1 must be specified. (See also 4.5.1, index\_showsplit)

- *cultureinfo*, e.g. cultureinfo=nl-NL , gives data about standards for time and language formats used by the site. For more info :

[http://msdn.microsoft.com/en-us/library/system.globalization.cultureinfo\(v=VS.95\).aspx](http://msdn.microsoft.com/en-us/library/system.globalization.cultureinfo(v=VS.95).aspx)

It is allowed to only specify the language part of it, like charset=en , but the results might be different, especially in country specific items like time formats.

- *charset*, e.g. charset=ISO-8859-1 or UTF-8. Charset is normally found somewhere at the beginning of the html source. Sets proper decoding of the html pages. This charset is applied to all grabbed html pages (index and show-detail). Sometimes the charset for these pages is different. In that case specify both, separated by a comma. The first will be used for the index page and the second for the show-detail page.

- *titlematchfactor*, e.g. titlematchfactor=50 , this is a number from 0 to 100 that specifies how strict the title comparison is done by WebGrab+Plus (as discussed in 2.1). Some sites use different show titles for the index pages and show detail page. Start with a high value e.g. 90 and adjust to lower if too many unnecessary show updates occur. (see also 4.5.1 element index\_title)

- *ratingsystem* (optional) Specifies the system attribute of the xmltv element rating. Some countries have a uniform system to classify shows (e.g. the MPAA in the US and KIJKWIJZER in the Netherlands). If the site's country has no such system it is best to use a two letter country spec like ES for Spain.

- *episodesystem* (optional), specifies the xml attribute *system* of the *episode-num* xmltv element. See xmltv specification for details. The most common values are *xmltv\_ns* and *onscreen*.

- *grabengine* (optional), specifies which of the two available grabengines (the part of the program that connects to the site and grabs the html pages) will be used for this site. Any other value than 'wget' will use the standard internal .net based grabengine. The other one is the external 'WGet.exe', some sites might behave better with it.

- *firstshow* (optional). Specifies which is the first show on the indexpage that will be processed (scrubbed). When not specified, or if firstshow=0, it starts with the first show found on the indexpage. This value is important for sites that lists shows on the indexpage from the previous day 'yesterday', because the program assumes that the first show is of 'today'. A mix-up of the date value will be the result. The firstshow value allows to skip these 'yesterdays' shows. Instead of a number, the string *now* can be used. This will skip all shows until a day change (passing midnight) is detected.

- *firstday* (optional), e.g. firstday=0123456 This is to be used if the site has a multiday index page (an overview of shows for several days). When in such a case, this indexpage doesn't change for several days (remains starting on the same day), the program needs info where to start. The firstday value tells the program how many days to skip to find the shows of 'today'. It needs to be entered as 7 numbers, from the first : days to skip on Monday .. To the last : days to skip on Sunday.

Suppose a multiday indexpage which lists the shows for a week starting Sunday. Then if we grab on Sunday there is no need to skip a day, but on Monday we must skip 1 day (the Sunday), on

Tuesday we must skip 2 days ... Etc. We specify firstday=1234560

- *subtitlestype* (optional). Specifies the xmltv attribute *type* of the element *subtitles*. Possible standard values are *teletext*, *onscreen* and *deaf-signed*.
- *retry* (optional). This is the same retry setting as the general retry setting in the config file (see 3 and APPENDIX B). If a site is markedly slower than others used in the same run, it is possible to set different retry, timeout and delay values for that site here. The syntax is the same as in the config file. E.g. `retry=<retry>12</retry>` or `retry=<retry time-out="10" channel-delay="5" index-delay="1" show-delay="1">4</retry>`
- *keptabs* (optional) This will disable the default replacement of tab \t characters in spaces in html pages. In some cases tabs can be useful in scrubstrings.
- *keepindexpage* (optional) Saves the index-page for use with other channels of the same site. Useful when a site list all or a group of channels on one index-page. It saves grabbing the same index-page again and again.
- *loadcookie* (optional) If a site requires a login with username and stores your personal settings in a cookie, it is necessary to load this cookie into WG++ for it to send to this site as part of the WebRequest. Specified as `loadcookie=cookie-file-name`. The cookie-file-name is the name of the cookie file which must be present in the WebGrab home folder. (see loadcookie.txt in the documentation folder for how to create such a file). The program filters the cookies in this cookie-file for the cookies relevant for the site, using the *url* (see above) as domain. Optionally the *cookie-file-name* can be followed by additional domain strings that specify which of the cookies for other domains will be kept. Example : `site{loadcookie=yourtv.com.au.cookie.txt}` or `site{loadcookie=yourtv.com.au.cookie.txt,yahoo.com}`

Example:

```
site {url=tvuids.nl|timezone=UTC+01:00|maxdays=6|cultureinfo=nl-NL|charset=ISO-8859-1,UTF-8|titlematchfactor=90|firstshow=5}
site {ratingsystem=KIJKWIJZER|episodesystem=xmltv-ns|grabengine=standard|retry=<retry time-out="15">10</retry>|keptabs}
```

## 4.4 Data that WebGrab+Plus needs to compose the url's to download pages

### 4.4.1 url index

This is the url WebGrab+Plus uses to download the index pages (see 2). Every site uses its own way to compose these url's, but as a rule it always contains references to the channel and to the timespan for which it is valid. WebGrab+Plus includes an `url_index` builder that composes this url based on an entry in the SiteIni file with the following syntax:

```
url_index{url|stringfragment-1|stringfragment-2| ... |stringfragment-n}
```

url: just an indication of the type of data that follows, (argument debug supported)

- stringfragment: a fragment of the urlstring for the position n. It can be either a fixed string fragment (independent from channel, date or subpage) like `http://www.tvuids` or one of the 3 types of variable string fragments: *channel* or *urldate* or *subpage*
- channel: The reference to the channel for which the url is meant. WebGrab+Plus uses the value of the `-site_id-` attribute of the channel table in the WebGrab++.config.xml file. Most sites use a simple channel number as `-site_id-` but some use rather complicated constructions. (e.g. TvGids.nl uses a number `1` for Nederland1, while Skynet.be uses `nederland-1?channelid=216` for the same). For most sites a channel list file is provided in a siteini.pack.
- urldate: The reference for the timespan or start date. Most websites have one index page per day. WebGrab+Plus supports this per day timespan style.

The program also supports multiday e.g. weekly index pages (see also 4.3 , maxdays). In that case the urldate can specify a start day.

Some other sites however have (occasional) index subpages e.g. when the number of shows of that day exceeds the space of the displayed webpage. In that case the `-subpage-` reference has to be specified:

- subpage: Specifies eventual subpages part of the Url. See 4.4.1.2

#### 4.4.1.1 urldate format:

The day string that appears at the position of `-urldate-` depends on the value of a separate SiteIni specification `urldate`, its syntax:

```
urldate.format{daycounter|todaynumber} or
urldate.format{weekdaynumber|Sunday-number} or
urldate.format{weekdayname|Monday-name|Tuesday-name|...|Sunday-name} or
urldate.format{datestring|string|optional cultureinfo} or
urldate.format{datenumber|standard|offset} or
urldate.format{list|day1string|day2string|..|daynstring|{other
    urldate format for following days}}
```

- Datestrings follow the .Net standard for datestrings as found in :

<<http://msdn.microsoft.com/en-us/library/az4se3k1.aspx> and <http://msdn.microsoft.com/en-us/library/8kb3ddd4.aspx>>

- When the `cultureinfo` used for the `datestring` is different from the one given in the site specification it can be added as option. Assume `cultureinfo=nl-NL` for the following examples.  
- the `list` method is to be used when the Site uses a value like `-today-` for today rather than a date value. It fills the url with these day strings, eventually followed by whatever `urldate` format specified for the remaining days.

- the `datenumber` method returns a number that represents a date-time value. It supports the following standards : VBA , the `daynumber` as used in MS Office ; UNIX , the number of seconds from 1970/1/1 00:00 UTC ; JAVA , the number of milliseconds from 1970/1/1 00:00 UTC ; TICKS , the number of 100 nanoseconds units from 00/00/00 00:00UTC

- the `weekdayname` method is to be used when the `daystring` required is a non standard weekday name that cannot be generated by the `datestring` method.

Some examples to illustrate:

```
urldate.format {daycounter|0} * output like: 0 1 2 ....
```

```
urldate.format {weekdaynumber|0} * suppose today is Tuesday, output like: 2 3 4 ..
```

```
urldate.format {weekdayname|lu|ma|mi|ju|vi|sa|do} * suppose today is Wednesday, output like:
mi ju vi ..
```

```
urldate.format {datestring|yyyy/MM/dd} * output like: 2010/05/10 2010/05/11 ...
```

```
urldate.format {datestring|ddd} * output like: dinsdag woensdag ...
```

```
urldate.format {datestring|d} * output like: 10-5-2010 11-5-2010 ...
```

```
urldate.format {datestring|d|en-GB} * output like: 10/5/2010 11/5/2010 ... (other culture,
other standard)
```

```
urldate.format {datestring|ddd/dd/MMM/yyyy} * output like: ma/10/mei/2010 di/11/mei/2010 ...
```

```
urldate.format {datestring|ddd/dd/MMM/yyyy|en-GB} * output like: Mon/10/May/2010 Tue/11/
May/2010 ... (other culture, other standard)
```

```
urldate.format{datestring|dddd-dd-MM-yyyy} * output like: maandag-10-05-2010 dinsdag-11-05-
2010 ...
```

```
urldate.format{list|vandaag|morgen|{datestring|ddd|nl-NL}} * output like (if today is Mon-
day): vandaag morgen woensdag donderdag ...
```

```
urldate.format{list|Today|{datestring|d|en-GB}} * suppose today is 9/5/2010 , output like:
Today 10/5/2010 11/5/2010 ...
```

#### 4.4.1.2 subpage format:

```
subpage.format{number|leadstring|first page number|stopstring} or
subpage.format{letter|leadstring|first page letter|stopstring}
```

- `leadstring`: fixed part of the subpage string.

- `stopstring`: The unique string that occurs on the subpage after the last one valid. When a subpage is specified in the `index_url` specification, the program will automatically step from

one page to the next until the stopstring is detected. After that the same subpage stepping will start for the next day. If the stopstring is not detected the stepping will stop after 8 subpage tries with a subpage warning and try the next day.

