FeedReader

XML News Feed Plugin
For RainMeter

By Martin R. Green

Contents

Overview 5

RSS and XML 5

RSS vs. Atom 6

Using the Plugin 7

Other Child Measure Types 9

  Feed Language 10

  Item Language 10

  Feed Images 10

  Item Description 11

  Item Categories 11

  Item Publication Date 11

  Item Age 13

  Custom Measures 13

  Plugin Version 14

  Still More… 15

Improving the Plugin 15

License 15

About the Author 16

Version History 17

  Version 2.2.3 17

  Version 2.2.2 17

  Version 2.2.1 17

  Version 2.2.0 17

  Version 2.1.1 17

  Version 2.1.0 17

  Version 2.0.1 17

  Version 2.0.0 17

  Version 1.1.1 18

  Version 1.1.0 18

  Version 1.0.0 18

Appendix A: Measure and Variable Reference 19

Parent Measure 19
Child Measure Types 19
    DllVersion 19
    FeedCustom 19
    FeedImageLink 19
    FeedImageTitle 19
    FeedImageURL 20
    FeedLanguage 20
    FeedLink 20
    FeedTitle 20
    FeedType 20
    ItemAge 20
    ItemCategories 20
    ItemCount 20
    ItemCustom 21
    ItemDesc 21
    ItemLanguage 21
    ItemLastUpdateSortable 21
    ItemLink 21
    ItemPubDate 22
    ItemPubDateSortable 22
    ItemTitle 22
Variables 22
    Attribute 22
    CategoryFilter 23
    Channel 23
    Delimiter 23
    Encoding 23
    ExecuteAfter 23
    ExecuteBefore 23
    ForceOffset 23
    ItemAgeLimit 24

ItemNum 24
Language Filter 24
LocalTime 25
LogXML 25
MatchFilterCase 25
MatchWholeFilter 25
NameSpace 25
Parent 26
Path 26
SubSpecialChars 27
Text 27
Type 27
URL 28

Overview

Traditionally, Rainmeter RSS Feed skins have used the WebParser plugin to extract information for display from news feeds. While this works, there are several drawbacks:

- A knowledge of regular expression (regex) syntax is required. Regex coding is very powerful, but extremely arcane.
- Regex parsing is very CPU intensive.
- There is no way to parse an unknown number of items from a feed with a single regex expression. If the feed has less than the number of items you wrote the expression for, then the whole parse fails and you won't get any of the items back. For this reason currently available RSS skins typically only display ten items or less, choosing to support the lowest common denominator number of items commonly in a feed.
- The WebParser plugin does not support dynamic variables, requiring a child WebParser for every single item in the feed that you want to display, even if you only display a few at a time.

When I wrote the first version of my **multiFEED** skin I got around the unknown number of items problem by adding a child measure for every item I was interesting in and parsing the feed for just one item in each measure. That way the items that existed were extracted, and the items that weren't there just returned nothing. Since I wanted both the item title and its link I also needed two grandchild measures for each item. To keep things manageable I limited **multiFEED** to fifty items per feed, but even then this meant 151 measures for each of the ten feeds just to get the item titles and links. Two more measures for the feed title and link and **multiFEED** needed 153 measures per feed, or 1,530 WebParser measures when all ten feeds were activated. And this didn't even include all the other measures needed to overcome the lack of dynamic variable support in WebParser, and the other measures and meters for display.

Although **multiFEED** 1.0 worked I just wasn't happy with the performance. With all ten feeds activated **multiFEED** could take up to thirty seconds to parse all ten feeds, and because it had to run over 500 regex processes to do it the CPU utilization hit 100% for the whole twenty to thirty seconds. Something had to change.

RSS and XML

Other Rainmeter RSS skins use WebParser because it is there, and because up till now there hasn't been a better alternative. When I began to rewrite **multiFEED** for better performance I knew WebParser had to go. Using WebParser treats a feed as unpredictable HTML, as if it was a web page, but it's not. RSS feeds are XML, and XML is highly structured and easy to read, for both human and machine. In fact, the .NET framework that Rainmeter and its plugins are developed with have very fast and robust XML handling functions built right in. Using WebParser **multiFEED** 1.0 needed several seconds to process each feed,and hogged the CPU to do it. Switching to a new plugin using the .NET XML classes allows a feed to be completely loaded in a small fraction of a second. In fact, with the new plugin, the time needed to read a whole feed is dominated by the internet download time. Once the entire feed has been read from the feed site, getting it ready for the child measures to access the items is effectively instantaneous to a user. With the new plugin and a reasonably fast internet connection, **multiFEED** 2.0 can load an unlimited number of items from up to ten feeds in under one second!

RSS vs. Atom

RSS is the original news feed XML document standard. It was very loosely defined, so consequently, other than the absolute basics, much of the useful information about the feed may be found in changing, or sometimes almost random places in the XML document, or may not be included at all. Even if the data you want is where you expect in the document, it may not be formatted as you expect since data formatting is left almost entirely to the feed publisher's imagination. Over time, some conventions have been adopted by the feed publishing community, but many big content publishers change or add to the XML structure to suit their own needs, and many of the smaller ones just don't know what they are doing.

