

EiffelRSS

SYNDICATION Developer Guide

Michael Käser <kaeserm@student.ethz.ch>

Martin Luder <luderm@student.ethz.ch>

Thomas Weibel <weibelt@student.ethz.ch>

Abstract

SYNDICATION is the main cluster of EiffelRSS with a feed object model and classes to load / write feeds. It is divided into three subclusters.

Contents

I	INTERFACE	1
1	Overview	2
2	Class SYNDICATION_FACTORY	3
2.1	Overview	3
2.2	Usage	3
2.3	Features	4
2.3.1	READER factory	4
2.3.2	WRITER factory	4
2.3.3	FEED_MANAGER factory	4
2.3.4	FEED factory	5
2.3.5	CHANNEL factory	5
2.3.6	ITEM factory	6
2.3.7	CATEGORY factory	7
3	Class FEED_MANAGER	8
3.1	Overview	8
3.2	Usage	8
3.3	Features	10
3.3.1	Initialization	10
3.3.2	Access	10
3.3.3	Setter	11
3.3.4	Element change	11
3.3.5	Refresh	11
3.3.6	Conversion	12
3.3.7	Conversion (sort)	12

4	Class FEED_READER	14
4.1	Overview	14
4.2	Usage	14
4.3	Features	15
4.3.1	Initialization	15
4.3.2	Basic operations	15
5	Class FEED_WRITER	16
5.1	Overview	16
5.2	Usage	16
5.3	Features	17
5.3.1	Initialization	17
5.3.2	Basic operations	17
II	FEED	18
6	Overview	19
7	Usage	20
8	Class FEED	22
8.1	Overview	22
8.2	Features	22
8.2.1	Initialization	22
8.2.2	Access	22
8.2.3	Setter	23
8.2.4	Status	23
8.2.5	Basic operations	24
8.2.6	Debug	25
9	Class CHANNEL	26
9.1	Overview	26
9.2	Features	26
9.2.1	Initialization	26
9.2.2	Access	26

9.2.3	Access (RSS 0.91)	29
9.2.4	Access (RSS 1.0)	29
9.2.5	Access (categories)	29
9.2.6	Access (observers)	30
9.2.7	Access (metadata)	30
9.2.8	Setter	30
9.2.9	Setter (RSS 0.91)	33
9.2.10	Setter (RSS 1.0)	33
9.2.11	Setter (categories)	33
9.2.12	Setter (observers)	33
9.2.13	Status	33
9.2.14	Status (RSS 0.91)	36
9.2.15	Status (RSS 1.0)	36
9.2.16	Status (categories)	36
9.2.17	Status (observers)	36
9.2.18	Status (metadata)	36
9.2.19	Basic operations	37
9.2.20	Basic operations (RSS 1.0)	38
9.2.21	Basic operations (categories)	38
9.2.22	Basic operations (observers)	38
9.2.23	Sort	39
9.2.24	Sort (categories)	39
9.2.25	Debug	40
9.3	Subclass CHANNEL_CLOUD	40
9.3.1	Initialization	40
9.3.2	Access	40
9.3.3	Setter	41
9.3.4	Debug	42
9.4	Subclass CHANNEL_IMAGE	42
9.4.1	Initialization	42
9.4.2	Constants	42
9.4.3	Access	43
9.4.4	Setter	43

9.4.5	Status	44
9.4.6	Debug	45
9.5	Subclass CHANNEL_TEXT_INPUT	45
9.5.1	Initialization	45
9.5.2	Access	45
9.5.3	Setter	46
9.5.4	Debug	46
10	Class ITEM	47
10.1	Overview	47
10.2	Features	47
10.2.1	Initialization	47
10.2.2	Access	48
10.2.3	Access (categories)	49
10.2.4	Access (metadata)	49
10.2.5	Setter	50
10.2.6	Setter (categories)	51
10.2.7	Setter (metadata)	51
10.2.8	Status	52
10.2.9	Status (categories)	53
10.2.10	Basic operations (categories)	53
10.2.11	Sort (categories)	53
10.2.12	Debug	54
10.3	Subclass ITEM_ENCLOSURE	54
10.3.1	Initialization	54
10.3.2	Access	54
10.3.3	Setter	55
10.3.4	Debug	55
10.4	Subclass ITEM_GUID	56
10.4.1	Initialization	56
10.4.2	Access	56
10.4.3	Setter	56
10.4.4	Debug	57

10.5 Subclass ITEM_SOURCE	57
10.5.1 Initialization	57
10.5.2 Access	57
10.5.3 Setter	57
10.5.4 Debug	58
11 Class CATEGORY	59
11.1 Overview	59
11.2 Features	59
11.2.1 Initialization	59
11.2.2 Access	60
11.2.3 Setter	60
11.2.4 Status	60
11.2.5 Debug	60
12 Observers	61
12.1 Overview	61
12.2 Class OBSERVABLE_CHANNEL	61
12.2.1 Access	61
12.2.2 Setter	61
12.2.3 Status	62
12.2.4 Basic operations	62
12.3 Class CHANNEL_OBSERVER	62
12.3.1 Observer	62
12.4 Class SIMPLE_CHANNEL_OBSERVER	63
12.4.1 Access	63
12.4.2 Observer	63
 III FORMATS	 64
 13 Overview	 65

14 Management: Class <code>FORMAT_LIST</code>	66
14.1 Overview	66
14.2 Usage	66
14.3 Features	66
14.3.1 Initialization	66
14.3.2 Access	67
14.3.3 Detection	67
15 Format implementations	68
15.1 Adding a new format	68
15.2 Base classes	68
15.2.1 <code>FORMAT_DEF</code>	68
15.2.2 <code>READER_DEF</code>	69
15.2.3 <code>WRITER_DEF</code>	70
15.3 Built-in formats	70
15.3.1 RSS 2.0	70
15.3.2 Error	70

List of Figures

1.1	BON diagram of cluster INTERFACE	2
3.1	BON diagram of class FEED_MANAGER	8
4.1	BON diagram of class FEED_READER	14
6.1	BON diagram of cluster FEED	19
13.1	BON diagram of cluster FORMATS	65

Part I

INTERFACE

Chapter 1

Overview

INTERFACE is the sub-cluster of syndication with all the classes a developer needs to use the library. There are classes to read into and write from a FEED, a FEED_MANAGER to administrate a list of FEEDs, and a factory class which makes it easy to create all necessary objects.

See figure 1.1 for an overview of the cluster.

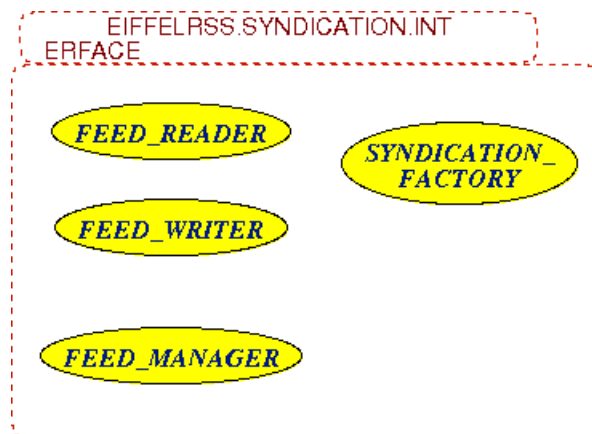


Figure 1.1: BON diagram of cluster INTERFACE

Chapter 2

Class SYNDICATION_FACTORY

2.1 Overview

SYNDICATION_FACTORY provides an easy way to create objects of classes from the cluster SYNDICATION.

