

# Package ‘CyFj11’

October 29, 2026

**Type** Package

**Title** Parse FlowJo V11 Workspace Files and Export to FlowJo V10

**Version** 0.99.113

**Description** Provides functions to parse FlowJo v11 workspace files (.flowjo) and extract gating information, populations, and associated FCS files. Includes utilities for exporting data to FlowJo v10 workspace format. The package enables bidirectional data flow between FlowJo and R/Bioconductor, facilitating reproducible analysis workflows.

**License** MIT + file LICENSE

**Encoding** UTF-8

**Depends** R (>= 4.6.0)

**Imports** jsonlite, dplyr, flowWorkspace, flowCore, readr, rlang, tibble, tidyr, xml2, methods, scales, magrittr, stats, utils

**Suggests** testthat (>= 3.0.0), Biobase, CytoML, FlowSOM, flowWorkspaceData, ggcyto, knitr, mockery, BiocStyle, qpdf, rmarkdown

**VignetteBuilder** knitr

**biocViews** FlowCytometry, DataImport, Infrastructure, Preprocessing

**URL** <https://github.com/C3BI-pasteur-fr/CyFj11>

**BugReports** <https://github.com/C3BI-pasteur-fr/CyFj11/issues>

**Roxygen** list(markdown = TRUE)

**Config/roxygen2/version** 8.1.0

**Config/testthat/edition** 3

**git\_url** <https://git.bioconductor.org/packages/CyFj11>

**git\_branch** devel

**git\_last\_commit** ab0d40a

**git\_last\_commit\_date** 2026-09-15

**Repository** Bioconductor 3.24

**Date/Publication** 2026-10-28

**Author** Bernd Jagla [aut, cre] (ORCID: <<https://orcid.org/0000-0002-7696-0484>>, affiliation: Institut Pasteur, scBiomarkers UTechS & Biostatistics Hub),  
 Slobodan Culina [aut] (affiliation: Institut Pasteur, scBiomarkers UTechS),  
 Francois Le Guerroue [aut] (affiliation: Institut Pasteur, scBiomarkers UTechS),  
 Esma Karkeni [aut] (affiliation: Institut Pasteur, scBiomarkers UTechS),  
 Milena Hasan [aut] (affiliation: Institut Pasteur, scBiomarkers UTechS)

**Maintainer** Bernd Jagla <bernd.jagla@pasteur.fr>

## Contents

|                                           |    |
|-------------------------------------------|----|
| CyFjl1-package . . . . .                  | 4  |
| add_populations_to_gatingset . . . . .    | 5  |
| add_population_node . . . . .             | 5  |
| adjust_gate_transformations . . . . .     | 7  |
| apply_extension . . . . .                 | 8  |
| apply_param_mapping . . . . .             | 9  |
| apply_transforms_to_gate . . . . .        | 9  |
| archive . . . . .                         | 10 |
| build_gating_tree . . . . .               | 11 |
| build_sample_keywords . . . . .           | 11 |
| build_spillover_matrix_xml . . . . .      | 11 |
| check_wsp_csv_coverage . . . . .          | 12 |
| compensation . . . . .                    | 12 |
| conversion . . . . .                      | 13 |
| convert_boolean_gate . . . . .            | 13 |
| convert_boolean_to_flowjo10 . . . . .     | 13 |
| convert_ellipse_gate . . . . .            | 14 |
| convert_ellipsoid_to_flowjo10 . . . . .   | 14 |
| convert_flowjo_gate . . . . .             | 15 |
| convert_gate_to_flowjo10_format . . . . . | 15 |
| convert_polygon_gate . . . . .            | 16 |
| convert_polygon_to_flowjo10 . . . . .     | 16 |
| convert_quadrant_gate . . . . .           | 16 |
| convert_range_gate . . . . .              | 17 |
| convert_rectangle_gate . . . . .          | 17 |
| convert_rectangle_to_flowjo10 . . . . .   | 17 |
| create_biexponential_transform . . . . .  | 18 |
| create_default_groups_v10 . . . . .       | 18 |
| create_gatingset_from_cytoset . . . . .   | 18 |
| create_linear_transform . . . . .         | 19 |
| create_log_transform . . . . .            | 19 |
| create_sample_names . . . . .             | 20 |
| create_transformation_list . . . . .      | 20 |
| derive_cytometer_attrs . . . . .          | 21 |
| display_to_raw . . . . .                  | 21 |
| emit_transform_xml . . . . .              | 22 |
| export-flowjo10 . . . . .                 | 22 |
| export_flowjo10_workspace . . . . .       | 23 |