Some examples of to illustrate:

```
subpage.format {number|1|<p>page not found</p>} * output for subsequent pages: 1 2 3 ..
subpage.format {number|section_|1|page not found} *output: section_1 section_2 section_3 ..
subpage.format {letter|p|a|page not found} * output: pa pb pc ...
```

Full examples of the url\_index specification:

Suppose WebGrab++.config.xml channel entry:

```
<channel update="i" site="tvguids.nl" site_id="1" xmltv_id="NED1-tvguids">NED1</channel>
```

And the url\_index and urldate.format entries in the siteini:

```
url_index{url|http://www.tvguids.nl/zoeken/?q=&d=|urldate|&z=|channel|&t=0&g=&v=0}
urldate.format {daycounter|0} for 3 days will result in:
```

```
<http://www.tvguids.nl/zoeken/?q=&d=0&z=1&t=0&g=&v=0>
<http://www.tvguids.nl/zoeken/?q=&d=1&z=1&t=0&g=&v=0>
<http://www.tvguids.nl/zoeken/?q=&d=2&z=1&t=0&g=&v=0>
```

Another example:

```
<channel update="i" site="skynet.be" site_id="nederland-1?channelid=216" xmltv_id="NED1-
skynet">NED1</channel>
```

```
url_index{url|http://www.skynet.be/entertainment-nl/tv/kanalen_|channel|&new_lang=nl&date=|urldate}
urldate.format {datestring|yyyy-MM-dd|nl-BE} for 3 days will result in :
```

```
<http://www.skynet.be/entertainment-nl/tv/kanalen_nederland-1?channelid=216&new_lang=nl&date=2010-06-22>
<http://www.skynet.be/entertainment-nl/tv/kanalen_nederland-1?channelid=216&new_lang=nl&date=2010-06-23>
<http://www.skynet.be/entertainment-nl/tv/kanalen_nederland-1?channelid=216&new_lang=nl&date=2010-06-24>
```

And one using the list method:

```
<channel update="i" site="tvguids.upc.nl" site_id="Nederland+1" xmltv_id="NED1-upc">NED1</channel>
```

```
url_index {url|http://tvguids.upc.nl/TV/Guide/Channel/|channel|/|urldate}
urldate.format {list|Today|Tomorrow|{datestring|dddd|en-GB}} today being Tuesday, for 3 days
will result in :
```

```
<http://tvguids.upc.nl/TV/Guide/Channel/Nederland+1/Today>
<http://tvguids.upc.nl/TV/Guide/Channel/Nederland+1/Tomorrow>
<http://tvguids.upc.nl/TV/Guide/Channel/Nederland+1/Thursday>
```

A subpage example:

```
<channel update="" site="plus.es" site_id="PLAYDC" xmltv_id="PlayDisney">PlayDisney</channel>
```

```
url_index{url()|http://www.plus.es/guiatv/resultados.html?tipo=dh5&frm=B&dia=|urldate|&c%5B%
5D=|channel|&f=TO&pr=L|subpage}
urldate.format {datestring|yyyy-MM-dd}
subpage.format {number|&pag=|1|No hay ningún título que cumpla con las condiciones de la búsqueda}
```

Supposing this channel has 2 subpages this will result in :

```
<http://www.plus.es/guiatv/resultados.html?tipo=dh5&frm=B&dia=2010-09-30&c%5B%
5D=PLAYDC&f=TO&pr=L&pag=1>
<http://www.plus.es/guiatv/resultados.html?tipo=dh5&frm=B&dia=2010-09-30&c%5B%
5D=PLAYDC&f=TO&pr=L&pag=2>
<http://www.plus.es/guiatv/resultados.html?tipo=dh5&frm=B&dia=2010-09-30&c%5B%
5D=PLAYDC&f=TO&pr=L&pag=3> (detects the stopstring -> step to next day)
<http://www.plus.es/guiatv/resultados.html?tipo=dh5&frm=B&dia=2010-10-01&c%5B%
5D=PLAYDC&f=TO&pr=L&pag=1>
<http://www.plus.es/guiatv/resultados.html?tipo=dh5&frm=B&dia=2010-10-01&c%5B%
```

5D=PLAYDC&f=TO&pr=L&pag=2>

<http://www.plus.es/guiatv/resultados.html?tipo=dh5&frm=B&dia=2010-10-01&c%5B%

5D=PLAYDC&f=TO&pr=L&pag=3> (detects the stopstring → stop, last page of last day)

#### 4.4.2 other url elements

The next important url is the one that points to the show detail page. Normally there is some kind of hyperlink with a `<a href=` tag that points to it but not always the complete url is found there. The SiteIni specification for url's other than the `url_index` (above) allows to add the missing components if necessary:

```
urlelementname {url(optional arguments)|leadstring|bs|optional es|
                  ee|optional be}
```

- `urlelementname`: Can be either `index_urlshow` (obligatory, points to the show detail page) and `index_urlchannellogo` (optional)
- `url` : just an indication of the type of data that follows, (argument debug supported)
- `leadstring`: The invariable part of the url that sometimes misses from the html link
- the remaining parts `|bs|optional es|ee|optional be` are the normal scrubstrings

Example for show Max Geheugentrainer on site tvgids.upc.nl

```
index_urlshow {url|http://tvgids.upc.nl|<a href="||">} results in
```

```
<http://tvgids.upc.nl/TV/Guide/Programme/9184772/MAX_Geheugentrainer/Nederland+1/>
```

And an example without a leadstring of the same show on site skynet.be:

```
index_urlshow {url||<a href="||">} results in
```

```
<http://www.skynet.be/entertainment-nl/tv/tv-gids/detail_max-geheugentrainer?
programkey=MagnetMedia__11435188>
```

(notice that the 'empty' leadstring `||` is required!!)

#### 4.5 Data needed to scrub xmltv elements from the downloaded pages

There are two section:

- the elements scrubbed from the index page
- the elements scrubbed from the show detail page

All scrub specifications of these sections follow the syntax as explained in section 4.2 with the action specifier `scrub` :

```
elementname.scrub{type(optional arguments)|bs|
                  optional es|ee|optional be}
```

##### 4.5.1 The elements scrubbed from the index page

The element names of these start with the prefix `index_`

See for more details the element name table in APPENDIX C

The elements of this section:

- `index_showsplit` obligatory `-yes-` , type `-multi-` only

Actually this is not a regular xmltv element but is required for the proper operation of the xmltv update process.

As explained in section 2 WebGrab+Plus uses the show data on the index page to determine if an update of the xmltv file is necessary for that particular show. To do that it first splits the index page into parts, one for each show. For that it needs the `-index_showsplit-` scrub specification that returns all those show parts as result.

Normally, the `indexpages` list the shows of one day in ascending start-time order. Some sites however list shows of several days on one page (multiday `indexpage`, see also 4.3, `maxdays`). As a complication it occurs that the shows on these multiday `indexpages` are not listed in pure ascending start-time order but in day section fragments (like morning, afternoon, evening etc.), e.g. first all morning shows of all days followed by all afternoon shows of all days

etc.

The program provides two options to sort the shows on this type of multiday index pages:

\*\* First option: With the special attribute *sort@*. This will sort the shows in ascending time order. It can be used if the division into the day section fragments occurs at fixed times of the day (e.g. the evening section always start at the first show after 20:00). In that case enter :

```
index_showsplit.scrub {type(sort@time-1, time-2, ..,time-n)|. . . .}
```

It can also be used if the division always occurs at – or immediately after – the next full- or half-hour following the last show of the previous day section. In that case enter for full-hour :

```
index_showsplit.scrub {type(sort)|. . . .} or  
index_showsplit.scrub {type(sort@fullhour)|. . . .}
```

and for half-hour:

```
index_showsplit.scrub {type(sort@halfhour)|. . . .}
```

\*\* Second option: By scrubbing the each day section fragment on the indexpages separately. Like , if the index page is split in a day-section and an evening-section :

```
index_showsplit {type(optional timespan=hours)|day-section-scrubstrings}  
index_showsplit {type(optional timespan=hours)|evening-section-scrubstrings}
```

Each scrub will result in an array of shows and the resulting arrays will be merged into one in ascending start time order. The program will attempt to determine the time structures of the arrays automatically. In some case it can help to specify a *timespan* attribute, which is the approximate duration of each day section. E.g. if the day-section is from 05:00 to 18:00, specify *timespan=13:00* and if the evening/night-section is from 18:00-05:00, specify *timespan=11:00*

Adding the attribute *debug* during the development phase will show the result of the sorting in the log file.

- *index\_date* obligatory *-no-*

Scrubs the date from the first index page which is the date of the first day of the timespan for which the shows will update. If no *index\_date* scrubspec is entered in the SiteIni file the date of – today- is taken. By default the scrubbed date value is only used as a check if the first grabbed index page is from ‘today’ . If not, the scrubbing of the shows will be stopped with an error message. When the dedicated argument *force* is added to the scrubstring, the scrubbed date value will be used as the date of the first show on the index page. The date of the following days is calculated by the program.

- *index\_start* obligatory *-yes-* , xmltv element *-start-*

Scrubs the start time of the show, essential for both the xmltv output file as for the update decision process of the program

- *index\_stop* obligatory *-no-* , xmltv element *-stop-*

Scrubs the stop time of the show. Because not all Sites provide this information (relying on the start time of the next show for it), WebGrab+Plus does the same if no scrubspec is entered in the SiteIni file. The resulting value either from direct scrub or from substitution is essential for the xmltv update decision process.

- *index\_duration* obligatory *-no-* , xmltv element *-stop-*

Alternative for *index\_stop*. Some Sites specify this rather than the stop time. WebGrab+Plus calculates the stop time from *stop=start + duration*. For that it must be in the format hh:mm. (The operations discussed in 4.6 provide ways to convert it to this format if it isn't)

- *index\_title* obligatory *-yes-* , compared with xmltv element *-title-*

The show title is found on two places in most (all?) Sites, on the index page and on the show

detail page. The latter is the most accurate and is used by WebGrab+Plus for the `xmltv` element `-title-`. The `index_title` is used (and essential) for the `xmltv` update decision process.

Because some sites have 'varying' differences between both show titles and because WebGrab+Plus compares the `index-title` with the title in the existing `xmltv` listing (which originates from the show detail page) a one to one comparison is not possible due to these differences. WebGrab+Plus uses a smart-comparison which results in a `titlematchfactor` as detailed in 2.1.

If a site lists a combination of elements together with the `index_title` (like `es category, title, subtitle ee` or other combinations including the title) it is best to scrub them with a separator argument without include or exclude arguments! This yields all these elements in the `index_title`. The title comparison is smart enough to find the real title within this combination. See also 4.3

- Besides the `index_elements` discussed above, the program allows most other elements to be scrubbed from either/and the index pages and the show-detail page. If the program finds a `scrubspec` of a certain element for both the `index-` and the detail page, and both render a result, it will add them together in a way that depends on the if it concerns a *multiple value* `xmltv` element or not (as discussed in 1 and 4.2.3)

#### 4.5.2 The elements scrubbed from the show-detail page

This section contains the bulk of the `xmltv` elements.

See for more details the element name table in 4.2.2 and APPENDIX D for an overview.

Some of the elements need a little more explanation:

- `title` obligatory *-yes-*, `xmltv` element `-title-`  
(obligatory *-no-* if index only update mode, see 2.2)  
See also 4.5.1 `index_title`, 2.1 the update process and 4.3 `titlematchfactor`

- `titleoriginal` obligatory *-no-*, `xmltv` element `-title-`  
See also 4.2.5.5 Dedicated arguments.

It is meant to allow multiple titles for different languages. The scrubstring can include the dedicated argument `lang`. The syntax of it is :

`lang` or `lang=xx` (xx=two letter language spec like `en` for English)

If a `lang` argument is given without a language value or with the value "xx" or if no `lang` argument is added, the title `lang` attribute in the `xmltv` will be `lang="xx"`, which is supposed to indicate the 'original' show title in an unspecified language. If a two letter language spec is provided it will use that in the `xmltv lang=` attribute.

Example:

```
title.scrub {single |scrubstring-1}
titleoriginal.scrub {single (lang=en)|scrubstring-2}
could result in something like:
```

```
<title lang="es">Mujeres Desperadas</title>
<title lang="en">Desperate Housewives</title>
```

- `productiondate` obligatory *-no-*, `xmltv` element `-date-`

The `productiondate` should yield the year of the production of the show. Because it is often hidden inside another element, like the description it is rather difficult to find unique element separators. WebGrab+Plus will scrub the first 'year' value (yyyy) between the element separators automatically. See also 4.2.5.3

- `subtitles` obligatory *-no-*, `xmltv` element `-subtitles-`  
see also 4.3 - `subtitlestype` -

- `premiere` obligatory *-no-*, `xmltv` element `-premiere-`

- `previousshown` obligatory *-no-*, `xmltv` element `-previously-shown-`

The three elements above have no value in the `xmltv` output file, they are there or not. E.g. if the show is subtitled the `xmltv` file shows `<subtitles/>` The program expects the scrubbed

value to have the string value "true" for it to write it to the xmltv file. Of course it is unlikely that the scrubbing will yield this value directly. This means that in most cases a modify operation (see 4.6) is required to modify the scrubbed value in "true" E.g. like:  
`subtitles.modify {replace(not "")|'subtitles'|true}`

### 4.5.3 Special Elements

#### - temp elements :

`index_temp_1` , `index_temp_2`, `index_temp_3`, `index_temp_4` and `temp_1`, `temp_2`, `temp_3`, `temp_4`

These special elements have no direct xmltv destination. They can be used to temporary scrub and store data that is later used together with the action specifier *modify* (see 4.2.3 and 4.6) to alter or create other elements.

