



**POD Translation**  
by *pod2pdf*

---

[ajf@afco.demon.co.uk](mailto:ajf@afco.demon.co.uk)

*eXpanda.pm*



# Table of Contents

## eXpanda.pm

|   |    |
|---|----|
| NAME  | 1  |
| VERSION   | 1  |
| SYNOPSIS  | 1  |
| DESCRIPTION   | 3  |
| INTERFACE   | 3  |
| new( filepath, %Options, )  | 3  |
| -filetype => String   | 3  |
| Apply( %Options, )  | 3  |
| -limit => [{ %Options }, {}, ... ]                                | 3  |
| -object => String   | 4  |
| -value => String   Hash reference # when using "-from" option     | 5  |
| -file => Filepath   | 5  |
| -from => String,  | 5  |
| -apply_information => { %Options}                                 | 6  |
| -reset => String,   | 6  |
| -label => String,   | 6  |
| Return( %Options, )   | 6  |
| -limit => [{ %Options }, {}, ... ],                               | 6  |
| -object => String,  | 6  |
| -network => { %Option},   | 7  |
| -hash => Boolean  | 8  |
| -get_information => { %Option }                                   | 8  |
| Analyze( %Options, )  | 8  |
| -method => String,  | 8  |
| -option => hash reference,  | 9  |
| -object => String   | 9  |
| -value => Hash reference  | 9  |
| out( filepath, %Options, )  | 9  |
| -filetype => String   | 9  |
| Follow 4 options are for Graph::Layout::Aesthetic module which... | 9  |
| knr => Float  | 9  |
| kmel => Float   | 9  |
| kner => Float   | 9  |
| kmei2 => Float  | 9  |
| Follow 3 options defines graphical output sizing attributes.      | 9  |
| -width => Float   | 9  |
| -height => Float  | 9  |
| -margin => Float  | 9  |
| Follows are general options for graphical output.                 | 9  |
| -no_direction => Boolean  | 9  |
| -no_node_label => Boolean   | 9  |
| -no_edge_label => Boolean   | 9  |
| -dual_arrow_spreads => Float                                      | 10 |
| -rotation => Float  | 10 |
| -object_size => Float   | 10 |
| -label_size => Float  | 10 |
| DIAGNOSTICS   | 10 |
| CONFIGURATION AND ENVIRONMENT                                     | 10 |
| DEPENDENCIES  | 10 |
| INCOMPATIBILITIES   | 10 |
| BUGS AND LIMITATIONS  | 10 |
| AUTHOR  | 10 |

|                        |    |
|------------------------|----|
| LICENCE AND COPYRIGHT  | 10 |
| DISCLAIMER OF WARRANTY | 10 |

## NAME

eXpanda - [Platform for observing interactome network with care a foot. ]

## VERSION

This document describes eXpanda version 0.0.5

## SYNOPSIS

```
#!/usr/bin/perl -w
use strict;
use eXpanda;

# Make Constructor
my $str = eXpanda->new('ribosome.sif');

# Analyze network in 'degree' method
$str->Analyze(
    -method => 'degree',
);

# Design node which suffuse '$label =~ /rps/' , with "fill" into red.
$str->Apply(
    -limit => [{
        -object => 'node:label',
        -operator => '=~',
        -identifier => 'rps'
    }],
    -object => 'node:design:fill',
    -value => 'red',
);

# Design node which suffuse '$label =~ /rrn/' , with "fill" into gradation
# from #00FF00 to #FFFFFF, using 'degree' scores.
$str->Apply(
    -limit => [{
        -object => 'node:label',
        -operator => '=~',
        -identifier => 'rrn'
    }],
    -from => 'node:score:degree',
    -object => 'node:design:fill',
    -value => {"#00FF00" => "#FFFFFF"},
);

# Design all nodes with 'w' (which means 'width') into 24 pixels.
$str->Apply(
    -object => 'node:design:w',
    -value => '24',
);

# Design node which suffuse '$label =~ /rpl/' , with "fill" into blue.
# This step overwrite with "fill" in earlier steps.
$str->Apply(
    -limit => [{
        -object => 'node:label',
        -operator => '=~',
        -identifier => 'rpl'
    }],
);
```

```

        -object => 'node:design:fill',
        -value => 'blue',
    );

# Design all nodes with 'font-size' (which defining font-size of label).
$str->Apply(
    -object => 'node:design:font-size',
    -value => '8',
);

# Design node which suffuse '$label eq 'rplV'', with all designs using
# file whose file path is './riobosome.style'.
$str->Apply(
    -limit => [{
        -object => "node:label",
        -operator => "eq",
        -identifier => "rplV"
    }],
    -object => "node:design",
    -file => './riobosome.style',
);

# Return list as sub graph which contains nodes which suffuse
# '$label =~ 'rpl'', and copy into $str2.
my $str2 = $str->Return(
    -limit => [{
        -object => 'node:label',
        -operator => '=~',
        -identifier => 'rpl',
    }],
    -object => 'as_network',
);

# Divide $str2 networks into each independent networks and
# save as "in$i.svg"($i is incremental number).
my $i = 1;
for my $in (@{$str2->Return(
    -network =>{
        -operator => 'independent',
    },
)}){
    $in->out("in$i\.svg",
        -object_size => 3,
    );
    $i++;
}

# Design node which join in list, as smiling in pink
my $hoge = undef;
for(qw(rpsB rplX rpsP rpmH rpsN)){
    push @{$hoge->{node}}, $_;
}
$str->Apply(
    -limit => [{
        -list => $hoge,
    }],
    -object => 'node:design:smile',
    -value => 'pink',
);