A later XML document type specification, the Atom feed format was designed to solve many of the perceived deficiencies with RSS. Although the publishers were still given free rein to add anything they wished to the document contents, the core data used in nearly all feeds was relegated to clearly defined locations, and publisher specific information was supposed to be constrained in XML namespaces. Certain common feed data was decreed to be mandatory, and other slightly less popular information was deemed optional, but still clearly defined if it was present. Date formats in particular were tightened up, a very welcome improvement over RSS dates.

Although it is an undeniably better specification for news dissemination than RSS, the lead RSS had and its adoption by big content publishers has slowed the popular usage of Atom. All the big players (CNN, Fox News, etc.) provide RSS feeds, but most of the available Atom feeds are published by smaller, more "techie" industries and individuals. You often have to go specifically looking for Atom feeds, whereas RSS feeds are thrust at you on nearly every web site you visit.

Although they are both packaged in an XML document, and both contain some similar entities, such as items, the internal document structure is different, and different nomenclature for the same concepts is used. For instance, each channel/feed can have multiple articles, but for RSS we call this an "item" whereas the same thing in an Atom feed is an "entry".

In RSS almost everything of interest is found in basic XML nodes, whereas in an Atom feed much of what we want is located in an XML element attribute.

I chose to stick with the RSS terminology even when the actual source is an Atom document. For instance, whether the actual news article is an item (RSS) or an entity (Atom), they are always called "items" in **FeedReader**, even if they are actually "entries" under the skin.

Using the Plugin

Like WebParser, the **FeedReader** plugin implements the parent-child model. A parent measure reads the feed from the internet or a file, then child measures extract the feed and item information for display. A basic parent **FeedReader** measure looks like this:

```
[MeasureRSSParent]
Measure=Plugin
Plugin=FeedReader
UpdateDivider=<divider>
URL=<RSS URL>
```

That's all you need to read a full typical RSS or Atom feed into your Rainmeter skin. If required you can add clauses that tell a parent **FeedReader** measure to execute one or more bangs before and/or after downloading the feed:

```
ExecuteBefore=[!PreCommand1][!PreCommand2]...
ExecuteAfter=[!PostCommand1][!PostCommand2]...
```

We can restrict the parent measure to only include items that meet a filter criterion. See the `CategoryFilter`, `LanguageFilter` and `ItemAgeLimit` variable definitions in Appendix A for more information.

`ForceOffset=` provides a workaround for feeds that supply publication dates with unrecognized time zone abbreviations, which can be a problem with RSS. If supplied, this value overrides the time zone specified in the raw publication date from the feed and instead treats it as offset from GMT by this many hours. `ForceOffset` works for Atom feeds too, but is unlikely to be needed.

```
; ---Override the time zone from the feed
ForceOffset=-3.5
```

IMPORTANT: FeedReader version 1.x had this setting on each child measure that returned sortable dates or the item age. Since all dates in a feed would need same override if the date format was wrong, and because the `ItemAgeLimit` filtering requires the parent measure to know the offset too, this variable was moved to the parent measure for version 2.0.

In the rare case that an RSS XML document contains more than one channel, you can select the one you want with the `Channel` variable.

```
Channel=2
```

Finally, if you want special character codes automatically translated for you without need for a `Substitute` clause, you can add:

```
SubSpecialChars=1
```

…to the parent measure. See Appendix A & B for more information.

Now you can get the feed and item information from the parent using child measures:

```
; ---Get the channel name
[MeasureFeedTitle]
Measure=Plugin
Plugin=FeedReader
Parent=MeasureRSSParent
Type=FeedTitle

; ---Get the channel link
[MeasureFeedLink]
Measure=Plugin
Plugin=FeedReader
Parent=MeasureRSSParent
Type=FeedLink
```

The `Parent=` clause tells the child which parent measure to use. All **FeedReader** child measures also support a `Text=` clause that works similarly to the one for the String meter type (use “%1” in the clause string to include the original value of the measure).

Now let’s get basic information for a couple of items:

```
; ---Item 1 measures
[MeasureItem1Title]
Measure=Plugin
Plugin=FeedReader
Parent=MeasureRSSParent
Type=ItemTitle
ItemNum=1

[MeasureItem1Link]
Measure=Plugin
Plugin=FeedReader
Parent=MeasureRSSParent
Type=ItemLink
ItemNum=1

; ---Item 2 measures
[MeasureItem2Title]
Measure=Plugin
Plugin=FeedReader
Parent=MeasureRSSParent
Type=ItemTitle
ItemNum=2
```

```
[MeasureItem2Link]
Measure=Plugin
Plugin=FeedReader
Parent=MeasureRSSParent
Type=ItemLink
ItemNum=2
```

Even though we have explicitly set the `ItemNum` value for each of these measures, the **FeedReader** plugin fully supports dynamic variables, so we could have done something like this instead:

```
[MeasureItem1Title]
Measure=Plugin
Plugin=FeedReader
Parent=MeasureRSSParent
Type=ItemTitle
DynamicVariables=1
ItemNum=[MeasureFirstItem]
```

With `DynamicVariables=1` the `Parent=` clause is also dynamic, so you can also switch any child measure between multiple parent feeds at runtime. These two features hugely simplified feed and item page switching in **multiFEED**.