#### - index\_variable\_element

This special element has no direct xmltv destination. It can be used if any of the other scrubstrings requires a value that varies (e.g. with each channel). Its value can be scrubbed and modified. It is the only element that :

a. can be used in a scrub string as part of `bs es ee` or `be` like in :

`index_variable_element.scrub {single|Billing\t\n||\t|\t} *`  scrubs the value from the index page and uses it to split the index pages in shows :

`index_showsplit.scrub {multi|'index_variable_element'||\n}`

b. they can also be used in *argument* values like:

`index_showsplit.scrub {multi(includeblock='index_variable_element'|"column">|time">|)}`

c. allows to pass certain channel values from the config file with a modify command (see 4.6 for details about the modify command) like in :

`index_variable_element.modify {addstart|'config_site_id'}`

This line copies the `site_id` for the actual channel from the channel list in the config file.

Other supported channel values are :

the `xmltv_id` entered as `'config_xmltv_id'`,

the `display_name` entered as `'config_display_name'`

and the `site_channel` entered as `'config_site_channel'`.

- *previous value* elements: In these elements the value of the previous scrub is stored. The program automatically stores the values of previous scrub of the following elements: `index_start`, `index_stop`, `index_duration`, `index_temp_1` to `-4` and `temp_1` to `-4`. These values can be recalled by adding a prefix `previous_` to the above element names.

A typical use is when a site displays indexpages graphically, each next show in a horizontal grid, the start and stop times hidden in pixel coordinates, then it is necessary to know the previous value of time elements to calculate the actual start and stop time. (See APPENDIX D and 4.6.4.5 Calculate)

### 4.6 Operations: Optional data that allows post modification of the scrubbed xmltv elements.

Action specifier: *modify*

With the data in this section of the SiteIni file it is possible to modify already scrubbed elements and/or obtain a value in by other means than scrub from a html page. The elements for which these modifications are supported are listed in APPENDIX D. The syntax to specify such a modification :

```
elementname.modify {commandname(optional arguments)
                    |optional expression-1|optional expression-2}
```

- element name : Any of the elements listed in APPENDIX D for which the action specifier *modify* is allowed.

- modify : the action specifier for this type of operation

- `commandname`, either :
  - `replace` : replaces the value of `expression-1` with that of `expression-2` (see 4.6.4.1)
  - `remove` : removes the value of `expression-1`, no need for `expression-2` (see 4.6.4.2)
  - `substring` : extracts a part of `expression-1`, no need for `expression-2` (see 4.6.4.3)
  - `addstart` : adds the value of `expression-1` to the start of the element, no need for `expression-2` (see 4.6.4.4)
  - `addend` : adds the value of `expression-1` to the end of the element, no need for `expression-2` (see 4.6.4.4)
  - `calculate` : performs a set of calculations, `expression-1` is an arithmetic expression, no need for `expression-2` (see 4.6.4.5)
  - `cleanup` : tidying up of elements, no expressions (see 4.6.4.6)
- arguments
  - *conditional arguments*: There are two possible sets of conditional arguments
    - *Pre-conditions* that needs to be true for the operation to be performed. They are evaluated first. It allows to evaluate the value of any element or compare element values with constants or other elements. (see 4.6.2.1)
    - *Post-conditions*, simple condition that only evaluate the value of the element to be modified before or after the operation. (see 4.6.2.2)
  - `debug` : Adding the word `-debug-` as argument will start logging of the modify process for the element in the `WebGrab++.Log.txt` file.
  - `format` : Specifies the output format for the `calculate` command. (see 4.6.4) Supported values are all the standard numeric format strings F and D, as described in <http://msdn.microsoft.com/en-us/library/dwhawy9k.aspx> like `format=F0`. Default is F2 (two decimal digit fixed point, like 16.35). Also supported is `format=time` and `format=date`, this will convert the numeric value in HH:mm and yyyy/MM/dd respectively as default date-time format. For other formats, add a comma and date-time format string after the word `time` or `date`, like `format=time,h:mmtt` or `format=date,dd/MMMM/yy` See <http://msdn.microsoft.com/en-us/library/8kb3ddd4.aspx> for date-format strings. It will convert four types of numeric input values:
    - Decimal day time values, like 16.35 will be converted into 16:21.
    - Integer date values in UNIX date format (Seconds counting from 01/01/1970). E.g. 1314866400 will be converted into 8:40 or 2011/09/01
    - Integer date values in JavaScript date format (1ms units counting from 01/01/1970). E.g. 1314866400000 will be converted into 8:40 or 2011/09/01
    - Integer date values in .net ticks format (100ns units counting from 01/01/0001). E.g. 634504668000000000 will be converted into 9:40 or 2011/09/01
 Adding `utc` as prefix, like `format=utctime` or `utcdatetime` will return the utc time or date (ignoring the local timezone)
 Finally there is `format=productiondate`. When entered it will return the first 4 digit numeric value it finds in `string-1` that is between 1900 and 'nextyear'.
 Example: `productiondate.modify {calculate(format=productiondate)|'description'}`
  - `type` : When command `expression-1` is specified by means of place-indices, it specifies the index-base. (see 4.6.1.3 Expression-1 with indices) It is also used with command `calculate - index-of` (see (4.6.4.5.2) Possible values are `type=string` (default, `expression-1` is specified as string, no indices), `type=char` (the indices specify character positions),

- type=word* (the indices specify word positions), *type=sentence* (the indices specify sentence positions) and *type=element* (the indices specify element positions in the case of multi value elements, see 4.6.1.2)
- *separator* : This specifies a separator string value that is used when converting multi value elements to a combined single value. It can be used together with the commands *remove*, *replace*, *addstart* and *addend*. If this argument is entered, the operation will try to convert the result in a single value string, adding the separator string between the multi value components. (see 4.6.1.2 Multiple value elements and modify)
- *style* : This argument can be added to the cleanup command (see 4.6.4.6) to specify the required style of the cleanup result. Possible values are *style=sentence*, *style=name*, *style=upper* (convert to UPPERCASE) and *style=lower* (convert to lowercase)
- *expression-1* and *-2* components,
  - these expressions can be composed with:
    - *text* : all characters with exception of | { and '
      - If any of these are needed in the string they have to be preceded by the backslash character \, like O'Neil must be entered as O\'Neil
    - *element* : to be entered between ' ', like 'title' or 'temp\_1',
      - the value of the element will be inserted in the expression result.
    - *scrubstring* : to be entered between '{ }'
      - like '{single|<a ref=|<p>|</>|<table}'
      - The use of a scrubstring in this way has a small limitation: The scrub is performed from the same html page as from which the element is originated.

So, if it concerns an element from the index page the scrub entered here is also done from the index page. In most cases it is easier and more flexible to use the 'temp' and 'index\_temp' elements instead. See also 4.6.1.1 for limitations.

  - *indices* : Expression-1 only.
    - Can be used to extract (with commands *substring* and *replace*) or remove (with command *remove*) parts of a source string.
    - The use of indices must be accompanied by the argument *type* to specify the index-base (see above; arguments). Indices must be entered as two (integer) numbers separated by a space. Each of these numbers can also be entered as element enclosed by ' '. The first number represents the *start* position, the second number the *length*. When the start position is entered as a negative integer value, the start position is counted from the end of the source string backwards.
  - arithmetic expression in the case of command *calculate* (see 4.6.4).
  - combinations of text, element, scrubstring, indices and arithmetic expression.
    - See 4.6.4 , the commands, for examples.

#### 4.6.1 Notes and examples of the effects of *modify*

##### 4.6.1.1 The order of the actions.

Webgrab+Plus executes the scrubbing and modifying of the elements in a certain order. Roughly like this:

1. Compose the *url\_index* and grab the *index-page(s)*
2. Scrub *index\_date*, *index\_variable\_element* and *index\_channellogo*
3. if needed, Modify *index\_date*, *index\_variable\_element* and *index\_channellogo*
4. Split the *index-page(s)* in *index shows*
5. Step through the *index shows* one by one
6. Scrub all other *index\_ elements* from the *index show*.
7. if needed, Modify all *index\_ elements* from the *index show*
8. Update decision. If no update - - back to 5, next *index show*

9. if: it is an index\_only channel or: if no valid url\_show is scrubbed,
  - back to 5, next index show
  - or else :
10. grab the show-detail page
11. scrub all show detail elements
12. if needed, modify the show detail elements
13. compose the xmltv elements and write them to the xmltv output file
14. if more shows, back to 5, next index show. Else next channel.

- The order in which the scrubbing of the elements is done (in 2. 6. and 11.) is fixed, not important for the results and independent of the order of the (index\_) scrubstrings in the siteini file.

- The order in which the modify operations are done (in 3. 7. and 12.) is determined by the order in which the modify operations (for that group) are listed in the siteini file. E.g. in 7. , the modification of all index\_elements other than index\_date, index\_variable\_element and index\_channellogo is done. It will modify these in the order they occur in the siteini file.

It is important to realise that this order will *influence the result* and also *poses restrictions* on the use of other elements in the modify operation. For the influence on the *result* consider the following :

Suppose : description = A short story

Case 1.

```
temp_1.modify {calculate(format=F0)|'description' " " #} —> result temp_1 = 2
description.modify {remove|short } —> result description = A story
```

Case 2.

```
description.modify {remove|short } —> result description = A story
temp_1.modify {calculate(format=F0)|'description' " " #} —> result temp_1 = 1
```

These simple cases illustrate that modify operations work on bases of the results of previous modify operations.

This also explains the *restrictions* : Operations that try to use elements that have no value (as yet) will not work. Like trying to use (non index\_) elements in the 7. That can't work because these element are not yet scrubbed, that occurs later in 10.

Multi Value Elements examples:					
element-to-modify.modify{command(argument) expression-1 expression-2}					
element-to-modify	element	command	argument	expression-1   expression-2	result
Abc def.	Ghi Jkl Mno	addstart		'element'.	Ghi Jkl Mno. Abc def.
Abc def.	Ghi Jkl Mno	addstart		'element'\	Ghi Jkl Mno Abc def.
Abc def.	Ghi Jkl Mno	addstart	separator=" & "	'element'.	Ghi & Jkl & Mno. Abc def.
Abc def.	Ghi Jkl Mno	addend	separator=", "	'element'.	Abc def. Ghi, Jkl, Mno.
Abc Def	Ghi Jkl	addend	separator=", "	* 'element'.	Abc*Ghi, Jkl.def*Ghi, Jkl.
Ghi, Jkl, def, Mno.	Ghi Jkl Mno	remove	separator=", "	'element'.	def,
- empty -	Ghi Jkl Mno	addstart		'element'	Ghi Jkl Mno
Abc Def Ghi	Ghi Jkl Mno	remove		'element'	Abc Def
Abc Def Ghi	Ghi Jkl Mno	replace		'element' Xyz	Abc Def Xyz
Abc Def Ghi	Ghi Jkl Mno	replace	separator=" & "	'element' Xyz	Abc & Def & Xyz

Consider the following:

A site has only show-detail links for a limited number of shows. We scrub both index\_description and description to get a description in both cases. That could create a double description in case of a show with a show-detail link. So, we would like to erase the index\_description if the description is not empty. It is only logical we try this:

```
index_description.modify {remove('description' not "")|'index_description'}
```

That, unfortunately, will not work because this operation is done in 7. and uses an element (description) that is only available after 10. A way to solve this particular case is :

```
index_description.modify {remove('index_urlshow' not "")|'index_description'} ( It tests for the show-detail link -index_urlshow- to exist which value is available in 7.)
```

#### 4.6.1.2 Multiple value elements and modify

Some explanation about the internal handling of elements: Most elements can have more than one value, either through separators, through multiple scrubs or by being a multi type scrub. Internally they are not stored as array but as a string with the | character as separator. Thus, an element with the values AAA BBB ccc ddd will have the internal representation AAA|BBB|ccc|ddd It depends on another element property , multiple xmltv value, true or false, how these values will be written to the xmltv file. If true (multiple), they will get multiple