```

```

);

# Save network in 4x object size with no label, as 'ribosome.svg'.
$str->out('ribosome.svg',
        -object_size => 4,
        -l => 1,
);

```

## DESCRIPTION

eXpanda helps to observe interactome networks.

## INTERFACE

eXpanda have 5 modules in user interface which named **'new'**, **'Apply'**, **'Return'**, **'Analyze'** and **'out'**. Followed texts are detailed commentary on these interfaces.

### **new( filepath, %Options, )**

"new" is the module to make constructor. Capable file-types are 'SIF', 'Cytoscape GML', 'Two dimensioned hash reference', 'Database Flat Data'. File format rules are described in "supplementary.txt". If you choose hash reference, the structure formation is as follows: c.f) {a = {b = 1, c = 1,} c = {d = 1}} To make this hash, also code as:

```

my $hash = undef;
$hash->{a}->{b} = 1;
$hash->{a}->{c} = 1;
$hash->{c}->{d} = 1;

```

Follows are options.

#### **-filetype => String**

This option defines filetype to input string as 'sif', 'gml', 'str' and 'xml'. Default value is 'sif'.

#### **Example:**

```
my $str-new('ribosome.sif', -filetype = 'sif')
```

### **Apply( %Options, )**

"Apply" is the module to modificate network.

Options are as Follows.

#### **-limit => [{ %Options }, {}, ... ]**

"-limit" is the option for limiting to applying objects in the network.

Limitation are multiply definable as array references.

Options are as follows.

```

-object => I<String>,
-operator => I<String>,
-identifier => I<Float, Integer or String>,

```

or

```

-object => I<String>,
-list => I<Hash reference>,

```

"-object", "-operator" and "-identifier" are the set. These 3 terms evaluates to limit objects. If you put "node:label", "eq", "lacZ", then module takes applying only to the nodes whose label is lacZ.

Describing about "-object" is below.

"-operator" is the operator. Capable strings as follows. "eq", "ne", "=~", "!~"(regular expressions!), "==" , "!=" , "=", "<=", ">", "<".

"-identifier" defines identifying terms. It usually put that *Float*, but *String* "average" or "center" are capable, too. These values automatically fill in static analysis.

If the object has Float or Integer data, then you can put "-operator" and "-identifier" in follow words:

"top", *"Integer"* returns top Integer objects scored.  
 "bottom", *"Integer"* returns bottom Integer objects scored.

"-list" takes list matching with object label. You can input here as *eXpanda blessed structure*, *filepath* or *hash reference*.

If you want to query eXpanda structure, you may just put constructed structure here.

Describing rules of file format are also described in "supplementary.txt". Also you can input sif type file. The input is also capable from 'sif' file format.

Hash reference definition rules are as follows.