Other Child Measure Types

In addition to providing title and link values for the feed and items, **FeedReader** child measures can supply a lot of other information about a feed too. There are two ways to determine how many items **FeedReader** loaded from the feed (subject to any applied filters). First, the parent **FeedReader** measure returns this value, so you can use it directly in your formulas, or indirectly with square brackets as an argument to another measure or meter. Alternately you can use a dedicated child measure to get this value:

```
; ---Get the number of feed items
[MeasureItemCount]
Measure=Plugin
Plugin=FeedReader
Parent=MeasureRSSParent
Type=ItemCount
```

Since the item count is available directly from the parent measure, this child measure type is of limited usefulness, but it is included for completeness, in case someone finds their skin needs to get the value this way. If getting the item count from the parent measure suits your needs I recommend using that method instead.

Feed Language

```
; ---Get the feed language
[MeasureFeedLanguage]
Measure=Plugin
Plugin=FeedReader
Parent=MeasureRSSParent
Type=FeedLanguage
```

Item Language

```
; ---Get the language of item 3
[MeasureFeedLanguage]
Measure=Plugin
Plugin=FeedReader
Parent=MeasureRSSParent
Type=ItemLanguage
ItemNum=3
```

Feed Images

Some RSS feeds include a feed image. In most cases, if present, this image is typically a small logo for the web site. Most of the big content provider feed (CNN, Fox News… etc.) include one. Many of the smaller feed providers don't. If you want access to the feed image and related values, there are three child types to do that:

```
; ---Get the title of the feed image
[MeasureFeedImageTitle]
Measure=Plugin
Plugin=FeedReader
Parent=MeasureRSSParent
Type=FeedImageTitle

; ---Get the URL of the feed image
[MeasureFeedImageURL]
Measure=Plugin
Plugin=FeedReader
Parent=MeasureRSSParent
Type=FeedImageURL
```

```
; ---Get the link associated with the feed image
[MeasureFeedImageLink]
Measure=Plugin
Plugin=FeedReader
Parent=MeasureRSSParent
Type=FeedImageLink
```

If there is no image information in the feed, then these three measures will return an empty string.

Item Description

```
; ---Get the Item "n" Description
[MeasureItemnDescription]
Measure=Plugin
Plugin=FeedReader
Parent=MeasureRSSParent
Type=ItemDesc
ItemNum=n
```

For RSS feeds this is the value of the **`<description>`** node, whereas with Atom feeds it comes from the **`<content>`** node. In either case this is generally an HTML summary of the article. If there is no description/content for the item then this measure will return an empty string.

Item Categories

Many feeds categorize items with one or more category key words or phrases. The **FeedReader** plugin can provide them as a delimited string value.

```
; ---Get the Item "n" Categories
[MeasureItemnCategories]
Measure=Plugin
Plugin=FeedReader
Parent=MeasureRSSParent
Type=ItemCategories
ItemNum=n
```

The default delimiter is "|", but this can be changed if needed:

```
Delimiter=/
```

Item Publication Date

The RSS standard includes a publication date value for each feed item (pubDate), and Atom feeds may provide a publication date, but are at least required to supply a "last updated" date for each item. **FeedReader** can give you the publication date in several ways:

```
; ---Get the Item "n" Pub Date exactly as supplied in feed
[MeasureItemnPubDate]
Measure=Plugin
Plugin=FeedReader
Parent=MeasureRSSParent
Type=ItemPubDate
ItemNum=n
```

The `ItemPubDate` type gives you the date exactly as supplied in the feed, which is a pseudo RFC1123 format. eg:

```
Fri, 21 Dec 2012 23:38:04 GMT
```

I say "pseudo" because RFC1123 requires the time zone to always be "GMT", but many feeds supply the publication date in other time zones, such as EST or PST. Converting these dates to GMT can be very difficult since there is no standardized naming for time zone abbreviations, and many duplications of abbreviations for completely different time offsets. Many abbreviations are used to specify regionally recognized zones, and two zones with the same abbreviation can be many hours apart.

The second way to get the publication date from **FeedReader** is with the `ItemPubDateSortable` child type:

```
; ---Get the Item "n" Universal Sortable publication date
[MeasureItemnPubDateSortable]
Measure=Plugin
Plugin=FeedReader
Parent=MeasureRSSParent
Type=ItemPubDateSortable
ItemNum=n
```

This provides the date in a standardized sortable format:

```
2012-04-10 13:30:00Z
```

The "Z" at the end signifies that this date and time are UTC (GMT). In formulas this child measure type returns the UTC seconds since Jan 1, 0001. Remember what I said earlier about many feeds "cheating" with the RFC1123 date format? Well that sloppiness rears its ugly head here. Since there is no reliable way to convert a "sloppy" RFC1123 date to UTC, **FeedReader** has to do the best it can to make sure the date is converted to UTC if needed. The common North American times zone abbreviations are handled, as are email style UTC offsets:

```
Fri, 21 Dec 2012 23:38:04 +0530
```

…but any other time zone abbreviation is ignored, the date is treated as GMT, and an error message is written to the Rainmeter log (unless `ForceOffset` is set on the parent measure).