2.2 Usage

```
class USAGE_EXAMPLE

create
  make

feature — Initialization

  make is
    — Creation procedure.
  do
    create syndication

    feed := syndication.new_feed ("EiffelRSS", create \
→ {HTTP_URL}.make ("http://eiffelrss.berlios.de/"), \
→ "EiffelRSS news")

    item := syndication.new_item (channel, "Version 23 \
→ released!", create {HTTP_URL}.make ("http://\
→ eiffelrss.berlios.de/Main/News"), "Version 23 of \
→ EiffelRSS got release today. Happy syndicating!")
```

```
        feed.add_item (item)
    end

feature — Arguments

    syndication: SYNDICATION_FACTORY
        — Syndication factory object

    feed: FEED
        — Feed object

    item: ITEM
        — Item object

end — class USAGE_EXAMPLE
```

2.3 Features

2.3.1 READER factory

new_reader_from_url

```
new_reader_from_url (a_url: STRING): FEED_READER
    — Create with 'a_url' as source of feed
```

2.3.2 WRITER factory

new_writer_from_feed

```
new_writer_from_feed (a_feed: FEED): FEED_WRITER
    — Create a writer object for the feed 'a_feed'
```

2.3.3 FEED_MANAGER factory

new_feed_manager

```
new_feed_manager: FEED_MANAGER
    — Create a new feed manager with default refresh \
    — period '30'
```

new_feed_manager_custom

```
new_feed_manager_custom (a_refresh_period: INTEGER): \
  FEED_MANAGER
  — Create a new feed manager with default refresh \
  →period 'a_refresh_period'
```

2.3.4 FEED factory**new_feed**

```
new_feed (a_title: STRING; a_link: URL; a_description: \
  STRING): FEED
  — Create a feed with title , link and description
```

new_feed_from_channel

```
new_feed_from_channel (a_channel: CHANNEL): FEED
  — Create a new feed from an existing channel
```

2.3.5 CHANNEL factory**new_channel**

```
new_channel (a_title: STRING; a_link: URL; a_description\
  →: STRING): CHANNEL
  — Create a channel with title , link and description
```

new_channel_cloud

```
new_channel_cloud (a_domain: STRING; a_port: INTEGER; \
  →a_path: STRING; a_register_procedure: STRING; \
  →a_protocol: STRING): CHANNEL_CLOUD
  — Create a channel cloud with domain , port , path , \
  →register procedure and protocol
```

new_channel_image

```
new_channel_image (a_url: URL; a_title: STRING; a_link: \
URL): CHANNEL_IMAGE
    — Create a channel image with URL, title , and link
```

new_channel_text_input

```
new_channel_text_input (a_title: STRING; a_description: \
STRING; a_name: STRING; a_link: URL): \
CHANNEL_TEXT_INPUT
    — Create a channel text input with title , description \
    →, name and link
```

2.3.6 ITEM factory**new_item**

```
new_item (a_channel: CHANNEL; a_title: STRING; a_link: \
URL; a_description: STRING): ITEM
    — Create an item with title , link and description
```

new_item_with_title

```
new_item_with_title (a_channel: CHANNEL; a_title: STRING\
→): ITEM
    — Create an item with title
```

new_item_with_description

```
new_item_with_description (a_channel: CHANNEL; \
→a_description: STRING): ITEM
    — Create an item with description
```

new_item_enclosure

```
new_item_enclosure (a_url: URL; a_length: INTEGER; \
→a_type: STRING): ITEM_ENCLOSURE
    — Create an item enclosure
```

new_item_guid

```
new_item_guid (a_guid: STRING): ITEM_GUID
  — Create an item guid with 'is_perma_link' set to \
  → False
```

new_item_guid_perma_link

```
new_item_guid_perma_link (a_guid: STRING): ITEM_GUID
  — Create an item guid with 'is_perma_link' set to \
  → True
```

new_item_source

```
new_item_source (a_name: STRING; a_url: URL): \
  → ITEM_SOURCE
  — Create an item source
```

2.3.7 CATEGORY factory**new_category**

```
new_category: CATEGORY
  — Create a category with title '[unnamed category]'
```

new_category_with_title

```
new_category_with_title (a_title: STRING): CATEGORY
  — Create a category with title 'a_title '
```

new_category_with_title_domain

```
new_category_with_title_domain (a_title: STRING; \
  → a_domain: URL): CATEGORY
  — Create a category with title 'a_title ' and domain '\
  → a_domain '
```

Chapter 3

Class FEED_MANAGER

3.1 Overview

FEED_MANAGER is a class to manage feeds. It provides features to add, remove and refresh feeds.

See figure 3.1 for an overview of the class.

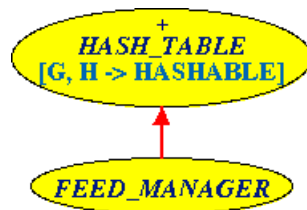


Figure 3.1: BON diagram of class FEED_MANAGER

3.2 Usage

```

class
  FEED_MANAGER_EXAMPLE

create
  make

feature — Initialization

  make is
    — Creation procedure.
  do
    — Create a simple feed
  
```

```

    create feed.make ("EiffelRSS", create {HTTP_URL}.
    --make ("http://eiffelrss.berlios.de"), "EiffelRSS
    --news")
    feed.set_refresh_period (15)
    feed.set_last_updated (create {DATE_TIME}.make_now
    --)

    -- Add some simple items, use 'feed.
    --last_added_item' or directly create an item for
    --finer control
    feed.new_item ("Version 23 released!", create {
    --HTTP_URL}.make ("http://eiffelrss.berlios.de/Main
    --/News"), "Version 23 of EiffelRSS got release
    --today. Happy syndicating!")

    feed.new_item ("EiffelRSS wins award", create {
    --HTTP_URL}.make ("http://eiffelrss.berlios.de/Main
    --/Awards"), "EiffelRSS has been awarded by ISE as
    --best syndication software written in Eiffel. For
    --more info see award-winning pages: http://
    --eiffelrss.berlios.de")

    -- Create feed manager
    create feed_manager.make
    feed_manager.add (feed, "http://eiffelrss.berlios.
    --de/Main/AllRecentChanges?action=rss")
    feed_manager.refresh_all
end

feature — Arguments

    feed: FEED
        — Example feed

    feed_manager: FEED_MANAGER
        — Feed manager

end — class FEED_MANAGER_EXAMPLE

```

3.3 Features

3.3.1 Initialization

make

```
make
    — Create a new feed manager with default refresh \
    →period '30'
```

make_custom

```
make_custom (a_refresh_period: INTEGER)
    — Create a new feed manager with default refresh \
    →period 'a_refresh_period'
```

