

# Package ‘CytoML’

October 14, 2021

**Type** Package

**Title** A GatingML Interface for Cross Platform Cytometry Data Sharing

**Version** 2.4.0

**Date** 2016-04-15

**Author** Mike Jiang, Jake Wagner

**Maintainer** Mike Jiang <wjiang2@fhcrc.org>

**Description** Uses platform-specific implemenations of the GatingML2.0 standard to exchange gated cytometry data with other software platforms.

**License** file LICENSE

**License\_restricts\_use** yes

**LazyData** TRUE

**Imports** cytolib(>= 2.3.9), flowCore (>= 1.99.10), flowWorkspace (>= 4.1.8), openCyto (>= 1.99.2), XML, data.table, jsonlite, RBGL, Rgraphviz, Biobase, methods, graph, graphics, utils, base64enc, plyr, dplyr, grDevices, methods, ggcyto (>= 1.11.4), yaml, lattice, stats, corpcor, RUnit, tibble, RcppParallel, xml2

**biocViews** ImmunoOncology, FlowCytometry, DataImport, DataRepresentation

**LinkingTo** Rcpp, BH(>= 1.62.0-1), RProtoBufLib, cytolib, Rhdf5lib, RcppArmadillo, RcppParallel(>= 4.4.2-1), flowWorkspace

**Suggests** testthat, flowWorkspaceData, knitr, parallel

**VignetteBuilder** knitr

**RoxygenNote** 7.1.1

**BugReports** <https://github.com/RGLab/CytoML/issues>

**URL** <https://github.com/RGLab/CytoML>

**Collate** 'AllClasses.R' 'GatingSet2cytobank.R' 'GatingSet2flowJo.R' 'RcppExports.R' 'gate-methods.R' 'compensation.R' 'cytobank2GatingSet.R' 'cytobankExperiment.R' 'cytolibml\_bin\_path.R' 'flowJoWorkspace\_Methods.R' 'diva2GatingSet.R' 'flowUtils\_functions.R' 'gatingML.R'

'read.gatingML.cytobank.R' 'graphGML\_methods.R'  
 'helperFunctions.R' 'parameter-methods.R' 'transforms.R'  
 'utils.R' 'writeGatingML.R' 'zzz.R'

**SystemRequirements** xml2, GNU make, C++11

**Encoding** UTF-8

**git\_url** <https://git.bioconductor.org/packages/CytoML>

**git\_branch** RELEASE\_3\_13

**git\_last\_commit** 6e64d69

**git\_last\_commit\_date** 2021-05-19

**Date/Publication** 2021-10-14

## R topics documented:

addCustomInfo . . . . .	3
ce_get_channels . . . . .	4
ce_get_compensations . . . . .	4
ce_get_markers . . . . .	5
ce_get_panels . . . . .	5
ce_get_samples . . . . .	6
ce_get_transformations . . . . .	6
compensate.GatingSet.graphGML-method . . . . .	7
constructTree . . . . .	7
cytobank_experiment-methods . . . . .	8
cytobank_to_gatingset . . . . .	8
CytoML-deprecated . . . . .	9
CytoML.par.init . . . . .	10
CytoML.par.set . . . . .	10
diva_get_samples . . . . .	11
diva_to_gatingset . . . . .	11
diva_workspace-class . . . . .	12
extend . . . . .	13
fj_ws_get_keywords . . . . .	14
fj_ws_get_samples . . . . .	15
fj_ws_get_sample_groups . . . . .	16
flowjo_workspace-class . . . . .	17
gatingset_to_cytobank . . . . .	17
gatingset_to_flowjo . . . . .	18
getChildren.graphGML.character-method . . . . .	20
getCompensationMatrices.graphGML . . . . .	20
getGate.graphGML.character-method . . . . .	21
getNodes.graphGML-method . . . . .	21
getParent.graphGML.character-method . . . . .	22
getTransformations.graphGML . . . . .	22
graphGML-class . . . . .	23
gs_compare_cytobank_counts . . . . .	23

<code>addCustomInfo</code>	3
<code>matchPath</code>	24
<code>open_cytobank_experiment</code>	25
<code>open_diva_xml</code>	25
<code>open_flowjo_xml</code>	26
<code>parse.gateInfo</code>	27
<code>parseWorkspace</code>	27
<code>plot,graphGML,missing-method</code>	31
<code>range.GatingHierarchy</code>	32
<code>read.gatingML.cytobank</code>	33
<code>show,graphGML-method</code>	34
<b>Index</b>	<b>35</b>

---

<code>addCustomInfo</code>	<i>add customInfo nodes to each gate node and add BooleanAndGates</i>
----------------------------	---

---

## Description

add customInfo nodes to each gate node and add BooleanAndGates

## Usage

```
addCustomInfo(root, gs, flowEnv, cytobank.default.scale = TRUE, showHidden)
```

## Arguments

<code>root</code>	the root node of the XML
<code>gs</code>	a GatingSet object
<code>flowEnv</code>	the environment that stores the information parsed by 'read.GatingML'.
<code>cytobank.default.scale</code>	logical flag indicating whether to use the default Cytobank asinhtGml2 settings. Currently it should be set to TRUE in order for gates to be displayed properly in Cytobank because cytobank currently does not parse the global scale settings from GatingML.
<code>showHidden</code>	whether to include the hidden population nodes in the output

## Value

XML root node

---

ce\_get\_channels      *Extract channels from cytobank\_experiment*

---

**Description**

Extract channels from cytobank\_experiment

**Usage**

```
ce_get_channels(x, panel_name = NULL)
```

**Arguments**

x                    A cytobank\_experiment object  
panel\_name        select panel to process