The **`ItemPubDateSortable`** child type takes an optional modifier:

```
LocalTime=1
```

If **`LocalTime=1`** then this measure returns the date adjusted for the local time zone offset of the user's PC. Since this date is no longer UTC, the returned string will have a trialing time from UTC offset instead of "Z". Also, in formulas, with **`ForceOffset`** set this child measure type will return the number of seconds since Jan 1, 0001, **adjusted for the user's local time zone**.

Item Age

This **FeedReader** child measure type provides the number of seconds since the item was published (or last updated for Atom feeds):

```
; ---Get the age in seconds of item "n"
[MeasureItemnAge]
Measure=Plugin
Plugin=FeedReader
Parent=MeasureRSSParent
Type=ItemAge
ItemNum=n
```

The user's PC clock is compared to the publication date. Note that accurate results depend on correctly converting the raw publication date from the feed to UTC, so use **`ForceOffset=`** with this child type if this is in doubt.

Custom Measures

I have tried to provide as much commonly used feed data as possible from **FeedReader**, but some feeds have additional, non-standard data along with the normal stuff, and other information is used too rarely to dedicate a child measure type to it. If you know where in the RSS or Atom XML for the feed the data you want is located, you can use the two "custom" child measure types to get to it.

Using these measure types requires some knowledge of XML path notation and the internal structure of RSS and Atom XML documents. I won't go into detail about this here. If you don't understand the rest of this section you probably don't need these child types.

The path you supply in the **`Path`** variable on the custom measure types should be relative to the feed/channel root node for the **`FeedCustom`** child type, and relative to the item root node for the **`ItemCustom`** type. For RSS feeds the XML channel root node is at "/rss/channel", while in at Atom feed the feed root is at "/feed". On the case of item root nodes, for RSS they are at "/rss/channel/item" and for Atom they are located at "/feed/entry".

```
[MeasureFeedCustom]
Measure=Plugin
Plugin=FeedReader
Parent=MeasureRSSParent
Type=FeedCustom
Path=lastBuildDate
```

This will get the last build date value for an RSS feed if it exists since this value's fully qualified path is **`/rss/channel/lastBuildDate`**. If there had been multiple **`/rss/channel/lastBuildDate`** nodes they would have been returned as a single string with delimiters between them. Again, the default delimiter is "|", but that can be changed with the **`Delimiter=`** modifier. You can get deeper in the path if you need to, so:

```
Path=image/title
```

…will return the image title for an RSS feed, if it exists, just like the **`FeedImageTitle`** child measure type does.

Getting data from the individual items works the same way, except you use the **`ItemCustom`** child type, and specify the item you are interested in:

```
[MeasureItemnCustom]
Measure=Plugin
Plugin=FeedReader
Parent=MeasureRSSParent
Type=ItemCustom
ItemNum=n
Path=guid
```

Once again, if there are multiple nodes at the specified path level, they are returned in a delimited string.

In Atom feeds, many of the values we are interested in are not the main value at the nodes, but instead they are attributes of the "Element" nodes. Some RSS feeds have publisher specific extensions that also use element attributes. See **`Attribute=`** in Appendix A for more details.

All data in an Atom feed must be accessed with an XML namespace prefix preceeding every level in the XPath. Some RSS feeds also have some nodes in a namespace. Trying to get the author name for an Atom item with a path of "author/name" will fail, and must be altered to "atom:author/atom:name". See the **`NameSpace=`** reference in Appendix A for details.

Plugin Version

If you are building a professional quality skin and want to include an "About" box popup, you may wish to include the **FeedReader** plugin version number in your text. There is another child type for that.

```
[MeasureFeedReaderVersion]
Measure=Plugin
Plugin=FeedReader
Parent=MeasureRSSParent
Type=DllVersion
```

Only the string value of this child type is meaningful. The numerical value is always zero.

Still More…

A full reference to these, and other child measure types can be found in Appendix A at the end of this document.

Improving the Plugin

I wrote this plugin to solve specific issues I had with using WebParser to process RSS feeds. Also, I didn't include dedicated child measure types for feed nodes I deemed to be of less significance. As such, I may not have anticipated all the data in your favourite RSS or Atom feed that you want access to. Custom child measure types should provide most things you might need to extract from a feed, but if you have a request for more functionality, or heaven forbid, a bug report, please contact me at LimeyCanuck1@gmail.com.

License

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivs 3.0 Unported License (*http://tinyurl.com/9njqju7*). This means you can use and distribute the unmodified plugin with your own skins, but you can't use it to make money or as part of a commercial product. If you do distribute it with your own skins you must give me credit as the developer of the plugin. You may also distribute the unmodified plugin as you wish, but if you do please share the full package including this documentation, not just the DLLs alone.

Contact me at LimeyCanuck1@gmail.com if you wish to make special licensing arrangements.