3.3.2 Access

default_refresh_period

```
default_refresh_period: INTEGER
    — Default refresh period in minutes
```

last_added_feed

```
last_added_feed: FEED
    — feed that was last added
```

feed_addresses

```
feed_addresses: LINKED_LIST[STRING]
    — Returns a sortable list representation of the \
    →feeds saved in FEED_MANAGER
```

feed_links

```
feed_links: LINKED_LIST[STRING]
    — Returns a sortable list representation of the \
    →feeds saved in FEED_MANAGER
```

3.3.3 Setter

set_default_refresh_period

```
set_default_refresh_period (a_refresh_period: INTEGER)
    — Set refresh periode in minutes
```

3.3.4 Element change

add

```
add (feed: FEED; url: STRING)
    — Add 'feed'
```

add_from_url

```
add_from_url (url: STRING)
    — Add feed with URL 'url'
```

3.3.5 Refresh

refresh

```
refresh (url: STRING)
    — Refresh feed with URL 'url', if the feed is \
    → outdated
```

refresh_force

```
refresh_force (url: STRING)
    — Refresh feed with URL 'url', even if the feed is \
    → not outdated
```

refresh_all

```
refresh_all
    — Refresh all feeds, if they are outdated
```

refresh_all_force

```
refresh_all_force
    — Refresh all feeds , even if they are not outdated
```

3.3.6 Conversion**list_representation**

```
list_representation: SORTABLE_TWO_WAY_LIST[FEED]
    — Returns a sortable list representation of the \
    → feeds saved in FEED_MANAGER
```

3.3.7 Conversion (sort)**sorted_by_last_updated**

```
sorted_by_last_updated: SORTABLE_TWO_WAY_LIST[FEED]
    — Returns a sorted list representation of the feeds \
    → , sorted by 'last_updated '
```

sorted_by_title

```
sorted_by_title: SORTABLE_TWO_WAY_LIST[FEED]
    — Returns a sorted list representation of the feeds \
    → , sorted by 'title '
```

sorted_by_link

```
sorted_by_link: SORTABLE_TWO_WAY_LIST[FEED]
    — Returns a sorted list representation of the feeds \
    → , sorted by 'link '
```

sorted_by_description

```
sorted_by_description: SORTABLE_TWO_WAY_LIST[FEED]
    — Returns a sorted list representation of the feeds \
    → , sorted by 'description '
```

reverse_sorted_by_last_updated

```
reverse_sorted_by_last_updated: SORTABLE_TWO_WAY_LIST[\nFEED]
    — Returns a sorted list representation of the feeds \
    →, reverse sorted by 'last_updated'
```

reverse_sorted_by_title

```
reverse_sorted_by_title: SORTABLE_TWO_WAY_LIST[FEED]
    — Returns a sorted list representation of the feeds \
    →, reverse sorted by 'title'
```

reverse_sorted_by_link

```
reverse_sorted_by_link: SORTABLE_TWO_WAY_LIST[FEED]
    — Returns a sorted list representation of the feeds \
    →, reverse sorted by 'link'
```

reverse_sorted_by_description

```
reverse_sorted_by_description: SORTABLE_TWO_WAY_LIST[\nFEED]
    — Returns a sorted list representation of the feeds \
    →, reverse sorted by 'description'
```

Chapter 4

Class FEED_READER

4.1 Overview

FEED_READER is a helper class which manages everything to load a feed. It converts the data to an XML document object, detects the format of the feed and uses the according reader object to convert the XML document into a FEED object.

See figure 4.1 for an overview of the class.

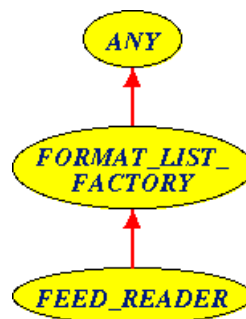


Figure 4.1: BON diagram of class FEED_READER

4.2 Usage

```

class
  READER_EXAMPLE

create
  make

feature — Initialization
  
```

```

make is
  — Creation procedure.
  local
    location: STRING
    reader: FEED_READER
    feed: FEED
  do
    — Get a feed location from the user
    io.put_string ("Enter an URL: ")
    io.read_line

    location := io.last_string.twin

    — Create the reader
    create reader.make_url (location)

    — Get the feed
    feed := reader.read

    — Print feed
    io.put_string ("%NReceived feed:%N")
    io.put_string ("===== %N %N %N")
    io.put_string (feed.to_string)
  end
end — class READER_EXAMPLE

```

4.3 Features

4.3.1 Initialization

make_url

```

make_url (a_url: STRING)
  — Create with 'a_url' as source of feed

```

4.3.2 Basic operations

read

```

read: FEED
  — Load the data from the given url into a FEED

```

Chapter 5

Class FEED_WRITER

5.1 Overview

FEED_WRITER is a helper class which manages everything to write a feed. It converts the data from an existing FEED object into an XML document object and saves it into a local file.

5.2 Usage

```
class
  WRITER_EXAMPLE

create
  make

feature — Initialization

  make is
    — Creation procedure.
  local
    feed: FEED
    writer: FEED_WRITER
  do
    — Create a simple feed
    create feed.make ("EiffelRSS", create {HTTP_URL}.
    ↪make ("http://eiffelrss.berlios.de/Main/
    ↪AllRecentChanges?action=rss"), "EiffelRSS news")

    — Add some simple items
    feed.new_item ("Version 23 released!", create {
    ↪HTTP_URL}.make ("http://eiffelrss.berlios.de/Main/
```

```

→/News"), "Version 23 of EiffelRSS got release \
→today. Happy syndicating!")
feed.new_item ("Microsoft uses EiffelRSS", create {
→{HTTP_URL}.make ("http://eiffelrss.berlios.de/\
→Main/WhoUsesEiffelRSS"), "Microsoft announced in \
→a press release today that they will use \
→EiffelRSS to syndicate news on their website.")
feed.new_item ("EiffelRSS wins award", create {
→HTTP_URL}.make ("http://eiffelrss.berlios.de/Main\
→/Awards"), "EiffelRSS has been awarded by ISE as \
→best syndication software written in Eiffel. For \
→more info see award-winning pages: http://\
→eiffelrss.berlios.de")

— Write feed to file
create writer.make_feed (feed)
writer.write ("example.xml", "RSS 2.0")
end

end — class WRITER_EXAMPLE

```

5.3 Features

5.3.1 Initialization

make_feed

```

make_feed (a_feed: FEED) is
— Create a writer object for the feed 'a_feed'

```

5.3.2 Basic operations

write

```

write (a_filename, a_format: STRING) is
— Write the feed to a local file with 'a_filename' in \
→ the format 'a_format'
— You can enumerate all available formats with \
→FORMAT_LIST (see FORMATS)

```

Part II

FEED

Chapter 6

Overview

FEED is the central datastructure of EiffelRSS. It defines an abstract syndication feed.

See figure 6.1 for an overview of the cluster.