---

ce\_get\_compensations      *Obtain the spillover matrices for the samples in a Cytobank experiment*

---

**Description**

Obtain the spillover matrices for the samples in a Cytobank experiment

**Usage**

```
ce_get_compensations(x)
```

**Arguments**

x                    A cytobank\_experiment object

**Value**

A named list of spillover matrices

---

ce_get_markers	<i>Extract markers from cytobank_experiment</i>
----------------	---

---

**Description**

Extract markers from cytobank\_experiment

**Usage**

```
ce_get_markers(x, panel_name = NULL)
```

**Arguments**

x	A cytobank_experiment object
panel_name	select panel to process

---

ce_get_panels	<i>Obtain counts of the number of samples associated with each marker panel in a Cytobank experiment</i>
---------------	--

---

**Description**

Obtain counts of the number of samples associated with each marker panel in a Cytobank experiment

**Usage**

```
ce_get_panels(x)
```

**Arguments**

x	cytobank_experiment object
---	----------------------------

**Value**

A tibble of panels with sample counts

---

ce_get_samples	<i>Obtain a mapping between the samples and marker panels in a Cytobank experiment</i>
----------------	--

---

**Description**

Obtain a mapping between the samples and marker panels in a Cytobank experiment

**Usage**

```
ce_get_samples(x)
```

**Arguments**

x                    A cytobank\_experiment object

**Value**

A tibble with rows containing sample names and their associated panel names

---

ce_get_transformations	<i>Obtain the transformations associated with each channel in a Cytobank experiment</i>
------------------------	---

---

**Description**

Obtain the transformations associated with each channel in a Cytobank experiment

**Usage**

```
ce_get_transformations(x, panel_name = NULL)
```

**Arguments**

x                    A cytobank\_experiment object  
panel\_name        select panel to process

**Value**

A transformerList object containing transformation objects for each transformed channel

---

compensate, GatingSet, graphGML-method  
*compensate a GatingSet based on the compensation information stored in graphGML object*

---

**Description**

compensate a GatingSet based on the compensation information stored in graphGML object

**Usage**

```
## S4 method for signature 'GatingSet,graphGML'
compensate(x, spillover, ...)
```

**Arguments**

x	GatingSet
spillover	graphGML
...	unused.

**Value**

compensated GatingSet

---

constructTree                    *Reconstruct the population tree from the GateSets*

---

**Description**

Reconstruct the population tree from the GateSets

**Usage**

```
constructTree(flowEnv, gateInfo)
```

**Arguments**

flowEnv	the environment contains the elements parsed by read.gatingML function
gateInfo	the data.frame contains the gate name, fcs filename parsed by parse.gateInfo function

**Value**

a graphNEL represent the population tree. The gate and population name are stored as nodeData in each node.

---

 cytobank\_experiment-methods

*Methods for interacting with cytobank\_experiment objects*


---

### Description

These methods mirror similar accessor methods for the GatingSet class.

### Usage

```
## S4 method for signature 'cytobank_experiment'
markernames(object)

## S4 method for signature 'cytobank_experiment'
colnames(x, do.NULL = "missing", prefix = "missing")

## S4 method for signature 'cytobank_experiment'
sampleNames(object)

## S4 method for signature 'cytobank_experiment'
pData(object)
```

### Arguments

object	A cytobank_experiment object
x	cytobank_experiment
do.NULL, prefix	not used

---

cytobank\_to\_gatingset *A wrapper that parses the gatingML and FCS files (or cytobank\_experiment object) into GatingSet*

---

### Description

A wrapper that parses the gatingML and FCS files (or cytobank\_experiment object) into GatingSet

### Usage

```
## Default S3 method:
cytobank_to_gatingset(x, FCS, trans = NULL, ...)

## S3 method for class 'cytobank_experiment'
cytobank_to_gatingset(x, panel_id = 1, ...)
```



**Arguments**

x	the cytobank_experiment object or the full path of gatingML file
FCS	FCS files to be loaded
trans	a 'transformerList' object to override the transformations from gatingML files. it is typically used by 'cytobank_experiment' parser(i.e. 'cytobank_to_gatingset.cytobank_experiment') to use the scales info recorded in yaml file.
...	other arguments
panel_id	select panel to process

**Value**

a GatingSet

**Examples**

```
## Not run:
acsfile <- system.file("extdata/cytobank_experiment.acs", package = "CytoML")
ce <- open_cytobank_experiment(acsfile)
xmlfile <- ce$gatingML
fcsFiles <- list.files(ce$fcsdir, full.names = TRUE)
gs <-< cytobank_to_gatingset(xmlfile, fcsFiles)
library(ggcyto)
autoplot(gs[[1]])

## End(Not run)
```

---

CytoML-deprecated

*Deprecated functions in package CytoML.*

---

**Description**

GatingSet2cytobank → [gatingset\\_to\\_cytobank](#)  
 GatingSet2flowJo → [gatingset\\_to\\_flowjo](#)  
 cytobankExperiment → [open\\_cytobank\\_experiment](#)  
 cytobank2GatingSet → [cytobank\\_to\\_gatingset](#)  
 parseWorkspace → [flowjo\\_to\\_gatingset](#)  
 getKeywords → [fj\\_ws\\_get\\_keywords](#)  
 getSamples → [fj\\_ws\\_get\\_samples](#)  
 getSampleGroups → [fj\\_ws\\_get\\_sample\\_groups](#)  
 openDiva → [open\\_diva\\_xml](#)  
 parseWorkspace → [diva\\_to\\_gatingset](#)

---

CytoML.par.init	<i>workspace version is parsed from xml node '/Workspace/version' in flowJo workspace and matched with this list to dispatch to the one of the three workspace parsers</i>
-----------------	--

---

### Description

workspace version is parsed from xml node '/Workspace/version' in flowJo workspace and matched with this list to dispatch to the one of the three workspace parsers

### Usage

CytoML.par.init()

---

CytoML.par.set	<i>CytoML.par.set sets a set of parameters in the CytoML package namespace.</i>
----------------	---

---

### Description

CytoML.par.get gets a set of parameters in the CytoML package namespace.