|                                                  |    |
|--------------------------------------------------|----|
| extract-flowjo-stats . . . . .                   | 24 |
| extract_all_gates . . . . .                      | 24 |
| extract_compensation . . . . .                   | 25 |
| extract_compensation_from_platforms . . . . .    | 25 |
| extract_flowjo_stats . . . . .                   | 26 |
| extract_gates_from_gatingset_v10 . . . . .       | 27 |
| extract_keywords_for_samples . . . . .           | 28 |
| extract_pdata . . . . .                          | 28 |
| extract_populations_from_gatingset_v10 . . . . . | 28 |
| extract_samples_from_gatingset_v10 . . . . .     | 29 |
| extract_transformations . . . . .                | 29 |
| extract_transformation_spec . . . . .            | 30 |
| extract_workspace_compensation . . . . .         | 30 |
| filter_samples . . . . .                         | 30 |
| find_root_population . . . . .                   | 31 |
| fj11_to_gatingset . . . . .                      | 31 |
| format_gate_num . . . . .                        | 34 |
| gates . . . . .                                  | 34 |
| generate_flowjo10_xml . . . . .                  | 34 |
| generate_group_subpopulations_xml . . . . .      | 35 |
| generate_logical_node_xml . . . . .              | 36 |
| generate_sample_subpopulations_xml . . . . .     | 36 |
| get_display_range . . . . .                      | 37 |
| get_fcs_header_keywords . . . . .                | 37 |
| get_graph_axes . . . . .                         | 38 |
| get_group_info . . . . .                         | 38 |
| get_referenced_channels . . . . .                | 39 |
| get_sample_file_map . . . . .                    | 39 |
| get_sanitize_slashes . . . . .                   | 40 |
| get_transform_spec . . . . .                     | 40 |
| get_verbose . . . . .                            | 41 |
| helpers . . . . .                                | 41 |
| helpers-conversion . . . . .                     | 41 |
| load_example_workspace . . . . .                 | 42 |
| map_compensation_names . . . . .                 | 42 |
| map_gate_params_to_gh . . . . .                  | 43 |
| map_param_names . . . . .                        | 43 |
| map_transformation_names . . . . .               | 44 |
| null-coalesce . . . . .                          | 45 |
| parse_boolean_expression . . . . .               | 45 |
| parse_compensation_data . . . . .                | 45 |
| parse_spill_keyword . . . . .                    | 46 |
| parse_transformation_info . . . . .              | 46 |
| populations . . . . .                            | 46 |
| pretty_print_flowjo . . . . .                    | 47 |
| process_zip_archive . . . . .                    | 47 |
| read_flowjo11_workspace . . . . .                | 48 |
| resolve_all_fcs_paths . . . . .                  | 48 |
| safe-coercion . . . . .                          | 49 |
| sanitize_path_with_separator . . . . .           | 49 |
| sanitize_population_name . . . . .               | 50 |
| search_fcs_files . . . . .                       | 50 |

|                                    |    |
|------------------------------------|----|
| set_sanitize_slashes . . . . .     | 51 |
| set_verbose . . . . .              | 51 |
| transformations . . . . .          | 52 |
| update_gate_param_names . . . . .  | 52 |
| validate_compensation . . . . .    | 52 |
| verify_gate_marker_names . . . . . | 53 |
| write_fcs_files_to_dir . . . . .   | 53 |
| xml_encode . . . . .               | 54 |

|              |           |
|--------------|-----------|
| <b>Index</b> | <b>55</b> |
|--------------|-----------|

---

|                |                                                                          |
|----------------|--------------------------------------------------------------------------|
| CyFj11-package | <i>CyFj11: Parse FlowJo V11 Workspace Files and Export to FlowJo V10</i> |
|----------------|--------------------------------------------------------------------------|

---

## Description

Provides functions to parse FlowJo v11 workspace files (.flowjo) and extract gating information, populations, and associated FCS files. Includes utilities for exporting data to FlowJo v10 workspace format. The package enables bidirectional data flow between FlowJo and R/Bioconductor, facilitating reproducible analysis workflows.