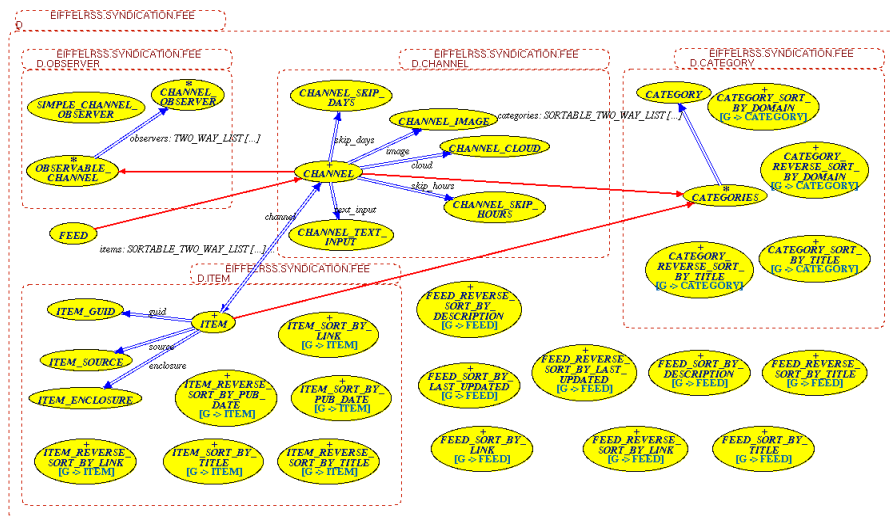


Figure 6.1: BON diagram of cluster FEED

Chapter 7

Usage

```

class
    FEED_EXAMPLE

create
    make

feature — Initialization

    make is
        — Creation procedure.
        do
            — Create a simple feed with some categories
            create feed.make ("EiffelRSS", create {HTTP_URL}.
            ↪make ("http://eiffelrss.berlios.de"), "EiffelRSS ↪
            ↪news")
            feed.add_category (create {CATEGORY}.make_title ("↪
            ↪RSS"))
            feed.add_category (create {CATEGORY}.make_title ("↪
            ↪Programming"))
            feed.add_category (create {CATEGORY}.make_title ("↪
            ↪Eiffel"))

            — Add a cloud to feed
            feed.create_cloud ("eiffelrss.berlios.de", 80, "/↪
            ↪RPC2", "xmlStorageSystem.rssPleaseNotify", "xml↪
            ↪rpc")

            — Add an image to feed
            feed.create_image (create {HTTP_URL}.make ("http↪
            ↪://eiffelrss.berlios.de/logo.png"), "EiffelRSS", ↪
            ↪create {HTTP_URL}.make ("http://eiffelrss.berlios↪
            ↪.de"))

```

```

— Add a text input field to feed
feed.create_text_input ("Search", "Search award-
-winning pages", "search", create {HTTP_URL}.make \
-("http://eiffelrss.berlios.de/Main/SearchWiki/"))

— Add some simple items, use 'feed.\
-last_added_item' or directly create an item for \
-finer control
feed.new_item ("Version 23 released!", create {\
-HTTP_URL}.make ("http://eiffelrss.berlios.de/Main\
-/News"), "Version 23 of EiffelRSS got release \
-today. Happy syndicating!")
feed.last_added_item.add_category (create {\
-CATEGORY}.make_title_domain ("News", create {\
-HTTP_URL}.make ("http://eiffelrss.berlios.de/Main\
-/News/")))

feed.new_item ("EiffelRSS wins award", create {\
-HTTP_URL}.make ("http://eiffelrss.berlios.de/Main\
-/Awards"), "EiffelRSS has been awarded by ISE as \
-best syndication software written in Eiffel. For \
-more info see award-winning pages: http://\
-eiffelrss.berlios.de")
feed.last_added_item.set_guid (create {ITEM_GUID}.\
-make_perma_link ("http://eiffelrss.berlios.de/\
-newsItem42"))

— Print feed
io.put_string ("Sample feed:%N")
io.put_string ("=====%N%N%N")
io.put_string (feed.to_string)
end

feature — Arguments
    feed: FEED
        — Example feed
end — class FEED_EXAMPLE

```

Chapter 8

Class FEED

8.1 Overview

FEED implements an abstract syndication feed.

8.2 Features

FEED inherits from CHANNEL, so all the features of CHANNEL are available as well.

8.2.1 Initialization

make_from_channel

make_from_channel (a_channel: CHANNEL) — <i>Create a new feed from an existing channel</i>
--

8.2.2 Access

last_updated

last_updated : DATE_TIME — <i>Time the channel was last updated</i>

refresh_period

refresh_period : INTEGER — <i>Refresh period in minutes</i>

8.2.3 Setter

set_channel

```
set_channel (a_channel: CHANNEL)
  — Set channel
```

set_refresh_period

```
set_refresh_period (a_refresh_period: INTEGER)
  — Set refresh periode in minutes
```

set_last_updated

```
set_last_updated (date: DATE_TIME)
  — Set time this channel was last updated
```

8.2.4 Status

has_refresh_period

```
has_refresh_period: BOOLEAN
  — Is 'refresh_period' set?
```

has_last_updated

```
has_last_updated: BOOLEAN
  — Is 'last_updated' set?
```

is_outdated

```
is_outdated: BOOLEAN
  — Is the feed outdated?
```

is_outdated_default

```
is_outdated_default (default_refresh_period: INTEGER) : \
-BOOLEAN
  — Is the feed outdated?
  — Use either 'refresh_period' or '\
- default_refresh_period' to determine
  — whether the feed is outdated.
```

8.2.5 Basic operations**create_cloud**

```
create_cloud (a_domain: STRING; a_port: INTEGER; a_path:\
- STRING; a_register_procedure: STRING; a_protocol: \
-STRING)
  — Create and add a cloud
```

create_image

```
create_image (a_url: URL; a_title: STRING; a_link: URL)
  — Create and add an image with URL, title , and link
```

create_text_input

```
create_text_input (a_title: STRING; a_description: \
-STRING; a_name: STRING; a_link: URL)
  — Create and add a text input with URL, title , and \
- link
```

new_item

```
new_item (a_title: STRING; a_link: URL; a_description: \
-STRING)
  — Create an item with title , link and description
```

8.2.6 Debug

to_string

to_string: **STRING**

- *Returns a string representation of feed*
- *This feature is especially useful for debugging*

Chapter 9

Class CHANNEL

9.1 Overview

CHANNEL is a class for abstract syndication channels. It uses the subclasses CHANNEL_CLOUD, CHANNEL_IMAGE and CHANNEL_TEXT_INPUT.