### Usage

CytoML.par.set(name, value)

CytoML.par.get(name = NULL)

### Arguments

name	The name of a parameter category to get or set.
value	A named list of values to set for category name or a list of such lists if name is missing.

### Details

It is currently used to add/remove the support for a specific flowJo versions (parsed from xml node '/Workspace/version' in flowJo workspace)

**Examples**

```
#get the flowJo versions currently supported
old <- CytoML.par.get("flowJo_versions")

#add the new version
old[["win"]] <- c(old[["win"]], "1.7")
CytoML.par.set("flowJo_versions", old)

CytoML.par.get("flowJo_versions")
```

---

diva_get_samples	<i>Get a table of samples from a FACSDiva workspace</i>
------------------	---

---

**Description**

Return a data.frame of sample group information from a FACSDiva workspace

**Usage**

```
diva_get_samples(x)
diva_get_sample_groups(x)
```

**Arguments**

x                    A diva\_workspace

**Value**

A data.frame with columns tub, name, and specimen

---

diva_to_gatingset	<i>Parse a FACSDiva Workspace</i>
-------------------	-----------------------------------

---

**Description**

Function to parse a FACSDiva Workspace, generate a GatingHierarchy or GatingSet object, and associated flowCore gates.

**Usage**

```
diva_to_gatingset(
  obj,
  name = NULL,
  subset = NULL,
  path = obj@path,
  worksheet = c("normal", "global"),
  swap_cols = list(`FSC-H` = "FSC-W", `SSC-H` = "SSC-W"),
  verbose = FALSE,
  ...
)
```

**Arguments**

obj	diva_workspace
name	sample group to be parsed, either numeric index or the group name
subset	samples to be imported. either numeric index or the sample name. Default is NULL, which imports all samples.
path	the FCS data path
worksheet	select worksheet to import. either "normal" or "global"
swap_cols	diva seems to swap some data cols during importing fcs to experiments this argument provide a list to tell the parser which cols to be swapped default is list('FSC-H' = 'FSC-W', 'SSC-H' = 'SSC-W')
verbose	whether print more messages during the parsing
...	other arguments

---

diva\_workspace-class *An R representation of a BD FACSDiva workspace*

---

**Description**

Inherited from [flowjo\\_workspace-class](#)

**Slots**

version: Object of class "character". The version of the XML workspace.  
 file: Object of class "character". The file name.  
 .cache: Object of class "environment". An environment for internal use.  
 path: Object of class "character". The path to the file.  
 doc: Object of class "XMLInternalDocument". The XML document object.  
 options: Object of class "integer". The XML parsing options passed to [xmlTreeParse](#).

**See Also**

[GatingSet](#) [GatingHierarchy](#)

---

extend	<i>extend the gate to the minimum and maximum limit of both dimensions based on the bounding information.</i>
--------	---

---

### Description

It is equivalent to the behavior of shifting the off-scale boundary events into the gate boundary that is described in bounding transformation section of gatingML standard.

### Usage

```

extend(
  gate,
  bound,
  data.range = NULL,
  plot = FALSE,
  limits = c("original", "extended")
)

## S3 method for class 'polygonGate'
extend(
  gate,
  bound,
  data.range = NULL,
  plot = FALSE,
  limits = c("original", "extended")
)

## S3 method for class 'rectangleGate'
extend(gate, ...)

## S3 method for class 'ellipsoidGate'
extend(gate, ...)

```

### Arguments

gate	a flowCore filter/gate
bound	numeric matrix representing the bounding information parsed from gatingML. Each row corresponds to a channel. rownames should be the channel names. colnames should be c("min", "max")
data.range	numeric matrix specifying the data limits of each channel. It is used to set the extended value of vertices and must have the same structure as 'bound'. when it is not supplied, c(-.Machine\$integer.max, -.Machine\$integer.max) is used.
plot	whether to plot the extended polygon.
limits	character whether to plot in "extended" or "original" gate limits. Default is "original".
...	other arguments

**Details**

The advantage of extending gates instead of shifting data are two folds: 1. Avoid the extra computation each time applying or plotting the gates 2. Avoid changing the data distribution caused by adding the gates

Normally this function is not used directly by user but invoked when parsing GatingML file exported from Cytobank.

**Value**

a flowCore filter/gate

**Examples**

```
library(flowCore)
sqrcut <- matrix(c(300,300,600,600,50,300,300,50),ncol=2,nrow=4)
colnames(sqrcut) <- c("FSC-H","SSC-H")
pg <- polygonGate(filterId="nonDebris", sqrcut)
pg
bound <- matrix(c(100,3e3,100,3e3),
  byrow = TRUE, nrow = 2,
  dimnames = list(c("FSC-H", "SSC-H"),
    c("min", "max")))
bound
pg.extened <- extend(pg, bound, plot = TRUE)
```

---

fj\_ws\_get\_keywords      *Get Keywords*

---

**Description**

Retrieve keywords associated with a workspace

**Usage**

```
fj_ws_get_keywords(obj, y, ...)
```

**Arguments**

obj	A flowjo_workspace
y	ccharacter or numeric specifying the sample name or sample ID
...	other arguments sampNloc a character the location where the sample name is specified. See parseWorkspace for more details.