If you want to include node labeled "lacZ" and "lacY", then code as:

```
$list->{node}->{lacZ} = 1;
```

Also you can put the edge lacZ to lacY:

```
$list->{edge}->{lacZ}->{lacY} = 1;
```

**Note: If you limit only nodes, the edges between limited nodes are only applied.**

**Example:**

```
$str->Apply(
  -limit => [{
    -object => 'node:label',
    -operator => '=~',
    -identifier => 'rpl'
  }],
  {
    -list => $list_hash,
  },
  {
    -list => './list.limit',
  }
],
-object => 'node:design:fill',
-value => 'blue',
);
```

**-object => *String***

"-object" option defines the object information layered with ":" in follow 3 methods. If you want to manipulate objects to apply in degree score of nodes, then you may put "node:score:degree" into "-object". Also you can put "node:information:GI", "node:design:fill", etc. Originally, the constructed network has no data except "node:label" and "edge:label", when you constructed network with sif, gml or svg data. If you have constructed network from databases, some informations are put in "*object:information:\**". If you want data which have inputted in each object, please check "supplementary.txt": The supplementary text file.

You have to put informations or scores using "Apply" method or "Analyze" method after construct structure.

In this rule, the end term in "*object:design:\**" is capable with limited strings such as follow listed:

[node attributes]

node:design:w *Float*

node:design:stroke-width *Float*

node:design:stroke-dash *List of Float* ( c.f) "2" or "5,10")

node:design:font-size *Float*

node:design:opacity *Float* (0 to 1)

node:design:stroke-opacity *Float* (0 to 1)

node:design:font-opacity *Float* (0 to 1)

node:design:fill *Color*

node:design:stroke *Color*

node:design:smile *Color*

node:design:font-color *Color*

node:design:font-family *Installed font name*

node:design:type *'oval', 'diamond', 'star' or 'heart'*

```
[edge attributes]
edge:design:stroke-width Float
node:design:stroke-dash List of Float ( c.f "2" or "5,10")
edge:design:opacity Float (0 to 1)
edge:design:stroke Color
edge:design:font-size Float
edge:design:font-opacity Float (0 to 1)
edge:design:font-family Installed font name
edge:design:font-color Color
```

Details are described in "supplementary.txt".

**-value => *String* | *Hash reference* # when using "-from" option**

Input value.

If you change design of objects, the value of option must be under the rule. This rule is also describing in "supplementary.txt". When "-from" option is set, is described in the section of "-from" option. "-value" basically defined as *Hash Reference*.

**-file => *Filepath***

This takes 3 types of files. score data, design data, and information data.

When you are using this option, "-object" option must be set as "node:design", "edge:design", "all:design", "node:score", "edge:score", "all:score", "node:information", "edge:information", "all:information".

The file rule is also described in "supplementary.txt".

**Example:**

```
$str-Apply(
  -limit = [{
    -object = "node:label",
    -operator = "eq",
    -identifier = "rplV"
  }],
  -object = "node:design",
  -file = './riobosome.style',
);
```

**-from => *String*,**

eXpanda is able to change design of objects from scored information. This option defines that. For example, if you finished degree scoring, you can fill 'node:score:degree' in this option.

The applying object from this option might be design. Then you have to fill "-object" option to "node:design:*detail String*" or "edge:design:*detail String*". When you apply *detailString* as Float data ( c.f w, stroke-width..) the "-value" option should be {"*Float*" = "*float*"}. These values define min value and max value of applying object which determined from "-from" scores. The "-value" options are able to define as {"*Color*" = "*Color*"} when you set color data in "-object".

The list of object details, which can put in '-object' option when '-from' option is on, are as follows:

```
[node attributes]
node:design:w Float
node:design:stroke-width Float
node:design:opacity Float (0 to 1)
node:design:stroke-opacity Float (0 to 1)
node:design:font-opacity Float (0 to 1)
node:design:font-size Float
node:design:fill Color
node:design:stroke Color
node:design:smile Color
node:design:font-color Color
[edge attributes]
edge:design:stroke-width Float
```

```

edge:design:opacity Float
edge:design:font-size Float
edge:design:font-opacity Float (0 to 1)
edge:design:stroke Color
edge:design:font-color Color

```

**Example:**

```

$str-Apply(
  -from = 'node:score:degree',
  -object = 'node:design:fill',
  -value = {"#00FF00" = "#FFFFFF"},
);

```

**-apply\_information => { %Options }**

This option defines applying information using networks.

```

-object => I<String>, -from => I<String>, -to => I<String>, -organism => I<St

```

"-object" option defines resource of apply\_information. You may input here as "node:label" or "node:information:String".

"-from" option is the default label name type, as "GI", "Uniprot", etc.

"-to" option defines conversion target type.

"-organism" defines organism which network joins. Sometimes this option should be use. c.f) The network described in gene name ("-from" option should be "name").