9.2 Features

9.2.1 Initialization

make

```
make (a_title: STRING; a_link: URL; a_description: \
-STRING)
  — Create a channel with title , link and description
```

9.2.2 Access

title

```
title: STRING
  — Channel title
```

link

```
link: URL
  — Channel link
```

description

```
description: STRING  
— Channel description
```

language

```
language: STRING  
— Channel language
```

copyright

```
copyright: STRING  
— Channel copyright
```

managing_editor

```
managing_editor: STRING  
— Channel managing editor
```

web_master

```
web_master: STRING  
— Channel web master
```

pub_date

```
pub_date: DATE_TIME  
— Channel pulication date
```

last_build_date

```
last_build_date: DATE_TIME  
— Channel last build date
```

feed_generator

```
feed_generator: STRING  
    — Channel feed generator
```

docs

```
docs: URL  
    — Channel docs
```

cloud

```
cloud: CHANNEL_CLOUD  
    — Channel cloud
```

ttl

```
ttl: INTEGER  
    — Channel time to live in minutes
```

image

```
image: CHANNEL_IMAGE  
    — Channel image
```

text_input

```
text_input: CHANNEL_TEXT_INPUT  
    — Channel text input
```

skip_hours

```
skip_hours: CHANNEL_SKIP_HOURS  
    — Channel skip hours
```

skip_days

```
skip_days: CHANNEL_SKIP_DAYS  
— Channel skip days
```

items

```
items: SORTABLE_TWO_WAY_LIST[ITEM]  
— Channel items
```

9.2.3 Access (RSS 0.91)**rating**

```
rating: STRING  
— Channel rating
```

9.2.4 Access (RSS 1.0)**items_toc**

```
items_toc: TWO_WAY_LIST[URL]  
— Channel items table of content
```

textinput

```
textinput: URL  
— Channel textinput URL
```

9.2.5 Access (categories)**categories**

```
categories: SORTABLE_TWO_WAY_LIST[CATEGORY]  
— Categories list containing category items
```

9.2.6 Access (observers)

observers

```
observers: TWO_WAY_LIST[CHANNEL_OBSERVER]
— List of subscribed observers
```

9.2.7 Access (metadata)

last_added_item

```
last_added_item: ITEM
— The last added channel item
```

9.2.8 Setter

set_title

```
set_title (a_title: STRING)
— Set title to 'a_title'
```

set_link

```
set_link (a_link: URL)
— Set link to 'a_link'
```

set_description

```
set_description (a_description: STRING)
— Set description to 'a_description'
```

set_language

```
set_language (a_language: STRING)
— Channel language
```

set_copyright

```
set_copyright (a_copyright: STRING)  
    — Channel copyright
```

set_managing_editor

```
set_managing_editor (a_managing_editor: STRING)  
    — Channel managing editor
```

set_web_master

```
set_web_master (a_web_master: STRING)  
    — Channel web master
```

set_pub_date

```
set_pub_date (date: DATE_TIME)  
    — Channel pulication date
```

set_last_build_date

```
set_last_build_date (date: DATE_TIME)  
    — Channel last build date
```

set_feed_generator

```
set_feed_generator (a_feed_generator: STRING)  
    — Channel feed generator
```

set_docs

```
set_docs (url: URL)  
    — Channel docs
```

set_cloud

```
set_cloud (a_cloud: CHANNEL_CLOUD)
    — Channel cloud
```

set_ttl

```
set_ttl (a_ttl: INTEGER)
    — Channel time to live in minutes
```

set_image

```
set_image (an_image: CHANNEL_IMAGE)
    — Channel image
```

set_text_input

```
set_text_input (a_text_input: CHANNEL_TEXT_INPUT)
    — Channel text input
```

set_skip_hours

```
set_skip_hours (some_skip_hours: CHANNEL_SKIP_HOURS)
    — Channel skip hours
```

set_skip_days

```
set_skip_days (some_skip_days: CHANNEL_SKIP_DAYS)
    — Channel skip days
```

set_items

```
set_items (item_list: like items)
    — Channel items
```

9.2.9 Setter (RSS 0.91)

set_rating

```
set_rating (rating_value: STRING)  
  — Channel rating
```

9.2.10 Setter (RSS 1.0)

set_items_toc

```
set_items_toc (item_toc_list: like items_toc)  
  — Channel items table of content
```

set_textinput

```
set_textinput (url: URL)  
  — Channel textinput URL
```

9.2.11 Setter (categories)

set_categories

```
set_categories (category_list: like categories)  
  — Set categories with a new category list
```

9.2.12 Setter (observers)

set_observers

```
set_observers (observer_list: like observers)  
  — List of subscribed observers
```

9.2.13 Status

has_language

```
has_language: BOOLEAN  
  — Is 'language' set and non-empty?
```

has_copyright

```
has_copyright: BOOLEAN  
— Is 'copyright' set and non-empty?
```

has_managing_editor

```
has_managing_editor: BOOLEAN  
— Is 'managing_editor' set and non-empty?
```

has_web_master

```
has_web_master: BOOLEAN  
— Is 'web_master' set and non-empty?
```

has_pub_date

```
has_pub_date: BOOLEAN  
— Is 'pub_date' set?
```

has_last_build_date

```
has_last_build_date: BOOLEAN  
— Is 'last_build_date' set?
```

has_feed_generator

```
has_feed_generator: BOOLEAN  
— Is 'feed_generator' set and non-empty?
```

has_docs

```
has_docs: BOOLEAN  
— Is 'docs' set?
```

has_cloud

```
has_cloud: BOOLEAN  
— Is 'cloud' set?
```

has_ttl

```
has_ttl: BOOLEAN  
— Is 'ttl' set?
```

has_image

```
has_image: BOOLEAN  
— Is 'image' set?
```

has_text_input

```
has_text_input: BOOLEAN  
— Is 'text_input' set?
```

has_skip_hours

```
has_skip_hours: BOOLEAN  
— Is 'skip_hours' set?
```

has_skip_days

```
has_skip_days: BOOLEAN  
— Is 'skip_days' set?
```

has_items

```
has_items: BOOLEAN  
— Is 'items' set?
```

9.2.14 Status (RSS 0.91)

has_rating

```
has_rating: BOOLEAN
— Is 'rating' set and non-empty?
```

9.2.15 Status (RSS 1.0)

has_items_toc

```
has_items_toc: BOOLEAN
— Is 'items_toc' set?
```

has_textinput

```
has_textinput: BOOLEAN
— Is 'textinput' set?
```

9.2.16 Status (categories)

has_categories

```
has_categories: BOOLEAN
— Are there any categories?
```

9.2.17 Status (observers)

has_observers

```
has_observers: BOOLEAN
— Is 'observers' set?
```

9.2.18 Status (metadata)

has_last_added_item

```
has_last_added_item: BOOLEAN
— Is 'last_added_item' set?
```

9.2.19 Basic operations

add_skip_hour

```
add_skip_hour (skip_hour: INTEGER)
  — Add a skip hour.
  — Only 0 <= 'skip_hour' <= 23 is valid
```

remove_skip_hour

```
remove_skip_hour (skip_hour: INTEGER)
  — Remove a skip hour
```

add_skip_day

```
add_skip_day (skip_day: STRING)
  — Add a skip day.
  — 'skip_day' will be added to 'skip_days' with the \
  →first letter to upper
  — and the rest to lower. For example: 'mOnDaY' is \
  →added as 'Monday'
```

remove_skip_day

```
remove_skip_day (skip_day: STRING)
  — Remove a skip day
```

add_item

```
add_item (item: ITEM)
  — Add an item
```

remove_item

```
remove_item (item: ITEM)
  — Remove an item
```

9.2.20 Basic operations (RSS 1.0)

add_item_toc

```
add_item_toc (item_toc: URL)
  — Add an item to the TOC
```

remove_item_toc

```
remove_item_toc (item_toc: URL)
  — Remove an item from the TOC
```