About the Author

I've been an independent contract software developer since 1988, working with a wide variety of tools and languages in the Corporate IT world. Prior to that I spent a decade working with high-tech marine navigational hardware and software. Although much of my IT work has used Powerbuilder in a client-server environment, I am also fluent in C, C++, and C#, JavaScript, and QML, among other languages. I am now developing primarily for mobile devices, particularly the terrific new BlackBerry 10 phones. I discovered Rainmeter in 2012 after being forced to leave a few beloved Windows desktop gadgets by the wayside when Microsoft deprecated the Windows Sidebar/Gadget engine.

My deviantART handle is LimeyCanuck (*http://LimeyCanuck.deviantart.com*).

I can be contacted at LimeyCanuck1@gmail.com, and I live in Whitby, Ontario, Canada with my family and four cats.

Martin Green

Version History

Version 2.2.3

- Added **`Encoding`** setting to parent measure.
- Added parent measure **`LogXML`** setting to parent measure.

Version 2.2.2

- In the extremely rare case that an RSS feed's XML did not include publication dates for the items (supposedly mandatory) **FeedReader** would fail to load those items at all. With version 2.2.2 when this happens the publication date is assumed to be now and the item is correctly loaded.

Version 2.2.1

- Version 2.2.0 was interpreting feeds as having UTF-16 encoding, whereas feeds are encoded in UTF-8. This made many special characters display incorrectly.

Version 2.2.0

- Dramatically increased the number of special HTML character codes handled by **FeedReader**. Some foreign language feeds were not loading properly with earlier versions.
- Default setting for **`SubSpecialChars`** changed from off to on.

Version 2.1.1

- Inadvertently uploaded test version of 2.1.0 with special character translation disabled for RSS feeds instead of final release version.
- The **`ItemDesc`** child measure type now returns the item **`<content>`** nodevalue for Atom feeds instead of nothing as in earlier versions.

Version 2.1.0

- Added automatic translation of special characters, if desired.
- For Atom feeds, **`ItemCategories`** child measures were returning a list of categories for ALL items, not just the one specified. Fixed.
- Custom child types were not handling Atom feed XPaths properly. Fixed.
- Custom child types were ignoring **`Attribute=`**. Fixed.

Version 2.0.1

- Bug fix – **FeedReader** was choking on RSS feeds with no language defined again.

Version 2.0.0

- Atom format news feed support!
- Multiple channel support. The RSS XML format allows more than one feed (channel in RSS terminology) in the XML document. Very rare, but version 2.0 now supports this.

- The **`FeedCustom`** and **`ItemCustom`** child measure types now fully support XML namespaces and XML element attributes.
- **`Parser=`** child measure variable changed to **`Parent=`**. Old syntax retained to avoid breaking existing skins using **FeedReader** 1.0, but use the new **`Parent`** variable for new designs.
- Feeds items can now be filtered so that only ones matching specified criteria are made available from the XML document.
- **`ForceOffset`** variable moved from child measures to the parent measure, which makes more sense and allows items filtering on the item age. Sorry for the inconvenience, but please modify your skins if you are using this variable and want to upgrade them to **FeedReader** 2.0.
- Added **`FeedType`**, **`FeedLanguage`** and **`ItemLanguage`** child measure types.

Version 1.1.1

Fixed a plugin crash when RSS feed didn't have a node for feed language.

Version 1.1.0

Lots of new features added in response to user comments about version 1.0.0:

- Forgot to document **`ExecuteBefore=`** and **`ExecuteAfter=`** on the parent measure. This functionality existed in version 1.0.0, but I forgot to tell anyone about it.
- Added **`FeedLanguage`** child measure type.
- Added **`FeedImageTitle`**, **`FeedImageURL`**, and **`FeedImageLink`** child measure types.
- Added **`ItemCategories`** child measure type.
- Added **`ItemPubDate`** and **`ItemPubDateSortable`** child measure types.
- Added **`ItemAge`** child measure type.
- Added **`FeedCustom`** and **`ItemCustom`** child measure types.

Version 1.0.0

Initial public release.

Appendix A: Measure and Variable Reference

Parent Measure

At minimum, a **FeedReader** parent measure has the **`URL=`** set to the feed XML document location and does not have **`Parent=`** defined. When updated it returns the number of items found in the feed/channel that passed all specified filter criteria, if any.

Parent measure behaviour can be modified with:

```
CategoryFilter=
Channel=
Delimiter=
Encoding=
ChilExecuteAfter=
ExecuteBefore=
ForceOffset=
ItemAgeLimit=
LanguageFilter=
LogXML=
MatchFilterCase=
MatchWholeFilter=
SubSpecialChars=
```

Child Measure Types

The child measure type is set with the **`Type=`** variable.

DllVersion

Returns the version string from the **FeedReader** DLL.

Variables: **`Text`**

FeedCustom

Allows custom access to the data in the feed/channel XML node or element attributes.

See the **`Path`**, **`Attribute`**, and **`Namespace`** variable descriptions for details.

Variables: **`Text`**, **`Delimiter`**, **`Path`**, **`Attribute`**, **`Namespace`**

FeedImageLink

Return the feed image link for the feed/channel, if defined. Only applies to RSS feeds.

Variables: **`Text`**

FeedImageTitle

Return the title of the feed image for the feed/channel, if defined. Only applies to RSS feeds.