## Author(s)

**Maintainer:** Bernd Jagla <bernd.jagla@pasteur.fr> (**ORCID**) (affiliation: Institut Pasteur, scBiomarkers UTechS & Bioinformatics & Biostatistics Hub)

Authors:

- Bernd Jagla <bernd.jagla@pasteur.fr> (**ORCID**) (affiliation: Institut Pasteur, scBiomarkers UTechS & Bioinformatics & Biostatistics Hub)
- Slobodan Culina <Slobodan.culina@pasteur.fr> (affiliation: Institut Pasteur, scBiomarkers UTechS)
- Francois Le Guerroue <francois.le-guerroue@pasteur.fr> (affiliation: Institut Pasteur, scBiomarkers UTechS)
- Esma Karkeni <esma.karkeni@pasteur.fr> (affiliation: Institut Pasteur, scBiomarkers UTechS)
- Milena Hasan <milena.hasan@pasteur.fr> (affiliation: Institut Pasteur, scBiomarkers UTechS)

## See Also

Useful links:

- <https://github.com/C3BI-pasteur-fr/CyFj11>
- Report bugs at <https://github.com/C3BI-pasteur-fr/CyFj11/issues>

---

```
add_populations_to_gatingset
```

*Add Populations to GatingSet*

---

### Description

Adds population nodes and gates to GatingSet based on gating trees

### Usage

```
add_populations_to_gatingset(
  gs,
  gating_trees,
  gates,
  sample_uuids,
  strip_comp_prefix = TRUE,
  verbose = FALSE
)
```

### Arguments

|                   |                                                                                                              |
|-------------------|--------------------------------------------------------------------------------------------------------------|
| gs                | GatingSet object                                                                                             |
| gating_trees      | List of gating trees (one per sample)                                                                        |
| gates             | List of gate objects                                                                                         |
| sample_uuids      | Vector of sample UUIDs                                                                                       |
| strip_comp_prefix | Logical. Strip "Comp-" prefix from gate parameter names when matching to GatingSet parameters? Default TRUE. |
| verbose           | Logical. Print verbose messages? Default FALSE.                                                              |

---

```
add_population_node
```

*Add Population Node Recursively to a GatingHierarchy*

---

### Description

Recursively walks a gating tree node and adds each population—together with its associated gate—to a single-sample GatingHierarchy. Both regular flow-cytometry gates (rectangle, polygon, ellipsoid, quadrant) and logical (boolean) gates (and/or/not) are supported.

### Usage

```
add_population_node(
  gh,
  node,
  gates,
  sample_uuid,
  parent = "root",
  strip_comp_prefix = TRUE,
  verbose = FALSE,
  deferred = NULL
)
```

## Arguments

|                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|-------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| gh                | A GatingHierarchy object (single sample) to which populations are added.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| node              | A named list representing one node of the gating tree produced by <a href="#">build_gating_tree</a> .<br>Expected fields:<br>name Character or list. Population name.<br>type Character. Gate type, e.g. "rectangle", "polygon", "and", "or", "not", "root".<br>definition_uuid Character or list. UUID of the corresponding population definition—used to look up the gate object.<br>logical_gate_info Optional list describing a boolean gate; present only when type is "and", "or", or "not". Contains operator and combined_populations.<br>children Optional list of child nodes.<br>parent Character. Full path of the parent population, used when recursing into children. |
| gates             | Named list of flowCore gate objects keyed by paste0(definition_uuid, "_", sample_uuid). Typically the output of <a href="#">extract_all_gates</a> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| sample_uuid       | Character scalar. UUID of the sample being processed. Used to look up the correct gate from gates and to provide informative warning messages.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| parent            | Character scalar. Path of the parent population in gh under which the current node will be added. Use "root" (the default) for top-level populations.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| strip_comp_prefix | Logical. When TRUE (default), the "Comp-" prefix is stripped from gate parameter names before matching against the GatingHierarchy parameters. Set to FALSE when compensation has already been applied with names that include the prefix.                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| verbose           | Logical. When TRUE, diagnostic messages are emitted via <a href="#">message</a> at each step (gate lookup, boolean expression construction, name mapping, etc.). Default FALSE.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| deferred          | An environment (or NULL) used to collect logical gates that could not be added because one or more component populations are not yet present in gh. Callers should pass the same environment object on all recursive calls so that <a href="#">add_populations_to_gatingset</a> can retry the deferred gates in a subsequent pass. When NULL, unresolvable logical gates raise a warning immediately instead of being deferred.                                                                                                                                                                                                                                                      |

## Details

For logical gates, all component populations must already exist in the GatingHierarchy before the boolean expression can be evaluated. When a component is missing the gate is pushed onto deferred rather than silently dropped, allowing [add\\_populations\\_to\\_gatingset](#) to retry in a subsequent pass once earlier populations have been resolved.