Indices examples:						
	element.modify {command(type=xx optionalseparator=xx) expression-1 optional expression-2}					
	expression-1:  indices	or	'element' indices			
--->	with command <u>substring</u> and <u>remove</u> :					
	element value	separator	type	indices	command <u>substring</u>	command <u>remove</u>
single value	Abc def ghi. Jkl mno pqr.		char	2 4 or -23 4	c de	Abf ghi. Jkl mno pqr.
	Abc def ghi. Jkl mno pqr.		word	2 3 or -4 3	ghi. Jkl mno	Abc def pqr.
	Abc def ghi. Jkl mno pqr.		sentence	0 1 or -2 1	Abc def ghi.	Jkl mno pqr.
	Abc def ghi. Jkl mno.		element	0 1 or -1 1	Abc def ghi. Jkl mno.	- empty -
multi value	Abc def ghi  Jkl mno pqr  Stu vwx yz		char	2 4	c de   l mn   u vw	Abf ghi  Jko pqr  Stx yz
	Abc def ghi  Jkl mno pqr  Stu vwx yz		char	-9 4	c de   l mn   tu v	Abf ghi  Jk opqr  Swx yz
	Abc def ghi  Jkl mno pqr  Stu vwx yz		word	1 1 or -2 1	def   mno   vwx	Abc ghi  Jkl pqr  Stu yz
	Abc def. Ghi jkl.  Mno pqr. Stu vwx yz.		sentence	0 1 or -2 1	Abc def.  Mno pqr.	Ghi jkl.  Stu vwx yz.
w. separator	Abc def ghi  Jkl mno pqr  Stu vwx yz		element	1 1 or -2 1	Jkl mno pqr	Abc def ghi  Stu vwx yz
	Abc def ghi jkl	","	element	1 1	def, ghi	Abc, jkl
	no length		char	2 or -23	c def ghi. Jkl mno pqr.	Ab
	out of range		char	2 50	c def ghi. Jkl mno pqr.	Ab
out of range		char	50 2	- empty -	Abc def ghi. Jkl mno pqr.	
out of range		char	-27 5	Abc	def ghi. Jkl mno pqr.	
--->	with command <u>replace</u> :					
	element value		type	indices	expression-2	command <u>replace</u>
single value	Abc def ghi. Jkl mno pqr.		char	2 4 or -23 4	x	Abxxxxf ghi. Jkl mno pqr.
	Abc def ghi. Jkl mno pqr.		word	2 3 or -4 3	xyz	Abc def xyz xyz xyz pqr.
	Abc def ghi. Jkl mno pqr.		sentence	0 1 or -2 1	Uvw xyz.	Uvw xyz. Jkl mno pqr.
	Abc def ghi. Jkl mno pqr.		element	0 1 or -1 1	xyz	xyz
multi value	Abc def ghi  Jkl mno pqr		char	2 4 or -9 4	x	Abxxxxf ghi  Jkxxxxo pqr
	Abc def ghi  Jkl mno pqr		word	1 1 or -1 1	xyz	Abc xyz ghi  Jkl xyz pqr
	Abc def. Ghi jkl.  Mno pqr.		sentence	0 1 or -1 1	Uvw xyz.	Uvw xyz. Ghi jkl.  Uvw xyz.
	Abc def  Jkl mno pqr  Stu vwx		element	1 1 or -2 1	Xyz	Abc def  Xyz  Stu vwx

xmltv elements like:

```
<element>AAA</element>
<element>BBB</element> etc.
```

If false, they will be added together, separated by a period(space), like:

```
<element>AAA. BBB. ccc. ddd.</element>
```

How does this effect result of operations?

For this, the argument *separator* plays a determining role.

In operations, all elements, in any of the expressions-1 or -2, are considered multi value elements (a single value element as a multi value element with just one value). Each value is evaluated for the requested operation individually, one at the time. At the end of this process, when the expression is assembled, the resulting components are 'added' together, sepa-

rated by the string specified by the *separator* argument. (see 4.6 , arguments). This results in two effects:

- If no *separator* argument is entered, or if its value is "|", the before mentioned multi value separator | is placed between the components. The effect of this is described at the start of this section.
- Any other value of the *separator* argument will combine the components in a single string, with this separator string between them.

#### 4.6.1.3 Expression-1 with indices

- Indices in expression-1 are only supported in combination with the commands *remove*, *substring* and *replace*. (not with commands *addstart*, *addend*, *cleanup* and *calculate*)
- Indices in expression-2 are not supported.

As explained in 4.6 - arguments, the argument *type* sets the base of indices used in *expression-1*. (see 4.6 expression-1 and -2 components, indices). These indices specify the position and the length (or number-) of a character (in case of *type=char*), or a word (in case of *type=word*), or a sentence (in case of *type=sentence*), or an element value (in case of *type=element*) (which only makes sense for *multi value elements*). The basic structure and syntax of *expression-1* containing indices is :

```
type=xxx|'element' startposition optional-length}
```

Between 'element' startposition optional-length a space is needed. If 'element' is the same as the element-to-modify, it can be left out of *expression-1*

```
type=xxx|startposition optional-length}
```

Startposition and/or length can also be entered by way of 'element' in which case the integer value of 'element' is used. In this case the element-to-modify cannot be left out! (see also 4.6.4.5.1 # Count and 4.6.4.5.2 @ Index-of)

The effect on the value of element to modify can best be described with :

- ***substitute the actual value with the result of the expression -.***

This is also the case if the *element-to-modify* is another than 'element' :

```
element-to-modify.modify {command(type=)|'element' indices}
```

The resulting value of the *element-to-modify* will be the result of *expression-1*, 'element' *indices*, whatever the original value of it. This original value will be substituted .

The following table illustrates the results of the expressions with indices:

#### 4.6.2 Conditional arguments

##### 4.6.2.1 Pre-Conditional arguments

These conditions are evaluated first. If true the operation is executed. The following conditional operators can be use:

- = string, equal ?, ignore (lower or upper) case , this operator is default and can be omitted
- == string, equal ?, match case
- ~ string, contains ?, ignore case
- ~~ string, contains ?, match case
- not added to one of the operators above (not= not== not~ and not~~) reverses the result
- > numerical, more ?
- < numerical, less ?
- >= numerical, more or equal ?
- <= numerical, less or equal ?

The syntax :

```
('compare-this-element' operator to-this)
```

- 'compare-this-element': The element to evaluate. Enter the name of the element enclosed by ''. If omitted, the element-to-modify is taken. In the case of the numerical operators > < >=

and <=, the element entered will be converted to a floating point number (the floating point conversion of the first number in the string - or 0 (zero) if it does not contain any numerical value)

- operator : one of the conditional operators listed above
- to-this : A "string" (enclosed by "") or an 'element' (enclosed by '), in which case it will be expanded to the value of the element.

A few examples:

```
('element' "abc") result 'true' if the value of element is abc or Abc etc
('element' == "abc") result 'true' if the value of element is abc, false if Abc etc
('element' ~ "abc") result 'true' if the value of element contains abc or ABC etc
('element' ~~ "abc") result 'true' if the value of element contains abc, false if ABC etc
('element' not ~ "abc") result 'true' if the value of element doesn't contain abc
('element' "") result 'true' if the value of element is "" (empty)
('element' 'other-element') result 'true' if the value of element is the value
of other-element
('element' ~ 'other-element') result 'true' if the value of element contains the value
of other-element.
```

Examples when 'element' is left out and the element to modify is taken :

```
element.modify {addstart('other-element')|abc} 'true' if the value of element is the value
of other-element, then abc will be added.
element.modify {addstart(not ~ "abc")|abc} 'true' if the value of element doesn't contain
abc, then abc will be added.
element.modify {addstart("")|abc} 'true' if the value of element is "" (empty), then abc
will be added.
```

Example of numerical conditional arguments:

```
('element' < "10") 'true' if the first numerical value in element is smaller than 2. E.g.
if element = "Episode 9"
```

#### 4.6.2.2 Post-Conditional arguments

These conditions will only be evaluated if the pre-conditions are *true*

*Values* : either *anycase*, *null* or *nonnull*

- *anycase* (*default*) : the operation will be performed regardless any of the conditions described below.
- *null* : the operation will only be performed if the element is *null* (in the case of *addstart* and *addend* commands)
- *nonnull* : the operation will only be performed if the element is *not null* in the case of *addstart* and *addend* commands  
: the operation will *not* be performed if the element will become *null* during/through the operation in the case of *replace* and *remove* commands

#### 4.6.3 Loops

Loops allow to run a set of operations for a number of times or until a certain condition is met.

The syntax:

```
loop {(optional-condition optional-max)|lines}
```

- (*optional*) *condition* : E.g. 'description' ~ "abc" . Must be *true* while the loop is running, when false the loop ends and the operations continue after the last line of the loop. Any of the *pre-conditional-arguments* can be used to specify this condition. When no condition is specified its value is assumed *true*.
- (*optional*) *max* : E.g. *max=6* . Can be used to set the number of times the loop will run if no condition is specified or when the loop doesn't end with a condition value *false* be-

fore reaching this *max* value. If *max* is not specified its default value is assumed 100.

- *lines* : The number of lines (following the line specifying the loop) that are contained in the loop.

A simple example to illustrate:

```
element.modify {addstart|10}
loop {'element' > "0" max=20|1}
element.modify {calculate|1 -}
```

This loop will subtract 1 from element until condition '*element*' > "0" is false, which happens if it reaches 0. The value of *max=20* will not be reached.

#### 4.6.4 The modify commands.

##### 4.6.4.1 Replace

Performs a replacement of all occurrences of the string value of *expression-1* in the element to modify with that of *expression-2*. These expressions may contain *text*, *elements* and *scrubstrings* components (see 4.6 expression-1 and -2 components). Expression-1 may also contain indices if combined with a *type* argument (see 4.6.1.3)

Example:

```
rating.modify {replace(null)|TODOS LOS PÚBLICOS|todos} , replaces the string TODOS LOS PÚBLICOS in element rating by the string todos.
```

##### 4.6.4.2 Remove

The command *remove* comes in two variants depending on the argument *type*:

- Without argument *type* or with *type=string*, it removes all occurrences of the string value of *expression-1* from the element to modify. As with the command *replace*, the *expression-1* may contain *text*, *elements* and *scrubstrings* components.

Example:

```
title.modify {remove(notnull type=string)|: 'subtitle'} * removes the value of element subtitle in title after the colon : , which is also removed.
```

Suppose `{single|<span id=programmeheading|: |</span><br>}` is the scrubstring for the subtitle element, the next gives the same result:

```
title.modify {remove(notnull)|: '{single|<span id=programmeheading|: |</span><br>}'}
```

- With *expression-1* containing indices, combined with argument *type=char* or *word* or *sentence* or *element*. In this case the part of the element, determined by the *indices* will be removed. (see 4.6.1.3)

##### 4.6.4.3 Substring

The command *substring* extracts parts of an element determined by the *indices* in *expression-1*. The result is the opposite of *remove* when this is used with indices. As, command *substring* only works with *expression-1* containing indices, the argument *type* with one of the values *char*, *word*, *sentence* or *element* is required . (see 4.6.1.3)

##### 4.6.4.4 Addstart and Addend

These commands simply add the result of *expression-1* to the *element-to-modify*. With these commands indices in *expression-1* are not supported.