**Details**

Retrieve a list of keywords from a flowjo\_workspace

**Value**

A list of keyword - value pairs.

**Examples**

```
## Not run:
d<-system.file("extdata",package="flowWorkspaceData")
wsfile<-list.files(d,pattern="manual.xml",full=TRUE)
ws <- open_flowjo_xml(wsfile)

fj_ws_get_samples(ws)
res <- try(fj_ws_get_keywords(ws,"CytoTrol_CytoTrol1.fcs"), silent = TRUE)
print(res[[1]])
fj_ws_get_keywords(ws, 1)

## End(Not run)
```

---

fj_ws_get_samples	<i>Get a list of samples from a flowJo workspace</i>
-------------------	--

---

**Description**

Return a data frame of samples contained in a flowJo workspace

**Usage**

```
fj_ws_get_samples(x, group_id = NULL)
```

**Arguments**

x	A flowjo_workspace
group_id	integer specifies the group from which samples are returned

**Details**

The samples with 0 populations are excluded. Returns a data . frame of samples in the flowjo\_workspace, including their sampleID, name

**Value**

A data . frame with columns sampleID, name

**Examples**

```
## Not run:
#ws is a flowjo_workspace
fj_ws_get_samples(ws);

## End(Not run)
```

---

`fj_ws_get_sample_groups`*Get a table of sample groups from a flowJo workspace*

---

**Description**

Return a data frame of sample group information from a flowJo workspace

**Usage**

```
fj_ws_get_sample_groups(x)
```

**Arguments**

x                    A flowjo\_workspace object.

**Details**

Note that the samples with 0 populations are also included (since count populations requires traversing xml for all samples thus can be expensive) Returns a table of samples and groups defined in the flowJo workspace

**Value**

A data.frame containing the groupName, groupID, and sampleID for each sample in the workspace. Each sample may be associated with multiple groups.

**See Also**

[flowjo\\_workspace-class flowjo\\_to\\_gatingset](#)

**Examples**

```
## Not run:  
#ws is a flowjo_workspace  
fj_ws_get_sample_groups(ws);  
  
## End(Not run)
```



---

`flowjo_workspace-class`*An R representation of a flowJo workspace.*

---

**Description**

Objects can be created by calls of the form `new("flowjo_workspace.xml", ...)`.

**Slots**

`doc`: Object of class "externalptr".

**See Also**

[GatingSet](#) [GatingHierarchy](#)

**Examples**

```
require(flowWorkspaceData)
d<-system.file("extdata",package="flowWorkspaceData")
wsfile<-list.files(d,pattern="A2004Analysis.xml",full=TRUE)
ws <- open_flowjo_xml(wsfile);
ws
fj_ws_get_samples(ws)
```

---

`gatingset_to_cytobank` *Convert a GatingSet to a Cytobank-compatible gatingML*

---

**Description**

this function retrieves the gates from `GatingSet` and writes a customized `GatingML-2.0` file that can be imported into `cytobank`.

**Usage**

```
gatingset_to_cytobank(
  gs,
  outFile,
  showHidden = FALSE,
  cytobank.default.scale = TRUE,
  ...
)
```

**Arguments**

<code>gs</code>	a GatingSet object
<code>outFile</code>	a file name
<code>showHidden</code>	whether to include the hidden population nodes in the output
<code>cytobank.default.scale</code>	logical flag indicating whether to use the default Cytobank <code>asinhtGml2</code> settings. Currently it should be set to <code>TRUE</code> in order for gates to be displayed properly in Cytobank because cytobank currently does not parse the global scale settings from GatingML.
<code>...</code>	<code>rescale.gate</code> default is <code>TRUE</code> . which means the gate is rescaled to the new scale that is understandable by cytobank. It is recommended not to change this behavior unless user wants to export to a gatingML file used for other purpose other than being imported into cytobank.

**Details**

The process can be divided into four steps: 1. Read in gate geometry, compensation and transformation from `gatingSet` 2. Rescale gate boundaries with `flowjo_biexp()` so gates can be displayed properly in Cytobank 3. Save gates and hierarchy structure to R environment 4. Write environment out to gatingML using `write.GatingML()`

**Value**

nothing

**Examples**

```
library(flowWorkspace)

dataDir <- system.file("extdata", package="flowWorkspaceData")
gs <- load_gs(list.files(dataDir, pattern = "gs_manual", full = TRUE))

gs_pop_remove(gs, "CD8")

#output to cytobank
outFile <- tempfile(fileext = ".xml")
gatingset_to_cytobank(gs, outFile) #type by default is 'cytobank'
```

---

`gatingset_to_flowjo`    *Convert a GatingSet to flowJo workspace*

---

**Description**

It is a R wrapper for the docker app (<https://hub.docker.com/r/rglab/gs-to-flowjo>)

## Usage

```
gatingset_to_flowjo(gs, outFile, showHidden = FALSE, docker_img = NULL, ...)
```

## Arguments

gs	a GatingSet object or a folder contains the GatingSet archive (generated by previous <code>save_gs</code> call)
outFile	the workspace file path to write
showHidden	whether to export hidden gates. Default is FALSE
docker_img	the docker image that does the actual work
...	other arguments passed to <code>save_gs</code>

## Details

Docker images for `gatingset_to_flowjo` will be maintained at <https://gallery.ecr.aws/x4k5d9i7/cytoverse/gs-to-wsp>

```
docker pull public.ecr.aws/x4k5d9i7/cytoverse/gs-to-wsp:latest
```