The capable option name is also defined in "supplementary.txt".

**Example:**

```

$str-Apply(
  -apply_information = {
    -object = 'node:information:GI',
    -from = 'GI',
    -to = 'PIR'
  },
  -object = 'node:label',
);

```

**-reset => String,**

This takes resetting.

You can input same strings with "-object", or "all", "node:all", "edge:all", and "except\_xy".

"except\_xy" is developed for mainly Cytoscape. This value resets other than coordinates.

**Example:**

```

$str-Apply(
  -reset = 'except_xy',
);

```

**-label => String,**

This option applies object label from existing other informations. *String* format is completely same with "-object".

**Example:**

```

$str-Apply(
  -label = 'node:information:GO',
);

```

**Return( %Options, )**

"Return" is the module to return information of object, object as is, or sometime also returns eXpanda blessed subgraph or something like that. Options are as Follows.

**-limit => [ { %Options }, { }, ... ],**

Same as "-limit" option in Apply.

**-object => String,**

Almost same as "-object" option in Apply.

Exception:

This input accept the string "as\_network". In this case, you may set the follow option.

The other exception:

If you input "1|true" into the option "-hash", this option should be hash reference. Then "Return()" returns hash reference as you set.

**NOTE: If you input "-object" as "as\_network", input something in "-limit", and not input "-network", then you can get subgraphs which have only limited objects.**

**Example 1:**

```
$str-Return(
    -limit = [{
        -object = "node:information:name",
        -operator = "=~",
        -identifier = "^lac"
    }],
    -object = 'node:label',
);
```

**Example 2:**

```
$str-Return(
    -limit = [{
        -list = './list.limit',
    }],
    -object = 'as_network',
);
```

**Example 3:**

```
$str-Return(
    -limit = [{
        -object = "node:score:degree",
        -operator = "=",
        -identifier = "average"
    }],
    -hash = 1,
    -object = { 'node:label' = 'node:information:GO' }
);
```

**-network => { %Option},**

Further options are follows:

```
-operator => I<String>,
-with => I<eXpanda Blessed structure>,
-threshold => I<Integer>
```

This option defines return networks operation.

"-operator" is capable with strings "cap", "cup", "independent" and "clique".

In "cap" and "cup", you have to set "-with". Then this operation takes cap or cup with the other network sets in "-with".

If you choose "independent", "Return" returns each independent subgraphs in the network as blessed structure.

"Return" also returns each complete subgraphs if you choose "clique". Then you may set '-threshold' which determines threshold of node numbers of subgraphs. Default:4.

**Example:**

```
my $i = 1;
for my $in (@{$str2-Return(
    -network = {
        -operator = 'independent',
    },
)}){
    $in-out("in$i.svg",
        -debug_size = 3,
    );
    $i++;
}
```

-hash => *Boolean*

Default: 0

If you do not set "as\_network" in "-object", "Return" module always return values as array reference in default. If you change this option to "1", you can receive hash reference.

**Example 1:**

```
@list = @{$str-Return(
    -limit = [{
        -object = 'node:score:degree',
        -operator = '=',
        -identifier = 'average',
    }],
    -object = 'node:label',
)};
```

**Example 2:**

```
%index = %{$str-Return(
    -limit = [{
        -object = 'node:score:degree',
        -operator = '=',
        -identifier = 'average',
    }],
    -hash = 1,
    -object = {'node:label' = 'node:information:GO'}
)};
```

-get\_information => { *%Option* }

Further options are as follows:

```
-object => I<String>,
-from => I<String>,
-to => I<String>,
```

Definition of the option is basically same with "-apply\_information" in "Apply()".

This options get data using web services. If you using this option, you do not have to set "-object" option into "Return()". If you check "-hash" option, then it returns hash reference as {"-object in -get\_information" = "translated information", ...}. If not, then it returns list of translated information as array reference.

**Example:**

```
%function_list = %{$str-Return(
    -limit = [{
        -object = 'node:score:degree',
        -operator = 'top',
        -identifier = '10',
    }],
    -get_information = {
        -object = 'node:information:GI',
        -from = 'GI',
        -to = 'protein_function',
    },
    -hash = '1',
)};
```

**Analyze( %Options, )**

"Analyze()" is the module for topological analyzing on the network.

-method => *String*,

This option defines what method applies to network. Now this method supports following names as:

"degree", "normarized\_degree", "clique", "connectivity", "accuracy", "coverage", "count\_nodes" and "count\_edges".