9.2.21 Basic operations (categories)

add_category

```
add_category (category: CATEGORY)
  — Add a category item
```

remove_category

```
remove_category (category: CATEGORY)
  — Remove a category item
```

9.2.22 Basic operations (observers)

add_observer

```
add_observer (an_observer: CHANNEL_OBSERVER)
  — Add an observer
```

remove_observer

```
remove_observer (an_observer: CHANNEL_OBSERVER)
  — Remove an observer
```

9.2.23 Sort

sort_items_by_title

```
sort_items_by_title
— Sort items by title
```

sort_items_by_pub_date

```
sort_items_by_pub_date
— Sort items by publication date
```

sort_items_by_link

```
sort_items_by_link
— Sort items by link
```

reverse_sort_items_by_title

```
reverse_sort_items_by_title
— Reverse sort items by title
```

reverse_sort_items_by_pub_date

```
reverse_sort_items_by_pub_date
— Reverse sort items by publication date
```

reverse_sort_items_by_link

```
reverse_sort_items_by_link
— Reverse sort items by link
```

9.2.24 Sort (categories)

sort_categories_by_title

```
sort_categories_by_title
— Sort categories by title
```

sort_categories_by_domain

```
sort_categories_by_domain
  — Sort categories by domain
```

reverse_sort_categories_by_title

```
reverse_sort_categories_by_title
  — Reverse sort categories by title
```

reverse_sort_categories_by_domain

```
reverse_sort_categories_by_domain
  — Reverse sort categories by domain
```

9.2.25 Debug**to_string**

```
to_string: STRING
  — Returns a string representation of channel
  — This feature is especially useful for debugging
```

9.3 Subclass CHANNEL_CLOUD**9.3.1 Initialization****make**

```
make (a_domain: STRING; a_port: INTEGER; a_path: STRING; \
  → a_register_procedure: STRING; a_protocol: STRING)
  — Create a channel cloud with domain, port, path, \
  → register procedure and protocol
```

9.3.2 Access**domain**

```
domain: STRING
  — Domain of the cloud
```

port

```
port: INTEGER
— Port of the cloud
```

path

```
path: STRING
— Path of the cloud
```

register_procedure

```
register_procedure: STRING
— Register procedure of the cloud
```

protocol

```
protocol: STRING
— Protocol of the cloud
```

9.3.3 Setter**set_domain**

```
set_domain (a_domain: STRING)
— Set domain to 'a_domain'
```

set_port

```
set_port (a_port: INTEGER)
— Set port to 'a_port'
```

set_path

```
set_path (a_path: STRING)
— Set path to 'a_path'
```

set_register_procedure

```
set_register_procedure (a_register_procedure: STRING)  
  — Set register_procedure to 'a_register_procedure'
```

set_protocol

```
set_protocol (a_protocol: STRING)  
  — Set protocol to 'a_protocol'
```

9.3.4 Debug**to_string**

```
to_string: STRING is  
  — Returns a string representation of cloud  
  — This feature is especially useful for debugging
```

9.4 Subclass CHANNEL_IMAGE**9.4.1 Initialization****make**

```
make (a_url: URL; a_title: STRING; a_link: URL)  
  — Create a channel image with URL, title, and link
```

9.4.2 Constants**Default_width**

```
Default_width: INTEGER is 88  
  — Default width of the image
```

Max_width

```
Max_width: INTEGER is 144  
  — Maximum width of the image
```

9.4.3 Access

url

```
url: URL  
— URL of the image
```

title

```
title: STRING  
— Title of the image
```

link

```
link: URL  
— Link of the image
```

width, height

```
width, height: INTEGER  
— Width and height of the image
```

description

```
description: STRING  
— Description of the image
```

9.4.4 Setter

set_url

```
set_url (a_url: URL)  
— Set url to 'a_url'
```

set_title

```
set_title (a_title: STRING)  
— Set title to 'a_title'
```

set_link

```
set_link (a_link: URL)
  — Set link to 'a_link'
```

set_width

```
set_width (a_width: INTEGER)
  — Set width to 'a_width'
```

set_height

```
set_height (a_height: INTEGER)
  — Set heigh to 'a_heigh'
```

set_description

```
set_description (a_description: STRING)
  — Set to 'a_description'
```

9.4.5 Status**has_width**

```
has_width: BOOLEAN
  — Is 'width' set?
```

has_height

```
has_height: BOOLEAN
  — Is 'height' set?
```

has_description

```
has_description: BOOLEAN
  — Is 'description' set and non-empty?
```

9.4.6 Debug

to_string

```
to_string: STRING  
  — Returns a string representation of image  
  — This feature is especially useful for debugging
```

9.5 Subclass CHANNEL_TEXT_INPUT

9.5.1 Initialization

make

```
make (a_title: STRING; a_description: STRING; a_name: STRING;  
      a_link: URL)  
  — Create a channel text input with title , description ,  
  — name and link
```

9.5.2 Access

title

```
title: STRING  
  — Title of the text input
```

description

```
description: STRING  
  — Description of the text input
```

name

```
name: STRING  
  — Name of the text input
```

link

```
link: URL
— Link of the text input
```

9.5.3 Setter

set_title

```
set_title (a_title: STRING)
— Set title to 'a_title'
```

set_description

```
set_description (a_description: STRING)
— Set description to 'a_description'
```

set_name

```
set_name (a_name: STRING)
— Set name to 'a_name'
```

set_link

```
set_link (a_link: URL)
— Set link to 'a_link'
```

9.5.4 Debug

to_string

```
to_string: STRING is
— Returns a string representation of text input
— This feature is especially useful for debugging
```

Chapter 10

Class ITEM

10.1 Overview

ITEM is a class for feed items. It uses the subclasses ITEM_ENCLOSURE, ITEM_GUID and ITEM_SOURCE.