## Value

nothing

## Examples

```
## Not run:
library(flowWorkspace)

path <- system.file("extdata", package="flowWorkspaceData")
gs_path <- list.files(path, pattern = "gs_manual", full = TRUE)
gs <- load_gs(gs_path)

#output to flowJo
outFile <- tempfile(fileext = ".wsp")
gatingset_to_flowjo(gs, outFile)

#or directly use the archive as the input (to avoid the extra copying inside of the wrapper)
gatingset_to_flowjo(gs_path, outFile)

## End(Not run)
```

getChildren, graphGML, character-method  
*get children nodes*

---

**Description**

get children nodes

**Usage**

```
## S4 method for signature 'graphGML,character'  
getChildren(obj, y)
```

**Arguments**

obj	graphGML
y	character parent node path

**Value**

a graphNEL node

**Examples**

```
## Not run:  
g <- read.gatingML.cytobank(xmlfile)  
getChildren(g, "GateSet_722326")  
getParent(g, "GateSet_722326")  
  
## End(Not run)
```

---

getCompensationMatrices.graphGML  
*Extract compensation from graphGML object.*

---

**Description**

Extract compensation from graphGML object.

**Usage**

```
## S3 method for class 'graphGML'  
getCompensationMatrices(x)
```

**Arguments**

x	graphGML
---	----------

**Value**

compensation object or "FCS" when compensation comes from FCS keywords

---

getGate,graphGML,character-method  
*get gate from the node*

---

**Description**

get gate from the node

**Usage**

```
## S4 method for signature 'graphGML,character'
getGate(obj, y)
```

**Arguments**

obj	graphGML
y	character node path

**Value**

the gate information associated with the node

---

getNodes,graphGML-method  
*get nodes from graphGML object*

---

**Description**

get nodes from graphGML object

**Usage**

```
## S4 method for signature 'graphGML'
getNodes(x, y, order = c("default", "bfs", "dfs", "tsort"), only.names = TRUE)
```

**Arguments**

x	graphGML
y	character node index. When missing, return all the nodes
order	character specifying the order of nodes. options are "default", "bfs", "dfs", "tsort"
only.names	logical specifying whether user wants to get the entire nodeData or just the name of the population node

**Value**

It returns the node names and population names by default. Or return the entire nodeData associated with each node.

**Examples**

```
## Not run:  
g <- read.gatingML.cytobank(xmlfile)  
getNodeNames(g)  
getNodeNames(g, only.names = FALSE)  
  
## End(Not run)
```

---

getParent, graphGML, character-method  
*get parent nodes*

---

**Description**

get parent nodes

**Usage**

```
## S4 method for signature 'graphGML,character'  
getParent(obj, y)
```

**Arguments**

obj	graphGML
y	character child node path

**Value**

a graphNEL node

---

getTransformations.graphGML  
*Extract transformations from graphGML object.*

---

**Description**

Extract transformations from graphGML object.

**Usage**

```
## S3 method for class 'graphGML'
getTransformations(x, ...)
```

**Arguments**

```
x          graphGML
...        not used
```

**Value**

transformerList object

---

graphGML-class	<i>A graph object returned by 'read.gatingML.cytobank' function.</i>
----------------	--

---

**Description**

Each node corresponds to a population(or GateSet) defined in gatingML file. The actual gate object (both global and tailored gates) is associated with each node as nodeData. Compensation and transformations are stored in graphData slot.

**Details**

The class simply extends the graphNEL class and exists for the purpose of method dispatching.

---

gs_compare_cytobank_counts	<i>compare the counts to cytobank's exported csv so that the parsing result can be verified.</i>
----------------------------	--

---

**Description**

compare the counts to cytobank's exported csv so that the parsing result can be verified.

**Usage**

```
gs_compare_cytobank_counts(
  gs,
  file,
  id.vars = c("FCS Filename", "population"),
  ...
)
```

**Arguments**

gs	parsed GatingSet
file	the stats file (contains the populatio counts) exported from cytobank.
id.vars	either "population" or "FCS filename" that tells whether the stats file format is one population per row or FCS file per row.
...	arguments passed to data.table::fread function

**Value**

a data.table (in long format) that contains the counts from openCyto and Cytobank side by side.

**Examples**

```
acsfile <- system.file("extdata/cytobank_experiment.acs", package = "CytoML")
ce <- open_cytobank_experiment(acsfile)
gs <- cytobank_to_gatingset(ce)
## verify the stats are correct
statsfile <- ce$attachments[1]
dt_merged <- gs_compare_cytobank_counts(gs, statsfile, id.vars = "population", skip = "FCS Filename")
all.equal(dt_merged[, count.x], dt_merged[, count.y], tol = 5e-4)
```

---

matchPath

*Given the leaf node, try to find out if a collection of nodes can be matched to a path in a graph(tree) by the bottom-up searching*

---

**Description**

Given the leaf node, try to find out if a collection of nodes can be matched to a path in a graph(tree) by the bottom-up searching

**Usage**

```
matchPath(g, leaf, nodeSet)
```

**Arguments**

g	graphNEL
leaf	the name of leaf(terminal) node
nodeSet	a set of node names

**Value**

TRUE if path is found, FALSE if not path is matched.