Variables: **`Text`**

FeedImageURL

Return the address of the feed image itself for the feed/channel, if defined. Only applies to RSS feeds.

Returns the language for the feed/channel, if defined. Only applies to RSS feeds.

Variables: **`Text`**

FeedLanguage

Returns the feed language if recorded in the feed.

Variables: **`Text`**

FeedLink

Returns the link associated with the feed/channel. Sometimes this just points to the same location as the feed XML document, but usually it links to the parent website for the feed.

Variables: **`Text`**

FeedTitle

Returns the title of the feed/channel.

Variables: **`Text`**

FeedType

Returns the type of the feed. Possible string values are “Atom”, “RSS”, or “Unknown”. Numerical return values are 0 (Unknown), 1 (RSS), or 2 (Atom).

Variables: **`Text`**

ItemAge

Returns the time since publication or last update of the item in the feed/channel, in seconds. If the specified item number is out of range, returns -1.

Variables: **`Text`**, **`ItemNum`**

ItemCategories

Returns the categories attributed to the specified item in the feed/channel, in a delimited string.

If the specified item number is out of range, returns an empty string.

Variables: **`Text`**, **`ItemNum`**, **`Delimiter`**

ItemCount

Returns the number of items in the feed/channel. If a filter was set for the parent feed, then returns the number of items that qualified.

Variables: **`Text`**

ItemCustom

Allows custom access to the data in the XML node or element for the specified item in the feed/channel. If the specified item number is out of range, returns an empty string.

See the **`Path`**, **`Attribute`**, and **`Namespace`** variable descriptions for details.

Variables: **`Text`**, **`ItemNum`**, **`Delimiter`**, **`Path`**, **`Attribute`**, **`Namespace`**

ItemDesc

Returns the description from the specified item in the feed/channel. This is often an HTML preview of the article. In Atom terminology this is the entry content. Returns an empty string if no description/content is defined for the item.

Variables: **`Text`**, **`ItemNum`**

ItemLanguage

Returns the language of the specified item in the feed/channel if recorded in the item.

If the specified item number is out of range, returns an empty string.

Variables: **`Text`**, **`ItemNum`**

ItemLastUpdateSortable

Returns the last time the specified item in the feed/channel was updated. In Atom feeds, according to the specification this date **must** be defined in the feed, though in a few I've seen it is not present, but the optional "published" date is instead. If so, the published date is returned in place of the missing last update date.

RSS feeds don't include a last updated date, so for RSS the publication date is returned instead.

The date is returned in a standardized "sortable" format. (ie. "**`2012-04-10 13:30:00Z`**")

If the specified item number is out of range, returns an empty string.

Variables: **`Text`**, **`ItemNum`**, **`LocalTime`**

ItemLink

Returns the link for the specified item in the feed/channel.

If the specified item number is out of range, returns an empty string.

Variables: **`Text`**, **`ItemNum`**

ItemPubDate

Returns the date published for the specified item in the feed/channel in ***exactly*** the format it is found in the feed. Since RSS feeds often supply this date in a non-standard format it may not be very usable. In the extremely rare (and illegal) case that no publication date is present in the XML for the item, the return value is an empty string.

Atom feeds ***may*** include a published date for the item, but the Atom specification says this is optional, and only the last date updated is required. In this case this child measure type will return the last updated date instead.

If the specified item number is out of range, returns an empty string.

Variables: **`Text`**, **`ItemNum`**

ItemPubDateSortable

Returns the date published for the specified item in the feed in a Universal Sortable format. Since RSS feeds often supply this date in a non-standard format the plugin may have trouble converting the raw date string in the feed XML into the correct date and time. If the time zone in the raw date is misinterpreted, using the **`ForceOffset`** variable may help. In the extremely rare (and illegal) case that no publication date is present in the XML for the item, the return value will be the current date and time when the item was read from the XML.

Atom feeds ***may*** include a published date for the item, but the Atom specification says this is optional, and only the last date updated is required. In this case this child measure type will return the last updated date instead.

The date is returned in a standardized "sortable" format. (ie. "**`2012-04-10 13:30:00Z`**")

If the specified item number is out of range, returns an empty string.

Variables: **`Text`**, **`ItemNum`**, **`LocalTime`**

ItemTitle

Returns the title for the specified item in the feed/channel.

If the specified item number is out of range, returns an empty string.

Variables: **`Text`**, **`ItemNum`**

Variables

Attribute

Tells the custom child type measures to get their value from the specified element attribute rather than the node inner text. If the attribute is not defined at the specified path location then the child measure returns an empty string.