10.2 Features

10.2.1 Initialization

make

```
make (a_channel: CHANNEL; a_title: STRING; a_link: URL; \
-a_description: STRING)
    — Create an item with title , link and description
```

make_title

```
make_title (a_channel: CHANNEL; a_title: STRING)
    — Create an item with title
```

make_description

```
make_description (a_channel: CHANNEL; a_description: \
-STRING)
    — Create an item with description
```

10.2.2 Access

channel

```
channel: CHANNEL
— Backlink to the corresponding channel
```

title

```
title: STRING
— Item title
```

link

```
link: URL
— Item link
```

description

```
description: STRING
— Item description
```

author

```
author: STRING
— Item author
```

comments

```
comments: URL
— Item comment URL
```

enclosure

```
enclosure: ITEM_ENCLOSURE
— Item enclosure
```

guid

```
guid: ITEM_GUID
— Item globally unique identifier (guid)
```

pub_date

```
pub_date: DATE_TIME
— Item publication date
```

source

```
source: ITEM_SOURCE
— Item source
```

10.2.3 Access (categories)**categories**

```
categories: SORTABLE_TWO_WAY_LIST[CATEGORY]
— Categories list containing category items
```

10.2.4 Access (metadata)**date_found**

```
date_found: DATE TIME
— Item date found
```

is_read

```
is_read: BOOLEAN
— Is the item read?
```

10.2.5 Setter

set_channel

```
set_channel (a_channel: CHANNEL)
  — Set channel to 'a_channel'
```

set_title

```
set_title (a_title: STRING)
  — Set title to 'a_title'
```

set_link

```
set_link (a_link: URL)
  — Set link to 'a_link'
```

set_description

```
set_description (a_description: STRING)
  — Set description to 'a_description'
```

set_author

```
set_author (an_author: STRING)
  — Set author to 'an_author'
```

set_comments

```
set_comments (url: URL)
  — Set comments to 'url'
```

set_enclosure

```
set_enclosure (an_enclosure: ITEM_ENCLOSURE)
  — Set enclosure to 'an_enclosure'
```

set_guid

```
set_guid (a_guid: ITEM_GUID)
    — Set guid to 'a_guid'
```

set_pub_date

```
set_pub_date (date: DATE_TIME)
    — Set publication date to 'date'
```

set_source

```
set_source (a_source: ITEM_SOURCE)
    — Set source to 'a_source'
```

10.2.6 Setter (categories)**set_categories**

```
set_categories (category_list: like categories)
    — Set categories with a new category list
```

10.2.7 Setter (metadata)**set_date_found**

```
set_date_found (date: DATE_TIME)
    — Set date_found to 'date'
```

set_read

```
set_read (value: BOOLEAN)
    — Set is_read to 'value'
```

10.2.8 Status

has_title

```
has_title: BOOLEAN  
— Is 'title' set and non-empty?
```

has_link

```
has_link: BOOLEAN  
— Is 'link' set?
```

has_description

```
has_description: BOOLEAN  
— Is 'description' set and non-empty?
```

has_author

```
has_author: BOOLEAN  
— Is 'author' set and non-empty?
```

has_comments

```
has_comments: BOOLEAN  
— Is 'comments' set?
```

has_enclosure

```
has_enclosure: BOOLEAN  
— Is 'enclosure' set?
```

has_guid

```
has_guid: BOOLEAN  
— Is 'guid' set?
```

has_pub_date

```
has_pub_date: BOOLEAN
— Is 'pub_date' set?
```

has_source

```
has_source: BOOLEAN
— Is 'source' set?
```

10.2.9 Status (categories)**has_categories**

```
has_categories: BOOLEAN
— Are there any categories?
```

10.2.10 Basic operations (categories)**add_category**

```
add_category (category: CATEGORY)
— Add a category item
```

remove_category

```
remove_category (category: CATEGORY)
— Remove a category item
```

10.2.11 Sort (categories)**sort_categories_by_title**

```
sort_categories_by_title
— Sort categories by title
```

sort_categories_by_domain

```
sort_categories_by_domain
  — Sort categories by domain
```

reverse_sort_categories_by_title

```
reverse_sort_categories_by_title
  — Reverse sort categories by title
```

reverse_sort_categories_by_domain

```
reverse_sort_categories_by_domain
  — Reverse sort categories by domain
```

10.2.12 Debug**to_string**

```
to_string: STRING
  — Returns a string representation of item
  — This feature is especially useful for debugging
```

10.3 Subclass ITEM_ENCLOSURE**10.3.1 Initialization****make**

```
make (a_url: URL; a_length: INTEGER; a_type: STRING)
  — Create an item enclosure
```

10.3.2 Access**url**

```
url: URL
  — URL of the enclosure
```

length

```
length: INTEGER  
  — Length in bytes of the enclosure
```

type

```
type: STRING  
  — MIME type of the enclosure
```

10.3.3 Setter**set_url**

```
set_url (a_url: URL)  
  — Set the URL to 'url'
```

set_length

```
set_length (a_length: INTEGER)  
  — Set the length to 'a_length'
```

set_type

```
set_type (a_type: STRING)  
  — Set the MIME type to 'a_type'
```

10.3.4 Debug**to_string**

```
to_string: STRING  
  — Returns a string representation of enclosure  
  — This feature is especially useful for debugging
```

10.4 Subclass ITEM_GUID

10.4.1 Initialization

make

```
make (a_guid: STRING)  
  — Create an item guid with 'is_perma_link' set to \  
  → False
```

make_perma_link

```
make_perma_link (a_guid: STRING)  
  — Create an item guid with 'is_perma_link' set to \  
  → True
```

10.4.2 Access

guid

```
guid: STRING  
  — String representing a globally unique identifier (\  
  → guid)
```

is_perma_link

```
is_perma_link: BOOLEAN  
  — Is this guid a perma link?
```

10.4.3 Setter

set_guid

```
set_guid (a_guid: STRING)  
  — Set guid to 'a_guid'
```

set_perma_link

```
set_perma_link (value: BOOLEAN)  
  — Set is_perma_link to 'value'
```

10.4.4 Debug**to_string**

```
to_string: STRING  
  — Returns a string representation of guid  
  — This feature is especially useful for debugging
```

10.5 Subclass ITEM_SOURCE**10.5.1 Initialization****make**

```
make (a_name: STRING; a_url: URL)  
  — Create an item source
```

10.5.2 Access**name**

```
name: STRING  
  — Name of the item source
```

url

```
url: URL  
  — URL of the item source
```

10.5.3 Setter**set_name**

```
set_name (a_name: STRING)  
  — Set name to 'a_name'
```

set_url

```
set_url (a_url: URL)
    — Set url to 'a_url'
```

10.5.4 Debug**to_string**

```
to_string: STRING
    — Returns a string representation of source
    — This feature is especially useful for debugging
```

Chapter 11

Class CATEGORY

11.1 Overview

CATEGORY is a class for channel and item categories.

11.2 Features

11.2.1 Initialization

make

```
make
  — Create a category with title '[unnamed category]')
```

make_title

```
make_title (a_title: STRING)
  — Create a category with title 'a_title '
```

make_title_domain

```
make_title_domain (a_title: STRING; a_domain: URL)
  — Create a category with title 'a_title ' and domain '↘
  →a_domain '
```

11.2.2 Access

title

```
title: STRING  
— Category title
```

domain

```
domain: URL  
— Category domain
```

11.2.3 Setter

set_title

```
set_title (a_title: STRING)  
— Set title to to 'a_title'
```

set_domain

```
set_domain (url: URL)  
— Set domain to to 'url'
```

11.2.4 Status

has_domain

```
has_domain: BOOLEAN  
— Is 'domain' set?
```

11.2.5 Debug

to_string

```
to_string: STRING  
— Returns a string representation of category  
— This feature is especially useful for debugging
```

Chapter 12

Observers

12.1 Overview

FEED is observable. This means it is of type `OBSERVABLE_CHANNEL` and has features to notify subscribed observers in the case of updates.