---

open\_cytobank\_experiment  
*Construct a cytobank\_experiment object from ACS file*

---

**Description**

Construct a cytobank\_experiment object from ACS file

**Usage**

```
open_cytobank_experiment(acs, exdir = tempfile())
```

**Arguments**

acs	ACS file exported from Cytobank
exdir	the directory to extract files to

**Value**

cytobank\_experiment object

---

open\_diva\_xml      *open Diva xml workspace*

---

**Description**

open Diva xml workspace

**Usage**

```
open_diva_xml(file, options = 0, ...)
```

**Arguments**

file	xml file
options	argument passed to <a href="#">xmlTreeParse</a>
...	arguments passed to <a href="#">xmlTreeParse</a>

**Value**

a diva\_workspace object

**Examples**

```
## Not run:
library(flowWorkspace)
library(CytoML)
ws <- open_diva_xml(system.file('extdata/diva/PE_2.xml', package = "flowWorkspaceData"))
ws
diva_get_sample_groups(ws)
gs <- diva_to_gatingset(ws, name = 2, subset = 1)
sampleNames(gs)
gs_get_pop_paths(gs)
plotGate(gs[[1]])

## End(Not run)
```

---

open_flowjo_xml	<i>Open/Close a flowJo workspace</i>
-----------------	--------------------------------------

---

**Description**

Open a flowJo workspace and return a `flowjo_workspace` object. Close a `flowjo_workspace`, destroying the internal representation of the XML document, and freeing the associated memory.

**Usage**

```
open_flowjo_xml(file, options = 0, sample_names_from = "keyword", ...)
```

**Arguments**

<code>file</code>	Full path to the XML flowJo workspace file.
<code>options</code>	xml parsing options passed to <code>xmlTreeParse</code> . See <a href="http://xmlsoft.org/html/libxml-parser.html#xmlParserOption">http://xmlsoft.org/html/libxml-parser.html#xmlParserOption</a> for details.
<code>sample_names_from</code>	character specifying where in the XML workspace file to obtain the sample names, either "keyword" for the included \$FIL keyword for each sample, or "sampleNode" for the name of the sample node
<code>...</code>	not used

**Details**

Open an XML flowJo workspace file and return a `flowjo_workspace` object. The workspace is represented using a `XMLInternalDocument` object. Close a `flowJoWorkspace` after finishing with it. This is necessary to explicitly clean up the C-based representation of the XML tree. (See the XML package).

**Value**

a `flowjo_workspace` object.

**Examples**

```
## Not run:
file<-"myworkspace.xml"
ws<-open_flowjo_xml(file);
ws

## End(Not run)
```

---

parse.gateInfo	<i>Parse the cytobank custom_info for each gate</i>
----------------	---

---

**Description**

Fcs filename and gate name stored in 'custom\_info' element are beyond the scope of the gatingML standard and thus not covered by the default 'read.gatingML'.

**Usage**

```
parse.gateInfo(file, ...)
```

**Arguments**

file	xml file path
...	additional arguments passed to the handlers of 'xmlTreeParse'

**Value**

a data.frame that contains three columns: id (gateId), name (gate name), fcs (fcs\_file\_filename).

---

parseWorkspace	<i>Parse a flowJo Workspace</i>
----------------	---------------------------------

---

**Description**

Function to parse a flowJo Workspace, generate a GatingHierarchy or GatingSet object, and associated flowCore gates. The data are not loaded or acted upon until an explicit call to recompute() is made on the GatingHierarchy objects in the GatingSet.

**Usage**

```

parseWorkspace(obj, ...)

## S4 method for signature 'flowjo_workspace'
parseWorkspace(obj, ...)

flowjo_to_gatingset(
  ws,
  name = NULL,
  subset = list(),
  execute = TRUE,
  path = "",
  cytoset = NULL,
  backend_dir = tempdir(),
  backend = get_default_backend(),
  includeGates = TRUE,
  additional.keys = "$TOT",
  additional.sampleID = FALSE,
  keywords = character(),
  keywords.source = "XML",
  keyword.ignore.case = FALSE,
  extend_val = 0,
  extend_to = -4000,
  channel.ignore.case = FALSE,
  leaf.bool = TRUE,
  include_empty_tree = FALSE,
  skip_faulty_gate = FALSE,
  compensation = NULL,
  transform = TRUE,
  fcs_file_extension = ".fcs",
  greedy_match = FALSE,
  mc.cores = 1,
  ...
)

```

**Arguments**

obj	flowjo_workspace
...	Additional arguments to be passed to FCS parser
ws	A flowjo_workspace to be parsed.
name	numeric or character. The name or index of the group of samples to be imported. If NULL, the groups are printed to the screen and one can be selected interactively. Usually, multiple groups are defined in the flowJo workspace file.
subset	numeric vector specifying the subset of samples in a group to import. Or a character specifying the FCS filenames to be imported. Or an expression to be passed to 'subset' function to filter samples by 'pData' (Note that the columns

	referred by the expression must also be explicitly specified in 'keywords' argument)
execute	TRUE FALSE a logical specifying if the gates, transformations, and compensation should be immediately calculated after the flowJo workspace have been imported. TRUE by default.
path	either a character scalar or data.frame. When character, it is a path to the fcs files that are to be imported. The code will search recursively, so you can point it to a location above the files. When it is a data.frame, it is expected to contain two columns: 'sampleID' and 'file', which is used as the mapping between 'sampleID' and FCS file (absolute) path. When such mapping is provided, the file system searching is avoided.
cytaset	a cytaset object that provides the alternative data source other than FCS files. It is useful sometime to preprocess the raw fcs files (e.g. standardize channels using cytoqc package) and then directly use them for flowJo parsing. when cytaset is provided, path argument is ignored.
includeGates	logical Should gates be imported, or just the data with compensation and transformation?
additional.keys	character vector: The keywords (parsed from FCS header) to be combined(concatenated with "_") with FCS filename to uniquely identify samples. Default is '\$TOT' (total number of cells) and more keywords can be added to make this GUID.
additional.sampleID	boolean: A boolean specifying whether to include the flowJo sample ID in a GUID to uniquely identify samples. This can be helpful when the filename or other keywords are not enough to differentiate between samples. Default is FALSE.
keywords	character vector specifying the keywords to be extracted as pData of GatingSet
keywords.source	character the place where the keywords are extracted from, can be either "XML" or "FCS"
keyword.ignore.case	a logical flag indicates whether the keywords matching needs to be case sensitive.
extend_val	numeric the threshold that determine wether the gates need to be extended. default is 0. It is triggered when gate coordinates are below this value.
extend_to	numeric the value that gate coordinates are extended to. Default is -4000. Usually this value will be automatically detected according to the real data range. But when the gates needs to be extended without loading the raw data (i.e. execute is set to FALSE), then this hard-coded value is used.
channel.ignore.case	a logical flag indicates whether the colnames(channel names) matching needs to be case sensitive (e.g. compensation, gating..)
leaf.bool	a logical whether to compute the leaf boolean gates. Default is TRUE. It helps to speed up parsing by turning it off when the statistics of these leaf boolean gates are not important for analysis. (e.g. COMPASS package will calculate them by itself.) If needed, they can be calculated by calling recompute method at later stage.