The function performs the following steps for each node:

1. Sanitizes the population name (replaces / with :) to avoid conflicts with the flowWorkspace path separator.
2. Skips the virtual root/ungated node (already present in every new GatingSet).
3. Looks up the gate object in gates using the key paste0(definition\_uuid, "\_", sample\_uuid).
4. For **regular gates**: verifies marker-name consistency, adjusts parameter names via [adjust\\_gate\\_transformation](#) and adds the gate with [gs\\_pop\\_add](#).

5. For **logical gates**: resolves each component population to its path in the hierarchy, builds a `booleanFilter` expression, and adds it with `gs_pop_add`. If any component path cannot be resolved the node is pushed to deferred.
6. Recurses into `node$children`.

### Value

NULL invisibly. The function modifies `gh` and `deferred` in place as side effects.

### See Also

[add\\_populations\\_to\\_gatingset](#) for the top-level caller that manages the deferred-retry loop, [build\\_gating\\_tree](#) for tree construction, [adjust\\_gate\\_transformations](#) for parameter-name remapping, [extract\\_all\\_gates](#) for gate extraction.

---

`adjust_gate_transformations`

*Compare and Adjust Gate Parameter Names to Match a GatingHierarchy*

---

### Description

Ensures that the parameter names stored inside a `flowCore` gate object are consistent with the parameter names present in a `GatingHierarchy` after compensation and transformations have been applied.

### Usage

```
adjust_gate_transformations(gh, gate_obj, strip_comp_prefix = TRUE)
```

### Arguments

|                                |                                                                                                                                                                                                                                                                                                                                                             |
|--------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>gh</code>                | A <code>GatingHierarchy</code> object (single sample). Used to retrieve both the registered transformers ( <code>gh_get_transformations</code> ) and the full list of column names from the root population data frame.                                                                                                                                     |
| <code>gate_obj</code>          | A <code>flowCore</code> gate object (e.g., <code>rectangleGate</code> , <code>polygonGate</code> , <code>ellipsoidGate</code> , <code>quadGate</code> ). Its internal parameter names will be remapped to match <code>gh</code> .                                                                                                                           |
| <code>strip_comp_prefix</code> | Logical. When <code>TRUE</code> (default), the "Comp-" prefix is stripped from gate parameter names before attempting to match them against the <code>GatingHierarchy</code> parameter names. Set to <code>FALSE</code> when gate names are expected to already include the "Comp-" prefix (i.e., compensation was applied externally with matching names). |

### Details

The function performs two distinct tasks:

1. **Parameter-name remapping.** Gate parameter names extracted from the FlowJo workspace may differ from the names actually present in the GatingHierarchy (e.g., a gate may reference "Comp-APC-Ax700-A" while the compensated cytoframe column is named "APC-Ax700-A", or "/" characters may have been sanitized to "\_" by flowCore). The function resolves these discrepancies via `map_gate_params_to_gh` and then rewrites the gate's internal parameter slots with `update_gate_param_names`.
2. **Transformation-space adjustment (currently disabled).** A code path exists—guarded by `if (FALSE)`—that would convert gate coordinates from raw data space to transformed (display) space using the transformers registered in the GatingHierarchy. This path is disabled because the current pipeline calls `extract_all_gates` with `use_transformed_coords = FALSE`, so gate coordinates are already expressed in the same space as the (untransformed) cytoframe data.

The set of parameter names considered for matching is the union of:

- names returned by `gh_get_transformations` (non-linear channels only), and
- all column names of the root cytoframe (so that linear / scatter parameters such as FSC-A and SSC-A are also available for matching).

If no gate parameter can be mapped to any GatingHierarchy parameter the original gate object is returned unchanged.