`CHANNEL_OBSERVER` is a deferred class which observers have to implement to observe a feed. There is also a simple observer, `SIMPLE_CHANNEL_OBSERVER` which you can use as a starting point for your own observer classes.

12.2 Class `OBSERVABLE_CHANNEL`

12.2.1 Access

observers

```
observers: TWO_WAY_LIST[CHANNEL_OBSERVER]
— List of subscribed observers
```

12.2.2 Setter

set_observers

```
set_observers (observer_list: like observers)
— List of subscribed observers
```

12.2.3 Status

has_observers

```
has_observers: BOOLEAN
  — Is 'observers' set?
```

12.2.4 Basic operations

add_observer

```
add_observer (an_observer: CHANNEL_OBSERVER)
  — Add an observer
```

remove_observer

```
remove_observer (an_observer: CHANNEL_OBSERVER)
  — Remove an observer
```

12.3 Class CHANNEL_OBSERVER

12.3.1 Observer

item_added

```
item_added (new_item: ITEM)
  — Is called when a new item is added to this channel
deferred
```

channel_updated

```
channel_updated (channel: CHANNEL)
  — Is called when a new channel is added
deferred
```

12.4 Class SIMPLE_CHANNEL_OBSERVER

12.4.1 Access

added_item

```
added_item: ITEM
    — The newly added item
```

updated_channel

```
updated_channel: CHANNEL
    — The newly updated channel
```

12.4.2 Observer

item_added

```
item_added (new_item: ITEM)
    — Is called when a new item is added to this channel
```

channel_updated

```
channel_updated (channel: CHANNEL)
    — Is called when a new channel is added
```

Part III

FORMATS

Chapter 13

Overview

FORMATS contains classes to manage the different formats and the actual implementation of these formats.

Each format has a format, a writer and a reader object and a unique name. It also provides a feature which can detect if the format can read a certain XML document.

There is a special format called “Error” which is used whenever an error occurs.

See figure 13.1 for an overview of the cluster.

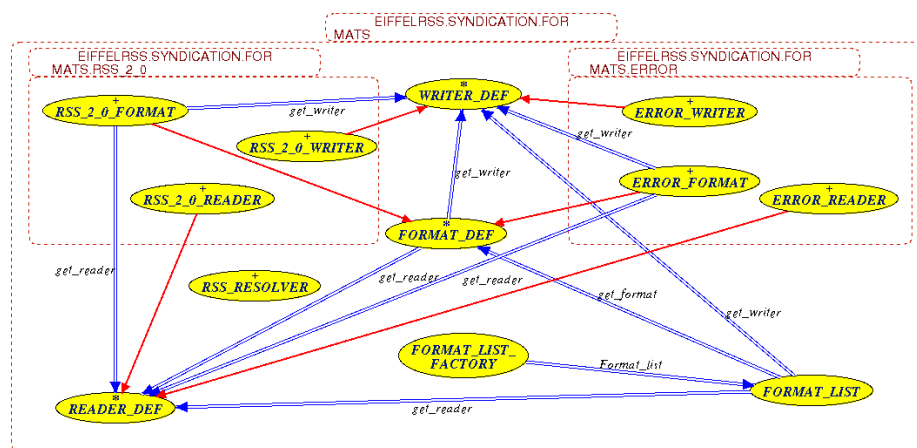


Figure 13.1: BON diagram of cluster FORMATS

Chapter 14

Management: Class FORMAT_LIST

14.1 Overview

FORMAT_LIST manages the different formats.

14.2 Usage

FORMAT_LIST uses a singleton pattern, so to actually use the format list your class has to inherit from FORMAT_LIST_FACTORY.

FORMAT_LIST inherits from LINKED_LIST, so all the features of LINKED_LIST are available as well.

14.3 Features

14.3.1 Initialization

make_list

<p>make_list — <i>Create the object and add the default formats</i></p>
--

14.3.2 Access

get_reader

```
get_reader (a_name: STRING): READER_DEF  
— Get the reader object for the name 'a_name'
```

get_writer

```
get_writer (a_name: STRING): WRITER_DEF  
— Get the writer object for the name 'a_name'
```

get_format

```
get_format (a_name: STRING): FORMAT_DEF  
— Get the format object for the name 'a_name'
```

14.3.3 Detection

detect_format

```
detect_format (a_document: XM_DOCUMENT): STRING  
— Get the format name for 'a_document'
```

Chapter 15

Format implementations

15.1 Adding a new format

Adding a new format to EiffelRSS? is very easy. You have to provide three objects which inherit from the deferred base classes `FORMAT_DEF`, `READER_DEF` and `WRITER_DEF`. If you only want to implement a reader or a writer, you can return `ERROR_WRITER` respectively `ERROR_READER` for the other feature.

To actually add the format to the library, you have to extend `FORMAT_LIST` with an object of the format class.

15.2 Base classes

15.2.1 `FORMAT_DEF`

get_reader

```
get_reader: READER_DEF
    — Return a reader object
deferred
```

get_writer

```
get_writer: WRITER_DEF
    — Return a writer object
deferred
```

get_name

```
get_name: STRING  
  — Return the format name  
deferred
```

is_of_format

```
is_of_format (a_document: XM_DOCUMENT): BOOLEAN  
  — Is this document a feed of our type?
```

15.2.2 READER_DEF**read**

```
read (a_document: XM_DOCUMENT): FEED  
  — Parse the document and return a feed  
deferred
```

get_name

```
get_name: STRING  
  — Return a string with the format name  
deferred
```

read_or_default_element

```
read_or_default_element (a_element: XM_ELEMENT; ↵  
↵default_value: STRING): STRING  
  — Read the text of 'a_element' or use 'default_value' ↵  
  ↵ if 'a_element' is Void or empty
```

read_or_default_attribute

```
read_or_default_attribute (a_attribute: XM_ATTRIBUTE; ↵  
↵default_value: STRING): STRING  
  — Read the value of 'a_attribute' or use ' ↵  
  ↵ default_value' if 'a_element' is Void or empty
```

valid_element_text

```
valid_element_text (an_element: XM_ELEMENT; a_name: \
-STRING): BOOLEAN
  — Has the subelement 'a_name' of 'an_element' text?
```

read_date

```
read_date (a_string: STRING): DATE_TIME
  — Convert an RFC 822 date string to a DATE_TIME \
  →object
```

15.2.3 WRITER_DEF**get_name**

```
get_name: STRING
  — Return a string with the format name
deferred
```

writer

```
write (a_feed: FEED): XM_DOCUMENT
  — Export 'a_feed' into an xml document
deferred
```

15.3 Built-in formats**15.3.1 RSS 2.0**

RSS_2_0_FORMAT is an example implementation of the RSS 2.0 standard. It reads almost all the possible data and has a very basic writer.

15.3.2 Error

ERROR_FORMAT is a special format which is used whenever an error occurs. This removes a lot of sources of errors because the library can ensure that the reader and writer objects are never Void.

ERROR_READER returns a generated feed which has one item with the error message as description.