CategoryFilter

The list of item categories to filter the items by. The list should be delimited by the delimiter character or string defined for the measure. The delimiter can be changed from the default of "|" using the `Delimiter` variable described below. If `CategoryFilter` is defined then only items with a category found in the filter list will be retrieved from the feed.

```
CategoryFilter=XML|RSS|Atom
```

Channel

Although very rare, RSS XML documents may contain more than one feed. RSS calls these channels. The default is channel one, but setting this variable allows access to other channels.

```
Channel=2
```

Delimiter

Many **FeedReader** measure types use a delimiter to separate input or output values. The default delimiter for all of them is "|", but if this clashes with something you need to use in the input or output string you can change the delimiter used with the measure by setting this variable.

```
Delimiter=++
```

Encoding

The XML for most feeds uses UTF-8 encoding, but there are some exceptions. If you read XML using the wrong encoding many special characters will be garbled, so this setting allows you to override the default encoding assumed by the parent measure (UTF-8). If you are encountering strange characters when you read your feed you can set `LogXML=1` to dump the raw XML to the Rainmeter log file then check the encoding defined in the first line of the XML. Use that value to set the parent measure `Encoding` parameter:

```
Encoding=ISO-8859-1
```

ExecuteAfter

A **FeedReader** parent measure can execute one or more Rainmeter bangs after it has loaded the XML from the URL.

```
ExecuteAfter=[!PostCommand1][!PostCommand2]...
```

ExecuteBefore

A **FeedReader** parent measure can execute one or more Rainmeter bangs before it loads the XML from the URL.

```
ExecuteBefore=[!PreCommand1][!PreCommand2]...
```

ForceOffset

When **FeedReader** reads dates from the feed XML it converts the date string according to RSS and Atom specified date formats. While Atom feeds reliably use the correct date format, RSS feeds are notorious for not following the "rules". All RSS dates are supposed to be a slightly modified RFC1123 format, which should always be UTC date/times with a "GMT" suffix. Unfortunately, many RSS feed publishers don't follow the rules, and supply dates with a different time zone (TZ) suffix, either an abbreviation like EST or PST, or an email style offset from GMT. **FeedReader** correctly handles North American TZ abbreviations and email style time offsets, but TZ abbreviations for the rest of the world are so non-standardized that if **FeedReader** encounters other unrecognized abbreviations if will just treat the date as UTC, which is probably wrong. This measure variable allow you to explicitly set the time offset in hours from GMT/UTC. If you do this, **FeedReader** will ignore the TZ suffix in the date from the feed, even if it was a valid one.

```
; ---Feed date/times are Nepal local time
TimeOffset=+5.75
```

ItemAgeLimit

A feed item has a date and time associated with it, either the original publication or the last update. An item's age is the time between that date and now. You may only wish to display the more recent items in a feed, and you can do that by setting this variable on a parent **FeedReader** measure. Ages are measured in seconds since publication/last update.

```
; ---Only display items from the past 30 minutes
ItemAgeLimit=1800
```

Note that ages are calculated only on a parent measure update. No matter how many times your child measures read the parent measure between updates, the "now" time used to calculate the item ages will always be the time of the last parent measure update, so the age won't change till the parent is updated again.

IMPORTANT: accurate age determination depends on **FeedReader** being able to correctly convert the published/updated date string to UTC, so if the XML for your feeds doesn't format dates correctly, be sure to use the `ForceOffset=` variable on the parent to compensate.

ItemNum

All the item child measure types use this variable to specify which item from the feed/channel the measure wants to work with.

```
ItemNum=3
```

Language Filter

A list of languages to filter the items by. The list should be delimited by the delimiter character or string defined for the measure. The delimiter can be changed from the default of "|" using the Delimiter variable described below. If `LanguageFilter` is defined then only items published in the languages included in the filter list will be retrieved from the feed.

```
; ---Only include items written in British or US English
```

```
LanguageFilter=en-gb|en-US
```

You need to know the format of the language specifiers in the XML document for this to work.

LocalTime

Any child measure that returns a “sortable” date defaults to UTC (GMT) date/times. If you set this variable to 1 the returned date will be converted to the local PC time instead.

```
LocalTime=1
```

LogXML

Turning this on will make the parent measure output the raw feed XML to the Rainmeter log file. This can be useful for troubleshooting if you are having trouble with a feed either not loading at all or not displaying properly.

```
LogXML=1
```

MatchFilterCase

Default behaviour for any child measure filtering variables is to ignore the case of the filter terms. Setting `CategoryFilter=”XML”` for instance will only include items that contain “XML”, “xml”, “Xml”… somewhere in the list of categories the item is flagged with. Override this behaviour by setting `MatchFilterCase=1`.

```
;---Only include items with “XML” in category, not “Xml” or “xml”
MatchFilterCase=1
```

MatchWholeFilter

Default behaviour for any child measure filtering variables is to include the item if at least one of the filter terms is found anywhere in the item attribute. So, for instance, category filtering on “XML” would match items with a category of “XML”, but also the categories “Learning XML” or “XML fundamentals”. Override this behaviour by setting `MatchWholeFilter=1`.

```
; ---Only include items with EXACT category “XML”, not “Learning XML”
MatchWholeFilter=1
```

NameSpace

The “custom” child types allow access to any node or element attribute in the XML feed document. Some of these nodes, elements, or attributes may use XML namespace prefixes (such as “`atom10:`”). To read anything from the XML with a prefix, the namespaces used must be defined to the child measure. This is not a tutorial on XML namespaces, but each definition must specify the namespace prefix and the URI. This is done by supplying both values to the child with the NameSpace variable, separating the values with the delimiter the child is using (set this with ”`Delimiter=`“ if needed).

```
; ---Define the “dc” namespace
NameSpace=”dc|http://purl.org/dc/elements/1.1/”
```

Sometimes more than one namespace is used to get to a value in the XML that you are interested in. If so, you can define additional namespaces to the child measure like this:

```
NameSpace2="itunes|http://www.itunes.com/dtds/podcast-1.0.dtd"
NameSpace3="taxo|http://purl.org/rss/1.0/modules/taxonomy/"
…etc.
```

Failure to define a namespace before using the prefix for it will result in **FeedReader** choking on the node, element, or attribute name. Every custom child measure needs to have the namespaces it uses defined individually. You can, of course, use global variables for this if they all require the same namespace definitions.