##### 4.6.4.5 Calculate

This command allows simple arithmetic calculations. Supported are + (add) - (subtract) \* (multiply) / (divide). Furthermore there are two special calculations supported; # (count, see 4.6.4.5.1 ) and @ (index-of, see 4.6.4.5.2)

Its syntax is based on RPN (Reverse Polish Notation), which differs from the standard  $a + b$  and uses  $a b +$ . Its advantage is that it avoids complex syntax like  $(a + b) * c$  which cannot be expressed without  $()$  in the standard way. In RPN it is simply  $a b + c *$

However, because it is thought that more complex calculations will seldom be necessary only a simplified version of RPN is implemented. Like the standard  $(a + b) * c / (d - e)$  would be  $a b + c * d e - /$  in full RPN. Here we must do that in three steps : (step-1)  $a b + c *$  then (step-2)  $d e -$  then (step-3)  $result-1 result-2 /$

The complete syntax of calculations :

```
Element.modify {calculate (optional arguments)|RPN expression}
```

It can also be used without any RPN expression, in that case the element value is only converted to a numeric value in the format specified by the *format* argument:

```
Element.modify {calculate (optional arguments)}
```

A few examples :

```
temp_1.modify {calculate(format=F2)|'temp_1' 240 /}
```

Divides temp\_1 by 240 and assigns the result back to temp\_1. If temp\_1 is not a numeric string, its value will be zero. If it contains numeric string(s), its value will be the value of first numeric string in it. For example suppose temp\_1 has the value *width=576px* , it will be converted into 576 and the result will be 2.40

In this first example the element to modify temp\_1 is the same as the element from which the value is used. In that case the following is the same:

```
temp_1.modify {calculate(format=F2)|240 /}
```

```
temp_1.modify {calculate(not "0" format=F2)|2 +} Adds 2 to temp_1 if temp_1 is not "0". Uses the (pre)conditional expression not "0" , see 4.6.2
```

```
index_start.modify {calculate(format=time)|'previous_start' 'index_temp_1'+ 'previous_index_temp_2'+} Use of the previous_start and previous_index_temp_2 elements . In it, the value of the 'same' element is stored of the 'previous' show. (see 4.5.1 , 4.5.2 and APPENDIX C). Here index_start is calculated as the previous_start added by the value of index_temp_1 and the value of previous_temp_1.
```

Suppose the start time is given in any of the three supported numerical values of the *format=time* argument (see 4.6 arguments *format*)

```
index_start.modify {calculate(format=time)} which is the same as
```

```
index_start.modify {calculate(format=time)|1 *} will convert the element value in the hh:mm time format.
```

#### 4.6.4.5.1 # Count:

It will return the number of occurrences that a certain string is contained in an element.

Count comes in two variants:

- 1. Occurrence. Without argument *type* (or with *type=string*):

It will return the number of occurrences that a certain string is contained in an element.

The syntax:

```
element.modify{calculate(optional-arguments)|'other-element' string #} or
element.modify{calculate(optional-arguments)|string #}
```

- 'other-element' : the element in which the number of occurrences of "string" is counted. If 'other-element' is omitted the element to modify is evaluated for the occurrence of "string".

- string : if entered as string value, like ";" it must be enclosed by "" to allow spaces in it. It can also be entered as 'element' , enclosed by ', like 'title', in which case the number of occurrences of the value of that element will be counted.

- # : the operant for count.

Example:

```
starrating.modify {calculate(format=F0)|"*" #} This will count the number of * characters in starrating and assign that value to it in the F0 format (0 decimal digits). Suppose starrat-
```

ing is `***`, then after this operation it will become 3.

2. Length: With argument *type* values *char*, *word*, *sentence* or *element*. Returns the *Length* of the *element*. The syntax:

```
element.modify{calculate(type=xx optional-arguments)|#} or
element.modify{calculate(type=xx optional-arguments)|'other-element' #}
```

- argument *type* : See 4.6, arguments.

If *type=char*, the length returned is the number of *characters* in *element* or *other-element*. Similarly, if *type=word*, *sentence* or *element* length is the number of *words*, *sentences* and *element values* respectively.

- optional-arguments: An obvious one here is *format*. (see 4.6 arguments)

#### 4.6.4.5.2 @ Index-of:

This will return the starting position (index) of a certain string contained in an element. The result is 'index based' (the first The syntax:

```
element.modify{calculate(optional-arguments)|'other-element' string @} or
element.modify {calculate(optional-arguments)|string @} }
```

- 'other-element' : the element in which start location of "string" is determined. If 'other-element' is omitted the element to modify is evaluated for the location of "string".  
- string : if entered as string value, like ";" it must be enclosed by "" to allow spaces in it. It can also be entered as 'element' , enclosed by '' , like 'title', in which case the start location of the value of that element will be returned.

- @ the operant for the index-of . If "string" occurs more than once it will return the start position of the first occurrence. If prefixed by a minus-sign, like -@ , the start position of last occurrence will be returned.

- (optional) argument *type* : See 4.6, arguments.

It specifies the index-base. Possible values are: *type=string* (default, the result is returned as string position), *type=char* (the result is returned as character positions), *type=word* (the result is returned as word positions), *type=sentence* (the result is returned as sentence positions) and *type=element* (the result is returned as element positions)

- Note: If the "string" doesn't occur in the element, -1 will be returned. Also, note that with types *word*, *sentence* and *element*, if in a word, sentence or element, the "string" is contained the index position it is returned.

Example: an element with value "the quick brown fox" and a "string" with value "own" .

If :

*type=char* : result = 12

*type=word* : result = 2 (from the word : brown)

*type=sentence* : result = 0 (this sentence contains string "own")

*type=element* : result = 0 (this element has only own value which contains "own")

#### 4.6.4.5.3 Date and time calculations

With Date and Time calculations it is possible to add of subtract timespan values from date or time values. Its syntax:

```
Element.modify {calculate(opt.args. format=time/date)|timespan +/-}
```

- timespan : The timespan to add or subtract. If *format=time* specify *hours:minutes*, if *format=date* specify *days:hours:minutes* or *days:hours*.

Examples:

`element.modify {calculate(format=time)|2:15 +}` gives the following results:

If element = 16:20 --> 18:35

If element = 6.5 --> 08:45 (the numeric value 6.5 is first converted to 6:30)

If element = 23.25 --> 01:30

`element.modify {calculate(format=date,yyyy/MM/d H:mm)|1:5:30 -}` gives the following results:  
If `element = 2011/11/15 9:55` --> `2011/11/14 4:25`  
If `element = 1314866400` --> `2011/08/31 3:10` (the numeric UNIX date value 1314866400 is first converted to `2011/09/01 8:40`)  
See 4.6 *arguments*, *format* for details about date en time formats and date-time format strings.

#### 4.6.4.6 Cleanup

This can be useful to tidy-up the result of a scrubbed element. It:

- tries to remove remaining html tags.
- replaces newline `\n` and tabs `\t` characters by a space.
- removes carriage returns.
- replaces multiple spaces by single spaces
- removes illegal xml characters.
- restores Unicode character sequences like `\\u00e6` with the actual chars
- performs optional upper- and lower case conversions depending on the *style* argument.

Note that it is allowed to add newline `\n` and tabs `\t` to elements with the `addstart`, `addend` and `replace` command. If a cleanup is executed after this is done, they will be removed again. Cleanup should be executed before these operations in such cases.

Its syntax:

`Element.modify {cleanup(optional arguments)}`

- optional argument: Cleanup has its own dedicated argument *style*. (see 4.6, *arguments*). This argument can be added to specify the required style of the cleanup result. Possible values are *style=sentence*, *style=name*, *style=upper* (convert to UPPERCASE) and *style=lower* (convert to lowercase)

#### 4.6.4 Examples of operations

`description.modify {addstart("")|no details}` adds - no details - to an empty description  
`description.modify {addstart(null)|no details}` same as above, adds - no details - to an empty description

`subtitle.modify {addstart(not "")|Episode: }` adds -- Episode: -- before the subtitle, but only if subtitle wasn't empty before the action.

`rating.modify {replace("")|nine|9}` replaces the word 'nine' by the number 9, also if the word 'nine' is the whole rating

`description.modify {replace(not "")|Afl.: 'subtitle'.| This episode:}` replaces the subtitle listing like - Afl.:Heads Up. - in the description by -- This episode: -- but only if the action doesn't replace the whole description.

`ratingicon.modify {addstart("")|'rating'.png}` adds a ratingicon if the site doesn't list one.

As an example of the use of a scrubstring in string-1, consider the following html:

```
<p class="verhaal">Amerikaanse (USA) Drama uit 1995 van Taylor Hackford. Met: Kathy Bates, Jennifer Jason Leigh, Judy Parfitt, Christopher Plummer e.a. Huishoudster Dolores wordt beschuldigd van de moord op haar vervelende en veeleisende werkgeefster, bij wie ze al jarenlang in dienst was. Door deze gebeurtenis wordt ook de dood van haar man, twintig jaar geleden, weer opgerakeld en rechercheur John Macky is vastbesloten Dolores dit keer wel voor moord achter slot en grendel te krijgen. Haar dochter Selena, een succesvolle journaliste in New York, keert voor de zaak terug naar haar geboorteplaats in het kille Maine, waar ze gelijk weer met haar eigen jeugd geconfronteerd wordt.</p>
<table style="width: 60%;">
<tr><th width="65">Genre</th><td>Film</td></tr><tr><th>Acteur</th><td>Kathy Bates, Jennifer Jason Leigh, Judy Parfitt, Christopher Plummer</td></tr>
```

```
<tr><th>Regisseur</th><td>Taylor Hackford</td></tr>
</table>
```

This site lists the actors double, in the description after - Met: - and later following <th>Acteur .

To scrub the description and the actor we use:

```
description.scrub {single|<p class="verhaal">|</p>|<table>
actor.scrub {single(separator=", "|<th>Acteur</th><td>|</td></tr>}
```

That leaves use with a description that contains all the actors which isn't perfect. However they can be removed with the following:

```
description.modify{remove(null)|Met: '{single|<p class="verhaal">|Met:|e.a.|<table>}' e.a.}
```

We cannot use the element actor in string-1 as in

```
description.modify{remove(null)|Met: 'actor' e.a.} because actor is not a single value xmltv
element anymore due to the use of the argument - separator="," - in the actor scrub speci-
fication.
```

We allow - null - to be sure that the actors listing is removed even if it is the whole description. We add the following to have at least something in the description:

```
description.modify {addstart(null)|No details}
```

An example with calculate and numeric conditional arguments:

Suppose the subtitle of a show is the first sentence in the description on the html detail page. So we use something like :

```
subtitle.scrub {single(separator=". " include=first)|. . . . . }
```

However not all shows have a subtitle, consequently the result can also contain just the first sentence of the description. To distinguish between subtitle and a normal sentence, count the words .. max 3 words is considered a subtitle (or at least most probable)

```
temp_1.modify {calculate(not "")|'subtitle' " " #} * count the spaces
subtitle.modify {remove('temp_1' > "2")|'subtitle'} * clear subtitle if more that 3 words
description.modify {remove('temp_1' < "3")|'subtitle'} * remove the subtitle from the de-
scription.
```

## 5. Tricks

- Force a show update like this:

```
title.modify{addend(~ "NOS WK Voetbal")|(!)}
```

By addition of (!) (or also (?)) to the title WebGrab+Plus will update the show despite the update decision outcome. This can be useful for shows that could have last minute changes that are not apparent from the index show times and index title. To get this update scheduled for already existing shows in the xmltv file a one-time full update run is necessary (set the update attribute to - f - in the channel entry of the WebGrab++.config.xml file for one run, like: <channel update="f" site="tvgids.nl" site\_id="1" xmltv\_id="NED1-tvgids">NED1</channel>)

- Avoid unnecessary show update:

If the index\_title is different from the title on the show detail page and a lower setting of the title match factor is unacceptable (too low for reliable title comparison) try the following:

```
title.modify{replace(null)|Sterren 24|'index_title'}
```

The show in this example has - Sterren.nl Extra - as index\_title and - Sterren 24 - as show detail title, which is too much difference even for a title match factor of 50. With this we simply replace the title with the index\_title for a show with title - Sterren 24 - only.