<code>include_empty_tree</code>	a logical whether to include samples that don't have gates.
<code>skip_faulty_gate</code>	a logical whether to skip the faulty gates so that the parser can still process the rest of gating tree.
<code>compensation</code>	a compensation object, matrix or data.frame or a list of these objects that allow the customized compensation () to be used instead of the one specified in flowJo workspace or FCS file. When it is a list, its names is supposed to be matched to sample guids (Default is the fcs filename suffixed by \$TOT. See "additional.keys" arguments for details of guids) When some of the samples don't have the external compensations matched, it will fall back to the flowJo xml or FCS looking for the compensation matrix.
<code>transform</code>	logical to enable/disable transformation of gates and data. Default is TRUE. It is mainly for debug purpose (when the raw gates need to be parsed.), and only valid when execute is FALSE.
<code>fcs_file_extension</code>	default is ".fcs"
<code>greedy_match</code>	logical: By default, if flowjo_to_gatingset finds multiple FCS files matching a sample by total event count as well as sampleID and/or keywords specified by additional.keys and additional.sampleID, it will return an error listing the duplicate files. If greedy_match is TRUE, the method will simply take the first file with either filename or \$FIL keyword matching the sample name and having the correct number of events.
<code>mc.cores</code>	numeric the number of threads to pass to the C++ parser to run in parallel
<code>h5_dir</code>	the path to write h5 data

### Details

A `flowjo_workspace` is generated with a call to `open_flowjo_xml()`, passing the name of the xml workspace file. This returns a `flowjo_workspace`, which can be parsed using the `flowjo_to_gatingset()` method. The function can be called non-interactively by passing the index or name of the group of samples to be imported via `flowjo_to_gatingset(obj,name=x)`, where `x` is either the numeric index, or the name. The `subset` argument allows one to select a set of files from the chosen sample group. The routine will take the intersection of the files in the sample group, the files specified in `subset` and the files available on disk, and import them.

### Value

a `GatingSet`, which is a wrapper around a list of `GatingHierarchy` objects, each representing a single sample in the workspace. The `GatingHierarchy` objects contain `graphNEL` trees that represent the gating hierarchy of each sample. Each node in the `GatingHierarchy` has associated data, including the population counts from flowJo, the parent population counts, the `flowCore` gates generated from the flowJo workspace gate definitions. Data are not yet loaded or acted upon at this stage. To execute the gating of each data file, a call to `execute()` must be made on each `GatingHierarchy` object in the `GatingSet`. This is done automatically by default, and there is no more reason to set this argument to FALSE.

**See Also**

[fj\\_ws\\_get\\_sample\\_groups.GatingSet](#)

**Examples**

```
## Not run:
#f is a xml file name of a flowJo workspace
ws <- open_flowjo_xml(f)
#parse the second group
gs <- flowjo_to_gatingset(ws, name = 2); #assume that the fcs files are under the same folder as workspace

gs <- flowjo_to_gatingset(ws, name = 4
                          , path = dataDir      #specify the FCS path
                          , subset = "CytoTrol_CytoTrol_1.fcs") #subset the parsing by FCS filename

gs <- flowjo_to_gatingset(ws, path = dataDir, name = 4
                          , keywords = c("PATIENT ID", "SAMPLE ID", "$TOT", "EXPERIMENT NAME") #tell the parser to extract keywords
                          , keywords.source = "XML" # keywords are extracted from xml workspace (alternatively can be set to "FCS")
                          , additional.keys = c("PATIENT ID") #use additional keywords together with FCS filename to uniquely identify gates
                          , execute = F) # parse workspace without the actual gating (can save time if just want to get the info)

#subset by pData (extracted from keywords)
gs <- flowjo_to_gatingset(ws, path = dataDir, name = 4
                          , subset = `TUBE NAME` %in% c("CytoTrol_1", "CytoTrol_2")
                          , keywords = "TUBE NAME")

#override the default compensation defined in xml with the customized compensations
gs <- flowjo_to_gatingset(ws, name = 2, compensation = comps); #comp is either a compensation object or a list of compensation objects

## End(Not run)
```

---

plot,graphGML,missing-method

*plot the population tree stored in graphGML.*

---

**Description**

The node with dotted order represents the population that has tailored gates (sample-specific gates) defined.