Atom feeds redefine the default namespace, and the .NET XML library doesn't handle this the way you'd expect. You can reset the default namespace the **FeedReader** plugin uses for a feed by registering a namespace with no prefix (ie. "|URI"), but the .NET libraries still won't find the nodes you specify with **`PATH=`**. The standard workaround for this is to declare a new namespace prefix for XML documents that redefine the default and then use that in all your Xpath expressions. For Atom feeds, FeedReader will automatically register an "**`atom:`**" prefix for the new default URI (read from the XML) that will give access to all the elements and attributes you want. The only caveat is that you must prefix every node in your XPath with "**`atom:`**". Note that this problem exists for any XML feed document that redefines the default namespace, so even if your feed is not Atom, you may still have to deal with it. In this case you would need to register your own namespace prefix/URI pair to match the default for the feed, and then reference all nodes with that prefix.

NOTE: The **`xml`** and **`xmlns`** namespaces are automatically defined for all **`FeedCustom`** and **`ItemCustom`** child measures. You do not have to explicitly register them again.

Parent

FeedReader child measures need to know the parent that is parsing the feed. This variable defines this. Only use this variable with a child measure, not the parent.

```
Parent=ParentMeasure
```

NOTE: In **FeedReader** 1.0 versions, the parent was set with **`Parser=`** instead. To avoid breaking existing skins, that syntax is a synonym for **`Parent=`**, but use **`Parent=`** in new skins.

Path

The custom child measure types allow access to nearly any information in an XML feed document by specifying the XML path to the data. For **`FeedCustom`** the path is relative to the RSS channel root node (specified with **`Channel=`**) "/rss/channel" or the Atom root node ("/feed") , while for **`ItemCustom`** it is relative to the item root node (specified with **`ItemNum=`**) which is "/rss/channel/item" for RSS and "/feed/entry" for Atom.

SubSpecialChars

IMPORTANT: Functionality and usage of this setting has changed since versions prior to 2.2.0.

Generally, when a skin uses WebParser to extract news feed data, the developer has to deal with special character codes, such as `&amp;` or `&#229;`, which normally requires a complicated `Subsititute=` measure clause. There are so many possible character codes you might encounter that most skin designers choose to only substitute for the most common ones. You can still do this with **FeedReader** if desired, but you will probably want **FeedReader** to do the hard work for you. Setting `SubSpecialChars=1` makes the parent **FeedReader** measure translate all special character codes before the child measure even sees them.

IMPORTANT: Turning this setting off may speed up feed loading since the translations take a small amount of time to perform, but if you do so **FeedReader** may not be able to load all your feeds. Some special characters are not properly handled by the programming framework used by **FeedReader** (Microsoft .NET) and any feeds containing them will confuse the feed loader. Turning on this setting will translate those unrecognized special characters BEFORE they are loaded.

In versions of **FeedReader** prior to 2.2.0 this setting was off by default, but as of 2.2.0 the default is on.

Text

All **FeedReader** measure types allow you to modify the string value result with the `Text=` variable. Like the standard Rainmeter measures, you can insert the default string value of a measure into the modified string value with "%1".

```
; ---This measure returns "The plugin version is 2.0.0"
[MeasurePluginVersion]
Measure=Plugin
Plugin=FeedReader
Parent=MeasureFeed1
Type=DllVersion
Text="The plugin version is %1"
```

Since you can do the same thing with your String meters this feature may be of limited use, but may be handy for use with other meter types. If you need more sophisticated result modification than `Text=` can provide you should probably be using `Substitute=` and `RegExpSubstitute=` clauses instead.

Type

The child type of the measure.

```
Type=FeedTitle
```

Valid child types are:

```
DllVersion
FeedCustom
FeedImageLink
FeedImageTitle
FeedImageURL
FeedLanguage
FeedLink
FeedTitle
FeedType
ItemAge
ItemCategories
ItemCount
ItemCustom
ItemDesc
ItemLastUpdateSortable
ItemLink
ItemPubDate
ItemPubDateSortable
ItemTitle
```

URL

This is the location of the XML feed document:

```
URL="http://feeds.engadget.com/weblogsinc/engadget"
```

Despite the name of this variable being URL, in fact you can specify a local file as the feed XML source:

```
URL="C:\feed.xml"
```

This can be handy when testing code using the `FeedCustom` and `ItemCustom` measure types since you can tweak a local copy of the feed XML and have **FeedReader** parse that instead of the document from the web.