These data are stores in "node:score:method\_name" or "edge:score:method\_name". It have return value when you set specific method name.

In detail, see "supplementary.txt".

-option => *hash reference*,

This is an method options for each method. This input may alters in each method.

What should be inputted in each methods are also described in "supplementary.txt".

-object => *String*

If you want to reflect score as soon as possible, you can fill design object here. c.f)

"node:design:fill".

-value => *Hash reference*

Input value for "-object". String rules are same as "-from" inputted "Apply()". See

"supplementary.txt" in detail.

**Example:**

```
$str-Analyze(
    -method = 'accuracy',
    -option = {
        -object = "node:label",
        -with = $str2,
        -with_object = "node:information:GI",
    }
);
```

**out( filepath, %Options, )**

"out()" method is for outputting network data.

-filetype => *String*

This option defines output file type. "Scalable Vector Graphics", "Graphics Markup Language" and "Simple Interaction File" are supported. You can input here string type value as "svg", "gml" or "sif".

**Follow 4 options are for Graph::Layout::Aesthetic module which determines coordinates of the objects. In detail, see Graph::Layout::Aesthetic document on CPAN or simply type "perldoc Graph::Layout::Aesthetic" on command line.**

knr => *Float*

This option defines node repulsion on spring graph draw algorithm. Default: 1.

kmel => *Float*

This option defines min edge length. Default: 1.

kner => *Float*

This option defines node edge repulsion. Default: 1.

kmei2 => *Float*

This option defines min edge intersect. Default: 0.

**Follow 3 options defines graphical output sizing attributes.**

-width => *Float*

This option defines graphics width. Default: 1000.

-height => *Float*

This option defines graphics height. Default: 1000.

-margin => *Float*

This option defines graphics object margin from border of the field. Default: 100.

**Follows are general options for graphical output.**

-no\_direction => *Boolean*

This option defines drawing the graph network as directional or not. Default: 1 (as no direction.)

-no\_node\_label => *Boolean*

This option defines which display labels or not. Default: 0 (as displaying node labels.)

-no\_edge\_label => *Boolean*

- This option defines which display labels or not. Default: 0 (as displaying edge labels.)
- dual\_arrow\_spreads => *Float*  
This option defines dual directed edges spreads. More makes spread dual arrow. Default: 2.
- rotation => *Float*  
This option defines rotation angle of the network. Default: 0.
- object\_size => *Float*  
This option change each object size magnification ratio. You can use this when the object size is too large for network. This option changes node width, stroke-width and edge stroke-width at once. Default: 1.
- label\_size => *Float*  
This option defines label size same as -object\_size. Default: 1.

**Example:**

```
$str-out('ribosome.svg',  
        -object_size = 4,  
        -label_size = 3,  
        -no_edge_label = 1,  
        );
```

## DIAGNOSTICS

if you input several incapable options/input, eXpanda returns some error messages.

## CONFIGURATION AND ENVIRONMENT

eXpanda requires no configuration files or environment variables.

## DEPENDENCIES

```
Carp  
Graph::Topology::Aesthetic  
SVG  
use LWP::UserAgent;  
use SOAP::Lite;  
use XML::Twig;
```

## INCOMPATIBILITIES

None reported.

## BUGS AND LIMITATIONS

No bugs have been reported.

Please report any bugs or feature requests to [po@sfc.keio.ac.jp](mailto:po@sfc.keio.ac.jp), or through the web interface at <http://medcd.iab.keio.ac.jp/eXpanda/>.

## AUTHOR

Yoshiteru Negishi <[po@sfc.keio.ac.jp](mailto:po@sfc.keio.ac.jp)>

## LICENCE AND COPYRIGHT

Copyright (c) 2006, Yoshiteru Negishi <[po@sfc.keio.ac.jp](mailto:po@sfc.keio.ac.jp)> . All rights reserved.  
This module is free software; you can redistribute it and/or modify it under the same terms as Perl itself.  
See perlartistic.

## DISCLAIMER OF WARRANTY

BECAUSE THIS SOFTWARE IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE SOFTWARE, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE SOFTWARE "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER

EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE SOFTWARE IS WITH YOU. SHOULD THE SOFTWARE PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR, OR CORRECTION.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE SOFTWARE AS PERMITTED BY THE ABOVE LICENCE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE SOFTWARE (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE SOFTWARE TO OPERATE WITH ANY OTHER SOFTWARE), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