- Full rating to Short rating

Sites normally list ratings like KIIKWIJZER ratings in a sentence like - Afgeraden voor kinderen jonger dan 9 jaar - of - Let op met kinderen tot 9 jaar - of - drugs- en/of alco-

holmisbruik -

The ratingicon is normally listed as a link to a picture file like - <http://u.omroep.nl/gids/pics/icons/kijkwijzer/negen.png>- or - <http://u.omroep.nl/gids/pics/icons/kijkwijzer/groftaalgebruik.png>-

With the help of the modify operations they can be simplified easily (e.g. for site tvgids.nl):

```
rating.modify {replace(null)|Afgeraden voor kinderen jonger dan 6 jaar|6+}
rating.modify {replace(null)|Let op met kinderen tot 9 jaar|9+}
rating.modify {replace(null)|Afgeraden voor kinderen jonger dan 12 jaar|12+}
rating.modify {replace(null)|Niet voor personen tot 16 jaar|16+}
rating.modify {replace(null)|Grof taalgebruik|Grof}
rating.modify {replace(null)|Drugs- en/of alcoholmisbruik|Drugs}
ratingicon.modify {replace(null)|http://u.omroep.nl/gids/pics/icons/kijkwijzer/zes.png|6.png}
ratingicon.modify {replace(null)|http://u.omroep.nl/gids/pics/icons/kijkwijzer/negen.png|9.png}
ratingicon.modify {replace(null)|http://u.omroep.nl/gids/pics/icons/kijkwijzer/twaalf.png|12.png}
ratingicon.modify {replace(null)|http://u.omroep.nl/gids/pics/icons/kijkwijzer/16.png|16.png}
ratingicon.modify {replace(null)|http://u.omroep.nl/gids/pics/icons/kijkwijzer/seks.png|seks.png}
ratingicon.modify {replace(null)|http://u.omroep.nl/gids/pics/icons/kijkwijzer/eng.png|angst.png}
ratingicon.modify {replace(null)|http://u.omroep.nl/gids/pics/icons/kijkwijzer/geweld.png|geweld.png}
ratingicon.modify {replace(null)|http://u.omroep.nl/gids/pics/icons/kijkwijzer/groftaalgebruik.png|grof.png}
ratingicon.modify {replace(null)|http://u.omroep.nl/gids/pics/icons/kijkwijzer/discriminatie.png|discriminatie.png}
ratingicon.modify {replace(null)|http://u.omroep.nl/gids/pics/icons/kijkwijzer/drugs.png|drugs.png}
```

A set of ratingicon files for the Dutch 'KIJKWIJZER' ratingsystem with the filenames as used in this example is included in the WebGrab+Plus distribution.

- Some *exclude* and *include* tricks:

- Exclude elements with more than one word (sentence)

```
element.scrub {single (exclude=" ")}|scrubstring}
```

This is very helpful to filter out (in general) one-word elements like category. A lot of sites use unsystematic html structures. Especially secondary elements like category can be mixed with other elements like *title* or *subtitle* behind the same *bs es ee* and *bs* values. This filter can help to separate them.

- Include a list of standard values

```
category.scrub {single (include="film""serie""documentary""sports")|scrubstring}
```

Another way to filter a category out of a mixed html scrubbed element. The list of strings to include should contain all the possible categories which occur in the html scrub.

- A combination of *include* and *exclude* to further refine the result.

Consider the following scrubstring:

```
category.scrub {single (include="film""serie""documentary""sports" exclude=" ")}|scrubstring}
```

The include list allows to yield an element with the value - memories of a seriekiller - , which probably is a title and not a category. The *exclude=" "* removes this element value.

—————END—————

## APPENDIX A XMLTV Elements, sample xmltv file format

```
<?xml version="1.0" encoding="UTF-8"?>
<tv generator-info-name="WebGrab+Plus version 1.0.7 Beta -- Jan van Straaten" generator-info-
url="http://www.servercare.nl">
  <channel id="xxxx">
    <display-name lang="xx">xxxx</display-name>
    <url>http://xxxx.xx</url>
    <icon>xxxx</icon>
  </channel>
  <programme start="yyyyMMddhhmss +hhmm" stop="yyyyMMddhhmss +hhmm" channel="xxxx">
    <title lang="xx">xxx xxxx</title>           (multiple language entries supported)
    <title lang="yy">xxxx xxx</title>          (multiple language entries supported)
    <sub-title lang="xx">xxxxx xxx</sub-title>
    <desc lang="xx">xxx xxxx xx xx</desc>
    <credits>
      <director>xxx xxxxx</director>          (multiple entries supported)
      <actor>xxx xxxxx</actor>                (multiple entries supported)
      <presenter>xxxx xxxx</presenter>        (multiple entries supported)
      <writer>xxxx xxxx</writer>              (multiple entries supported)
      <producer>xxxx xxxx</producer>          (multiple entries supported)
      <composer>xxxx xxxx</composer>          (multiple entries supported)
    </credits>
    <rating system="xxxxx">                   ↑
      <value>xxxx</value>                     (multiple entries supported)
      <icon src="xxxx.png" />                 (multiple entries supported)
    </rating>                                 ↓
    <category lang="xx">xxxx xxxxx</category> (multiple entries supported)
    <date>yyyy</date>
    <star-rating system="xxxxx">
      <value>x/x</value>
    </star-rating>
    <episode-num system="xxxx">xx/xx.x</episode-num>
    <video>
      <aspect>16:9</aspect>
      <quality>HDTV</quality>
    </video>
    <subtitles type="teletext"/>
    <previously-shown/>
  </programme>
</tv>
```

## APPENDIX B Example WebGrab++.config.xml file

```
<?xml version="1.0"?>
<!-- Configuration file for WebGrab+Plus, the incremental Electronic-Program-Guide web grabber
by Jan van Straaten, May 2011
Version V1.0.7 Beta
new in this version:
extended retry setting, possibility to disable skip, new time-offset channels-->
<settings>
<!-- filename
The path (required) + filename where the epgguide xml file is /will be located. It must include
drive and folder. Like C:\ProgramData\ServerCare\WebGrab\guide.xml
If the file already exist (from last run or from another xmltv source) it will read it and use
what fits the requested output. In that case the file will be updated. If no such file exist it
will be created.
Change the following to your own needs -->
<filename>C:\ProgramData\ServerCare\WebGrab\guide.xml</filename>
<!-- modes:
d or debug      saves the output xmltv file in a file with -debug addition in the file name .
                 The original xmltv file will be kept.
m or measure    measures the time for each updated show or new show added
n = nomark      disables the udate-type marking (n) (c) (g) (r) at the end of the description
v or verify     verifies the result following a channel update
w or wget       use wget as grab engine (can improve site recognition)
Note that modes can be added in one line, separated by comma's or spaces, or both. -->
<mode>m</mode>
<!-- proxy:
This setting is only required if your computer is connected to internet behind a proxy
specify proxy address as ip:port like <proxy>192.168.2.4:8080</proxy>
or as <proxy>automatic</proxy> which attempts to read the proxy address from your connection set-
tings. If your proxy requires a username and password, add them like
<proxy user="username" password="password">192.168.2.4:8080</proxy> -->
<proxy>automatic</proxy>
<!-- user agent:
The user agent string that is sent to the tvguide website. Some sites require this. Valid values
are either <user-agent>random</user-agent>, in which case the program generates a random string,
or any other string like <user-agent>Mozilla/5.0 (Windows; U; MSIE 9.0; WIndows NT 9.0; en-US)</
user-agent>-->
<user-agent>Mozilla/5.0 (Windows; U; MSIE 9.0; WIndows NT 9.0; en-US)</user-agent>
<!-- logging:
simply put 'on' in there to start logging, anything else will turn it off -->
<logging>on</logging>
<!--retry
The most simple form of retry defines the amount of times the grabber engine should attempt to
capture a web page before giving up and continuing with the next page, like <retry>4</retry>
It is also the place to specify delays between retries and the grabbing of html pages with the
following attributes: timeout; the delay between retries (default is 10 sec), channel-delay; the
delay between subsequent channels (default is 0), index-delay; the delay between the grabbing of
index pages (default is 0), show-delay; the delay between the grabbing of detail show pages
(default is 0). In the most complete version it will look like this:
<retry time-out="5" channel-delay="5" index-delay="1" show-delay="1">4</retry> -->
```

```

<retry time-out="5">4</retry>

<!--skip
It takes two values H,m separated by a comma:
The first H : if a show takes more than H hours, it's either tellsell or other commercial fluff, or
simply a mistake or error, we want to skip such shows.
The second m : if a show is less or equal than m minutes it is probably an announcement , in any
case not a real show.
When entered as <skip></skip> the defaults are 12 hours, 1 minute. To disable this function enter
<skip>noskip</skip> or just leave out this entry completely-->

<skip>13, 1</skip>

<!--timespan
The timespan for which shows will be grabbed.
It takes one or two values separated by a comma. The first is the number of days (including today)
to download, note that 0 is today. The second (optional) is a time specified between 0:00 and 24:00
which will reduce the download to only the one show (per day) which is scheduled around the speci-
fied time. Any value between start time (including) and stop time will do
This -one-show-only mode is helpful if a SiteIni file needs to be debugged-->

<timespan>0</timespan>

<!-- update mode
i or incremental      only updates of changes , gabs, repairs and new shows
l or light            forces update of today and new shows, rest as incremental
s or smart           forces update of today and tomorrow and new shows, rest as light
f or full or force   forces full update

If one of these values is entered here it will apply to all channels selected for update
(see channel). This value overrules the value of 'update' for in the individual channels
If no value is entered here the individual 'update' values from the channellist are taken -->

<update></update>

<!-- The channel-list :
Each channel to be grabbed has a separate entry in the list, the most common form is:
<channel update=.. site=.. site_id=.. xmltv_id=.. >display-name</channel>
Besides this form, there is a possibility to specify special channels like 'combi-channels' and
'timeoffset-channels', see further down for more information-->

<!-- Channel list files :
The easiest way to compose this channel-list is to copy the required channels from the channel-list
files which can be found in the SiteIni.Pack for nearly every supported tvguide site. -->

<!-- update :
The mode values here can be set for each channel differently if not overruled by the general update
setting (see above). Allowed values are as the same as the general update setting. Any other value
will be ignored. If any of the allowed values of 'update' is entered, this channel will be updat-
ed , no value no update ! In that case the epg data of that channel will remain as it is. -->

<!-- site:
The website to be used to get the EPG from. The value entered here is the name of the .ini file
that supplies the specific parameters for the site without .ini extension.
e.g tvgids.nl.ini becomes site="tvgids.nl" and gids.publiekeomroep.nl.ini becomes
site="gids.publiekeomroep.nl".-->

<!-- site_id:
This is the number or text used by the site as reference to the correct html page for this channel.
It is used by the program to compose the url for the shows for a channel. For nearly all sites sup-
ported by the program a channel-list file is provided in the siteini-pack. It list most of the
available channels including this site_id -->

<!-- xmltv_id :
The xmltv_id can be any string that suits your needs, you will find it back as the "channel" in

```

your xml file as in :

```
<programme start="20100218072500 +0200" stop="20100218075500 +0200" channel="RTL7-id"> -->
```

```
<!-- display-name:
```

This will be used in the xmltv file to give the channel's displayname. That is the name the epgprogram will use to display the channel. Give it any value you like. It is no problem if site\_id , xmltv\_id and display-name are equal -->

```
<!-- Important !
```

Be aware that all channels entered here will be included in the xmltv channel table even if no update is requested. This allows the update of individual channels without affecting the data of the others in the list. A channel not in this list will be removed from your xmltv listing together with all the show data of it if found there by WebGrab+Plus. (If you use WebGrab+Plus with a xmltv input file from another source, it will remove all data from channels not in this list and create an entry for new channels)

WebGrab+Plus uses the xmltv\_id to identify a channel in an existing xmltv file.-->

```
<channel update="f" site="tvguide.co.uk" site_id="145" xmltv_id="Film4">Film4</channel>
```

```
<channel update="i" site="bfbs.com" site_id="8001" xmltv_id="BFBS">BFBS</channel>
```

```
<channel update="i" site="yelo.be" site_id="ned1" xmltv_id="NED1.wg">NED1</channel>
```

```
<channel update="" site="sincroguia.tv" site_id="18" xmltv_id="LA1">LA1</channel>
```

```
<channel update="" site="laguiatv.com" site_id="Tele+5" xmltv_id="Tele5">Tele5</channel>
```

```
<channel update="i" site="directv.com" site_id="554" xmltv_id="TMCeHD(r)">MovieEastHDR</channel>
```

```
<channel update="i" site="tvgids.nl" site_id="1" xmltv_id="NED1">NED1</channel>
```

```
<!-- Timeoffset-channels. Many sites list channels that differ only from another through a time difference. Instead of grabbing the epg separately it is possible to just copy and timeshift the 'source' channel with a special channel specification.
```