**Usage**

```
## S4 method for signature 'graphGML,missing'
plot(x, y = "missing", label = c("popName", "gateName"))
```

**Arguments**

x	a graphNEL generated by constructTree function
y	not used
label	specifies what to be displayed as node label. Can be either 'popName' (population name parsed from GateSets) or 'gateName' (the name of the actual gate associated with each node)

**Value**

nothing

**Examples**

```
## Not run:
g <- read.gatingML.cytobank(xmlfile)
plot(g)

## End(Not run)
```

---

range.GatingHierarchy *the parameter range from the flow data associated with GatingHierarchy*

---

**Description**

the parameter range from the flow data associated with GatingHierarchy

**Usage**

```
## S3 method for class 'GatingHierarchy'
range(..., na.rm = FALSE, type = c("instrument", "data"), raw.scale = FALSE)
```

**Arguments**

...	GatingHierarchy object
na.rm	not used
type	character of "instrument" or "data" indicating whether to retrieve the instrument or the actual data range
raw.scale	logical whether convert the range from transformed scale to raw scale

**Value**

matrix



## Examples

```
## Not run:
range(gh, type = "data")#return data range
range(gh) #return instrument range
range(gh, raw.scale = TRUE) #inverse transform the range to the raw scale

## End(Not run)
```

---

read.gatingML.cytobank

*Parser for gatingML exported by Cytobank*

---

## Description

The Default parser (read.gatingML) does not parse the population tree as well as the custom information from cytobank. (e.g. gate name, fcs filename).

## Usage

```
read.gatingML.cytobank(file, ...)
```

## Arguments

file	Gating-ML XML file
...	additional arguments passed to the handlers of 'xmlTreeParse'

## Value

a graphGML that represents the population tree. The gate and population name are stored in node-Data of each node. Compensation and transformations are stored in graphData.

## Examples

```
## Not run:
g <- read.gatingML.cytobank(xml) #parse the population tree
#plot(g) #visualize it

## End(Not run)
```

---

show,graphGML-method    *show method for graphGML*

---

**Description**

show method for graphGML

**Usage**

```
## S4 method for signature 'graphGML'  
show(object)
```

**Arguments**

object            graphGML

**Value**

nothing

# Index

addCustomInfo, [3](#)

ce\_get\_channels, [4](#)  
ce\_get\_compensations, [4](#)  
ce\_get\_markers, [5](#)  
ce\_get\_panels, [5](#)  
ce\_get\_samples, [6](#)  
ce\_get\_transformations, [6](#)  
colnames, cytobank\_experiment-method  
(cytobank\_experiment-methods),  
[8](#)  
compensate, GatingSet, graphGML-method,  
[7](#)  
constructTree, [7](#)  
cytobank2GatingSet  
(cytobank\_to\_gatingset), [8](#)  
cytobank\_experiment-methods, [8](#)  
cytobank\_to\_gatingset, [8](#), [9](#)  
cytobankExperiment  
(open\_cytobank\_experiment), [25](#)  
CytoML-deprecated, [9](#)  
CytoML.par.get (CytoML.par.set), [10](#)  
CytoML.par.init, [10](#)  
CytoML.par.set, [10](#)

diva\_get\_sample\_groups  
(diva\_get\_samples), [11](#)  
diva\_get\_samples, [11](#)  
diva\_to\_gatingset, [9](#), [11](#)  
diva\_workspace-class, [12](#)

extend, [13](#)

fj\_ws\_get\_keywords, [9](#), [14](#)  
fj\_ws\_get\_sample\_groups, [9](#), [16](#), [31](#)  
fj\_ws\_get\_samples, [9](#), [15](#)  
flowjo\_to\_gatingset, [9](#), [16](#)  
flowjo\_to\_gatingset (parseWorkspace), [27](#)  
flowjo\_workspace-class, [12](#), [17](#)

GatingHierarchy, [12](#), [17](#)

GatingSet, [12](#), [17](#), [31](#)  
GatingSet2cytobank  
(gatingset\_to\_cytobank), [17](#)  
GatingSet2flowJo (gatingset\_to\_flowjo),  
[18](#)  
gatingset\_to\_cytobank, [9](#), [17](#)  
gatingset\_to\_flowjo, [9](#), [18](#)  
getChildren, graphGML, character-method,  
[20](#)  
getCompensationMatrices.graphGML, [20](#)  
getGate, graphGML, character-method, [21](#)  
getKeywords (fj\_ws\_get\_keywords), [14](#)  
getNodes, graphGML-method, [21](#)  
getParent, graphGML, character-method,  
[22](#)  
getSampleGroups  
(fj\_ws\_get\_sample\_groups), [16](#)  
getSamples (fj\_ws\_get\_samples), [15](#)  
getTransformations.graphGML, [22](#)  
graphGML-class, [23](#)  
gs\_compare\_cytobank\_counts, [23](#)

markernames, cytobank\_experiment-method  
(cytobank\_experiment-methods),  
[8](#)

matchPath, [24](#)

open\_cytobank\_experiment, [9](#), [25](#)  
open\_diva\_xml, [9](#), [25](#)  
open\_flowjo\_xml, [26](#)  
openDiva (open\_diva\_xml), [25](#)

parse.gateInfo, [27](#)  
parseWorkspace, [27](#)  
parseWorkspace, diva\_workspace-method  
(diva\_to\_gatingset), [11](#)  
parseWorkspace, flowjo\_workspace-method  
(parseWorkspace), [27](#)  
pData, cytobank\_experiment-method  
(cytobank\_experiment-methods),  
[8](#)

plot, graphGML, missing-method, [31](#)  
print.cytobank\_experiment  
    (cytobank\_experiment-methods),  
    [8](#)  
  
range.GatingHierarchy, [32](#)  
read.gatingML.cytobank, [33](#)  
  
sampleNames, cytobank\_experiment-method  
    (cytobank\_experiment-methods),  
    [8](#)  
save\_gs, [19](#)  
show, diva\_workspace-method  
    (diva\_workspace-class), [12](#)  
show, flowjo\_workspace-method  
    (flowjo\_workspace-class), [17](#)  
show, graphGML-method, [34](#)  
  
xmlTreeParse, [12](#), [25](#), [26](#)