For that use the attributes same\_as and offset as follows:

Example of timeoffset-channels :

```
<channel update="i" site="laguiatv.com" site_id="Canal +" xmltv_id="Canal +">Canal +</channel>
```

```
<channel same_as="Canal +" offset="2" xmltv_id="Canal + 2">Canal + 2</channel>
```

The source channel (here ="Canal +" ) must always be listed before the timeoffset-channel (here "Canal + 2") The offset can also be negative like offset="-1"

```
<!-- Combi-channels. With these one can combine parts several channels in combi-channel. These parts can consist of daytime periods or shows with certain subjects. Please refer to Combi-Channels-Guide.txt for more info. The arguments period, include/exclude and site_channel can be used to specify these 'combi-channels' See the separate guide how-->
```

Example of a combi-channel:

```
<channel update="i" site="tvgids.nl" site_id="40" site_channel="AT5" xmltv_id="CombiChannel_Id" period="00:00-06:00" >CombiChannel_Name</channel>
```

```
<channel update="i" site="gids.publiekeomroep.nl" site_id="67" site_channel="RTL8" xmltv_id="CombiChannel_Id" period="06:00-24:00" >CombiChannel_Name</channel>
```

```
<channel update="i" site="tvgids.upc.nl" site_id="Ered.+live+2" site_channel="EredivisieLive2" xmltv_id="CombiChannel_Id" exclude="Eredivisie Live Tekst TV">CombiChannel_Name</channel> -->
```

```
</settings>
```

## APPENDIX D Example SiteIni files

### For site tvguids.nl, the 'standard' dutch TV guide site

```
* WebGrab+Plus ini for grabbing EPG data from TvGuide websites
* Site tvguids.nl
* revision 1 Added index_site_channel and index_site_id
* revision 2 improved index_date.scrub, missing be
* revision 3 Adapted for site changes
* revision 4 Adapted for V.1.0.5
* revision 5 Small corection in director, catch Film op 2 as film serie
* Jan van Straaten 16 Jan 2011
*
site {url=tvguids.nl|timezone=UTC+01:00|maxdays=6|cultureinfo=nl-NL|charset=ISO-8859-1}
site {titlematchfactor=90|ratingsystem=KIJKWIJZER}
url_index{url|http://www.tvguids.nl/zoeken/?q=&d=|urldate|&z=|channel|&t=0&g=&v=0}
urldate.format {daycounter|0}
index_urlshow {url|http://www.tvguids.nl|<a href=||>}
index_showsplit.scrub {multi|<div class="programs">|<div class="program">|</div>|<script>}
index_date.scrub {single|<div class="programs">|<h2>|</h2>|</h2>}
index_start.scrub {single|<span class="time">| |</span>}
index_stop.scrub {single|<span class="time">|</span>|</span>}
index_title.scrub {single(separator=":")|<span class="title">|</span>|</span> } * splits at the :
character to enable title comparison
*the following 2 entries create a site channel file
*index_site_channel.scrub {multi(separator=">" exclude="="" --")|<option value=||</option>|</option>}
*index_site_id.scrub {multi()|<optgroup label="Hoofdzenders">|<option value="|" |<input
type="submit"}
*
title.scrub {single(separator=": " include=first)|<h2>In het kort</h2>|<strong>Titel:</strong>|</
li>|<div id="prog-info-footer">|</div>}
subtitle.scrub {single|div id="prog-content">|alt="|" />|</div>} * at the beginning of the descrip-
tion
subtitle.scrub {single(separator=": " exclude=first)|<h2>In het kort</h2>|<strong>Titel:</
strong>|</li>|<div id="prog-info-footer">|</div>}
*subtitle.scrub {single()|<strong>Titel aflevering:</strong>|<br />|</li> } * rare, some found in
Hallmark
description.scrub {multi(exclude="<a href=")|div id="prog-content">|<p>|</p>|</div>} * multi be-
cause it can have more than one paragraph! exclude because some showdescriptions contain a refer-
ence (with <a href=) to a 'trailer'
director.scrub {single(separator=", ")|<h2>In het kort</h2>|<strong>Regisseur:</strong>|</li>|<div
id="prog-info-footer">|</div>}
actor.scrub {single(separator=", ")|<h2>In het kort</h2>|<strong>Acteurs:</strong>|</li>|<div
id="prog-info-footer">|</div>}
presenter.scrub {single(separator=", ")|<h2>In het kort</h2>|<strong>Presentatie:</strong>|</
li>|<div id="prog-info-footer">|</div>}
rating.scrub {multi(exclude="style=")|<strong>Uitzendtijd:|" alt="|" />|<div class="}
ratingicon.scrub {multi|<strong>Uitzendtijd:|In het kort</h2>|<strong>Genre:</strong>|</li>|<div id="prog-info-
footer">|</div>}
category.scrub {single|div id="prog-content">|<strong>|</strong>|</div>}
productiondate.scrub {single|<h2>In het kort</h2>|Jaar van premiere:</strong>|</li>|<div id="prog-
info-footer">|</div>}
*the following lines catch the film series on NED2, used to manipulate title and subtitle (below)
temp_1.scrub {single(separator=":" include="Cinema 2")|<h2>In het kort</h2>|<strong>Titel:</
strong>|</li>|<div id="prog-info-footer">|</div>}
temp_1.scrub {single(separator=":" include="NPS Wereldcinema")|<h2>In het kort</
```

```

h2>|<strong>Titel:</strong>|</li>|<div id="prog-info-footer"></div>}
temp_1.scrub {single(separator=":" include="Zomergast film")|<h2>In het kort</h2>|<strong>Titel:</
strong>|</li>|<div id="prog-info-footer"></div>}
temp_1.scrub {single(separator=":" include="filmzomer")|<h2>In het kort</h2>|<strong>Titel:</
strong>|</li>|<div id="prog-info-footer"></div>} * like Franse filmzomer
temp_1.scrub {single(separator=":" include="Telefilm")|<h2>In het kort</h2>|<strong>Titel:</
strong>|</li>|<div id="prog-info-footer"></div>}
temp_1.scrub {single(separator=":" include="Film1ab")|<h2>In het kort</h2>|<strong>Titel:</
strong>|</li>|<div id="prog-info-footer"></div>}
temp_1.scrub {single(separator=":" include="Film op 2")|<h2>In het kort</h2>|<strong>Titel:</
strong>|</li>|<div id="prog-info-footer"></div>}
*the following 3 lines swaps title and subtitle in case of Film series (in temp_1) on NED2
title.modify {replace(null)|'temp_1'|'subtitle'} * replace film serie title in temp_1 (like 'Cinema
2') with the film title in subtitle
subtitle.modify {addstart(notnull)|'temp_1': } * adds film serie title in temp_1 (like 'Cinema 2')
to the subtitle
*subtitle.modify {remove(notnull)|: 'title'} * removes the film title from the subtitle
subtitle.modify {remove|&}
subtitle.modify {replace|#039;|\'}
description.modify {replace|</strong>|: }
description.modify {replace|#039;|\'}
description.modify {replace|nbsp;| }
description.modify {remove|rdquo;}
description.modify {remove|ldquo;}
description.modify {remove|&}
description.modify {cleanup}
*convert to short ratings :
rating.modify {replace(null)|Voor alle leeftijden|Alle}
rating.modify {replace(null)|Afgeraden voor kinderen jonger dan 6 jaar|6+}
rating.modify {replace(null)|Afgeraden voor kinderen jonger dan 9 jaar|9+}
rating.modify {replace(null)|Afgeraden voor kinderen jonger dan 12 jaar|12+}
rating.modify {replace(null)|Niet voor personen tot 16 jaar|16+}
rating.modify {replace(null)|Grof taalgebruik|Grof}
rating.modify {replace(null)|drugs- en/of alcoholmisbruik|Drugs}
ratingicon.modify {replace(null)|http://tvguidesassets.nl/img/kijkwijzer/alle_transp.png|alle.png}
ratingicon.modify {replace(null)|http://tvguidesassets.nl/img/kijkwijzer/6_transp.png|6.png}
ratingicon.modify {replace(null)|http://tvguidesassets.nl/img/kijkwijzer/9_transp.png|9.png}
ratingicon.modify {replace(null)|http://tvguidesassets.nl/img/kijkwijzer/12_transp.png|12.png}
ratingicon.modify {replace(null)|http://tvguidesassets.nl/img/kijkwijzer/16_transp.png|16.png}
ratingicon.modify {replace(null)|http://tvguidesassets.nl/img/kijkwijzer/
geweld_transp.png|geweld.png}
ratingicon.modify {replace(null)|http://tvguidesassets.nl/img/kijkwijzer/grof_transp.png|grof.png}
ratingicon.modify {replace(null)|http://tvguidesassets.nl/img/kijkwijzer/angst_transp.png|angst.png}
ratingicon.modify {replace(null)|http://tvguidesassets.nl/img/kijkwijzer/
discriminatie_transp.png|discriminatie.png}
ratingicon.modify {replace(null)|http://tvguidesassets.nl/img/kijkwijzer/drugs_transp.png|drugs.png}
ratingicon.modify {replace(null)|http://tvguidesassets.nl/img/kijkwijzer/seks_transp.png|seks.png}

```

## Site TVguide.co.uk, index only version

```

* WebGrab+Plus ini for grabbing EPG data from TvGuide websites
* site tvguide.co.uk - index-only version
* revision 1
* Jan van Straaten 18 Jan 2011
*
site {url=tvguide.co.uk|timezone=UTC+00:00|maxdays=6|cultureinfo=en-GB|charset=ISO-8859-1}
site {titlematchfactor=50}
url_index{url|http://www.tvguide.co.uk/channellisting.asp?ch=|channel|&cTime=|urldate|}

```

```

urldate.format {datestring|MM/dd/yyyy}
index_urlshow {url|http://www.tvguide.co.uk/detail.asp?id=|<td height="133"|<a
href="javascript:popup(|)|target=}
index_showsplit.scrub {multi(|<table border="0" cellpadding="0" ||<tr><td> </td></tr>|<tr><td> </
td></tr>}
index_start.scrub {single|<td height="133"|<span class="tvchannel">| </span><br>|<a href=}
index_title.scrub {single(separator=": " include=first)|<td height="133"|<span
class="programmeheading" >|</span>|<span class="programmetext">}
index_subtitle.scrub {single(separator="(" include=first)|<span class="programmeheading" >|: |</
span>|<span class="programmetext">}
index_description.scrub {single|<td height="133"|<span class="programmetext">|</span></a>|</span></
a>}
index_category.scrub {single|<span class="tvchannel">Category|<span class="programmetext">|</
span>|</span></a><br>}
index_productiondate.scrub {single(separator="(" include=last)|<span class="programmeheading" >|:
|</span>|<span class="programmetext">}
index_starrating.scrub {single|<span class="programmetext">Rating<br>|<span
class="programmeheading">|</span>|</tr></table>}
*
index_starrating.modify {addend(notnull)|/10} * adds /10 divider to starrating (MCE requirement)
index_description.modify {addstart(null)|No details} * adds a description if none

```

## Site Osnetwork.com, a siteini that needs the command calculate

```

* WebGrab+Plus ini for grabbing EPG data from TvGuide websites
* Site : osnetwork.com
* revision : 0
* Jan van Straaten Febr 2011
* Listing always for one week starting sunday
*
site {url=osnetwork.com|timezone=UTC+00:00|maxdays=7.1|cultureinfo=en-GB|charset=ISO-8859-
1|titlematchfactor=90}
url_index{url(debug)|http://www.osnetwork.com/onlineguide/schedules/GuideListingsWeekly_en_gb.aspx?
Channel=|channel|&Start=|urldate}
urldate.format {datestring|d|en-US}
index_showsplit.scrub {multi(exclude="width:-")|<label style=||</label>|</label>}index_temp_1.scrub
{single(|margin|left:|px|</span>} * left margin in pixels
index_temp_2.scrub {single(|margin|width:|px|</span>} * duration in pixels
index_urlshow {url(|http://www.osnetwork.com/|href="|" |" onclick|</span>}
index_title.scrub {single(|margin|<a title="|" href=|</span>}
*
title.scrub {single(separator=": " include=first)|<div id="cnumCont">|<h2>|</h2>|<form
name="aspnetForm"}
subtitle.scrub {single(separator=": " exclude=first)|<div id="cnumCont">|<h2>|</h2>|<form
name="aspnetForm"}
description.scrub {single|<p id="Program_pSynopsis" class="Synopsis">|<span>|</span>|</p>}
director.scrub {single|<strong>Director:</strong>|<span>|</span>|</li>}
actor.scrub {single(separator=", ")|<strong>Starring:</strong>|<span>|</span>|</li>}
rating.scrub {single|<strong>Rating:</strong>|<span>|</span>|</li>}
productiondate.scrub {single(|<div id="cnumCont">| - |</h3>|</strong>}
*
* operations:
index_temp_1.modify {calculate(not "0" format=F2)|2+}*left margin offset correction, add 2 if not 0
index_temp_1.modify {calculate(format=F2)|240/} *left margin in decimal hours
index_temp_2.modify {calculate(format=F2)|240/} *program duration in decimal hours
* we do not use index_duration because that will force stop = start + duration
* instead we use/ need here start = previous_start + margin + previous duration :

```

```
index_start.modify {calculate(format=time)|'previous_start' 'index_temp_1'+ 'previous_in-
dex_temp_2'+}
```

## Site directv.com, a multiday indexpage example

```
* WebGrab+Plus ini for grabbing EPG data from TvGuide websites
* Site : directv.com - USA and N /S - America
* revision : 0
* Jan van Straaten, 24 December 2010
* this site has a multiday index page for 5 days ---> maxdays must be 5.1
* needs WG++ 1.0.4 or higher
*
site {url=directv.com|timezone=UTC-04:00|maxdays=5.1|cultureinfo=en-US|charset=UTF-8}
site {titlematchfactor=90|ratingsystem=MPAA|episodesystem=onscreen}
url_index{url|http://www.directv.com/entertainment/channel/details/|channel|?format=SD}
urldate.format {}
index_urlshow {url()|https://www.directv.com|<var title="debug">|href="|?format=|<var ti-
tle="data">}
index_showsplit.scrub {multi|<ul class="listings-data">|<dd class="tim|<dt></dt>|<div
style="display:none;">}
index_start.scrub {single(exclude="TIME")|e">|</dd>|<dd class="title">}
index_title.scrub {single()|<dd class="title">|</dd>|<dd class="action">}
*
title.scrub {single|<var title="programTitle">"|</var>|<div class=}
description.scrub {single|<h4>Summary</h4>|<p>|</p>|<h4>}
director.scrub {single(separator=",")|<span class="details-sub">Director:|</span>|</p>|</p>}
producer.scrub {single(separator=",")|<span class="details-sub">Executive Producers:|</span>|</
p>|</p>}
producer.scrub {single(separator=",")|<span class="details-sub">Producers:|</span>|</p>|</p>}
actor.scrub {single(separator=",")|<span class="details-sub">Cast:|</span>|</p>|</p>}
rating.scrub {multi|<span class="details-sub">Rated:|</span>|</p>|</p>}
category.scrub {single(separator=",")|<span class="details-sub">|</span>|</p>|</p>}
productiondate.scrub {single|<span class="details-sub">Released:|</span>|</p>|</p>}
starrating.scrub {single|<span id="star_rating"|star-rating-bg-|">|</span>}
*
* operations:
index_urlshow.modify {addend(notnull)|?format=SD}
description.modify {addstart(null)|no details}
category.modify {cleanup(null)}
actor.modify {cleanup}
producer.modify {cleanup}
rating.modify {cleanup}
starrating.modify {replace|-x|.5} * 3-x is used to indicate 3.5 ?
starrating.modify {addend(notnull)|/5}
```

## Site Film1.nl, a site that uses includeblock, indices and loop

```
* WebGrab+Plus ini for grabbing EPG data from TvGuide websites
* Site : film1.nl
* revision : 0 needs V1.0.8, includeblock, loop, indexof and remove with indices
* Jan van Straaten, September 2011
*
site {url=film1.nl|timezone=UTC+01:00|maxdays=7|cultureinfo=nl-NL|charset=ISO-8859-
1|titlematchfactor=90}
site {ratingsystem=Kijkwijzer|grabengine=wget}
*
* the url doesn't contain a channel value, all channels are present in the index pages, selection
is done with includeblock in showsplit
url_index{url(debug)|http://www.film1.nl/film_kijken/film1_programmagids/?day=|urldate}
```

```

*http://www.film1.nl/film_kijken/film1_programmagids/?day=donderdag
urldate.format {datestring|dddd}
*
index_variable_element.modify{addstart|'config_site_id'} * site_id = includeblock values (block
numbers to include)
index_showsplit.scrub {multi(includeblock='index_variable_element')|<div class="column">|<ul
class="time">|}
index_urlshow {url|http://www.film1.nl|h3><a href="|">|</h3>}
*
index_start.scrub {single|<li>|</li>|</li>}
index_title.scrub {single|<h3><a href=|>|<span>|</h3>}
index_productiondate.scrub {single|<h3><a href=|<span>(|</span>|</h3>}
*
title.scrub {single(separator=":" include=first)|<div class="main">|<span pro-
perty="v:itemreviewed">|</span>|</span>}
subtitle.scrub {single(separator=":" exclude=first)|<div class="main">|<span pro-
perty="v:itemreviewed">|</span>|</span>}
description.scrub {multi|<td class="tab-tbl-title">Genre:</td>|<p>|</p>|<div class="social-
networking">}
director.scrub {multi(exclude="<a href="")|<td class="tab-tbl-title">Regie:</td>|>|</a>|</tr>}
actor.scrub {multi(exclude="<a href="")|<td class="tab-tbl-title">Cast:</td>|>|</a>|</tr>}
rating.scrub {multi|<ul class="info">|<li>}
category.scrub {multi(exclude="<a href="")|<td class="tab-tbl-title">Genre:</td>|>|</a>|</tr>}
starrating.scrub {single|<td class="tab-tbl-title">Waardering:</td>|"v:average">|</span>|</span>}
*
* operations:
*
description.modify {remove|style="font-style: italic;"}
description.modify {remove|> }
description.modify {replace|\|| ### }
description.modify {cleanup}
description.modify {replace|</a| }
*
* operations loop, removes all links in description:
loop {'temp_1' not "-1" max=20}|4} * loops the next 4 lines, while temp_1 (indexof <a href) not -1
temp_1.modify {calculate(type=char format=F0)|'description' "<a href" @} * indexof <a href
temp_2.modify {calculate(type=char format=F0)|'description' "html" @} * indexof html
temp_2.modify {calculate(format=F0)|'temp_2' 'temp_1' - 6 +} * length of link
description.modify {remove('temp_1' not "-1" notnull type=char)|'description' 'temp_1' 'temp_2'}
* loop end
*
description.modify {replace| ### |. }
starrating.modify {addend(not "")|/10}
*
* convert rating pict to a simple value:
rating.modify {replace|leeftijd_al_small.gif|alle}
rating.modify {replace|leeftijd_6_small.gif|6+}
rating.modify {replace|leeftijd_9_small.gif|9+}
rating.modify {replace|leeftijd_12_small.gif|12+}
rating.modify {replace|leeftijd_16_small.gif|16+}
rating.modify {replace|geweld_small.gif|geweld}
rating.modify {replace|eng_small.gif|eng}
rating.modify {replace|groftaalgebruik_small.gif|grof}
rating.modify {replace|discriminatie_small.gif|discriminatie}
rating.modify {replace|sex_small.gif|sex}
rating.modify {replace|drugs_small.gif|drugs}

```

## APPENDIX D Supported Element names in a siteini file

Siteini name	Xml tv name (ref xml tv. dtd)	action specifier	obligatory	multiple xml tv entry	multiple scrub	remarks
url_index	-	(url) & modify	y	-	n	the url of the show index page
url_date	-	format	y	-	-	date format for url builder
subpage	-	format	n	-	-	subpage format for url builder
index_url_show	-	(url) & modify	y	-	n	the url of the show detail page
index_url_channel_logo	icon	(url) & modify	n	-	n	sub-element of channel
index_variable_element	-	scrub & modify	n	-	n	a variable in scrubstrings (see 4.5.3)
index_showsplit	-	scrub	y	-	y	splits the indexpage in shows
index_date	(part of) start & stop	scrub & modify	n *1	n	n	*1 alternative = today used to compose attributes start and stop of element programme
index_start	start	scrub & modify	y	n	n	
index_stop	stop	scrub & modify	n *2	n	n	*2 alternative=nextstart
index_duration	stop *3	scrub & modify	n	n	n	*3 =start + duration
index_title	(title) *	scrub & modify	y	n	n	* only if no title
index_subtitle	sub-title	scrub & modify	n	n	y	
index_description	desc	scrub & modify	n	n	y	
index_director	director *	scrub & modify	n	y	y	
index_actor	actor *	scrub & modify	n	y	y	
index_presenter	presenter *	scrub & modify	n	y	y	* sub-elements of element credits
index_writer	writer *	scrub & modify	n	y	y	
index_producer	producer *	scrub & modify	n	y	y	
index_composer	composer *	scrub & modify	n	y	y	
index_rating	value *	scrub & modify	n	y	y	* attribute of element rating
index_ratingicon	icon *	scrub & modify	n	y	y	* attribute of element rating
index_category	category	scrub & modify	n	y	y	
index_productondate	date	scrub & modify	n	n	y	
index_starrating	value *	scrub & modify	n	n	y	* sub-element of element star-rating
index_episode	episode-num	scrub & modify	n	n	y	
index_subtitles	subtitles *	scrub & modify	n	n	y	* 'boolean' type elements
index_premiere	premiere *	scrub & modify	n	n	y	no value, when 'true' listed like
index_previouslyshown	previously-shown *	scrub & modify	n	n	y	<subtitles/>
index_videoaspect	aspect *	scrub & modify	n	n	y	sub-element of video
index_videoquality	quality *	scrub & modify	n	n	y	
index_temp_1 to _4	-	scrub & modify	n	-	y	used to customize values(see 4.5.3)
index_site_channel	-	scrub & modify	n	-	y	to create a channel -
index_site_id	-	scrub & modify	n	-	y	list file
title	title	scrub & modify	y *	n	y	* (n if indexonly)
titleoriginal	title *	scrub & modify	n	n	y	* (other lang)
subtitle	sub-title	scrub & modify	n	n	y	
description	desc	scrub & modify	n	n	y	
director	director *	scrub & modify	n	y	y	
actor	actor *	scrub & modify	n	y	y	
presenter	presenter *	scrub & modify	n	y	y	* sub-elements of credits
writer	writer *	scrub & modify	n	y	y	
composer	composer *	scrub & modify	n	y	y	
producer	producer *	scrub & modify	n	y	y	
rating	value *	scrub & modify	n	y	y	* attribute of element rating
ratingicon	icon *	scrub & modify	n	y	y	* attribute of element rating
category	category	scrub & modify	n	y	y	
productondate	date	scrub & modify	n	n	y	
starrating	value *	scrub & modify	n	n	y	* sub-element of element star-rating
episode	episode-num	scrub & modify	n	n	y	
subtitles	subtitles *	scrub & modify	n	n	y	* 'boolean' type element
premiere	premiere *	scrub & modify	n	n	y	no value, when 'true' listed like
previouslyshown	previously-shown *	scrub & modify	n	n	y	<subtitles/>
videoaspect	aspect *	scrub & modify	n	n	y	* sub-elements of video
videoquality	quality *	scrub & modify	n	n	y	
temp_1 to _4	-	scrub & modify	n	-	y	used to customize values(see 4.5.3)
previous_start _stop _duration		Read-Only elements with the value of the previous scrub. (see 4.5.3)				
previous_index_temp_1 to _4		Cannot be scrubbed or modified.				
previous_temp_1 to -4		They can be usefull in operations of other elements. E.g. osnetwork.com.ini				