### Value

The input `gate_obj` with its internal parameter names updated to match those in `gh`. All gate coordinate slots (`@min`, `@max`, `@boundaries`, `@boundary`, `@mean`, `@cov`) are updated consistently via `update_gate_param_names`. If no mapping is found, the original object is returned unmodified.

### See Also

`map_gate_params_to_gh` for the underlying name-mapping logic, `update_gate_param_names` for the slot-level renaming, `apply_transforms_to_gate` for the (currently disabled) coordinate-transformation step, `add_population_node` which calls this function before passing a gate to `gs_pop_add`.

---

apply\_extension

*Apply extension to coordinates*

---

### Description

Apply extension to coordinates

### Usage

```
apply_extension(coords, extend_val = 0, extend_to = -4000)
```

### Arguments

|                         |                               |
|-------------------------|-------------------------------|
| <code>coords</code>     | Numeric vector of coordinates |
| <code>extend_val</code> | Threshold value               |
| <code>extend_to</code>  | Replacement value             |

## Value

Extended coordinates

---

apply\_param\_mapping      *Apply Parameter Name Mapping to Names Vector*

---

## Description

Applies a parameter name mapping to convert source names to target names.

## Usage

```
apply_param_mapping(source_names, mapping, on_no_match = "keep")
```

## Arguments

|              |                                                                                                                                         |
|--------------|-----------------------------------------------------------------------------------------------------------------------------------------|
| source_names | Character vector of source names                                                                                                        |
| mapping      | Named list from map_param_names()                                                                                                       |
| on_no_match  | Character. What to do with unmapped names: "keep" - keep original name (default) "drop" - remove from result "warn" - keep with warning |

## Value

Character vector of mapped names

---

apply\_transforms\_to\_gate      *Apply Transformation Functions to Gate Coordinates*

---

## Description

Converts the coordinate values stored inside a flowCore gate object from one data space to another by evaluating a set of named transformation functions. This is typically used to move gate coordinates from raw (instrument) space into a transformed (display) space so that they align with the transformed cytoframe data held in a GatingHierarchy.

## Usage

```
apply_transforms_to_gate(gate_obj, trans_list)
```

## Arguments

|            |                                                                                                                                                                                                                                                                                                                                                                          |
|------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| gate_obj   | A flowCore gate object whose coordinate slots will be transformed. Supported classes: rectangleGate, quadGate, polygonGate, ellipsoidGate. Objects of other classes are returned unmodified.                                                                                                                                                                             |
| trans_list | A named list of transformation functions, where each name is a GatingHierarchy parameter name (e.g., "APC-Ax700-A") and each value is a single-argument function that maps numeric raw values to transformed values. Typically a subset of the list returned by <a href="#">gh_get_transformations</a> . An empty list causes the function to return gate_obj unchanged. |

## Details

Each gate type stores its defining coordinates in different slots; the function handles all four common types:

`rectangleGate` The `@min` and `@max` named numeric vectors are transformed channel-by-channel for every parameter that appears in `trans_list`.

`quadGate` The `@boundary` named numeric vector (the two divider positions, one per axis) is transformed for every matching parameter.

`polygonGate` Each column of the `@boundaries` matrix that matches a parameter in `trans_list` is transformed element-wise, preserving the polygon vertex order.

`ellipsoidGate` The `@mean` vector is transformed parameter-by-parameter. A warning is emitted because applying a non-linear transform to the mean without also adjusting the covariance matrix `@cov` does not correctly preserve the ellipsoid shape; the covariance matrix is left unchanged.

Parameters present in the gate but absent from `trans_list` (e.g., scatter channels with a linear or identity transform) are left unchanged. If `trans_list` is empty the original gate object is returned immediately without modification.

## Value

The input `gate_obj` with coordinate slots updated in place for all parameters found in `trans_list`. The object class and all other slots are preserved.

## Note

This function is currently **not called** by the active pipeline. Gate coordinates are extracted in transformed (display) space by [extract\\_all\\_gates](#) when `use_transformed_coords = TRUE` (the default), making an additional in-place transformation of gate coordinates unnecessary. The function is retained for use cases where gate coordinates must be converted programmatically after extraction.

## See Also

[adjust\\_gate\\_transformations](#) which determines which transformations need to be applied and calls this function, [extract\\_all\\_gates](#) for gate extraction with optional coordinate-space control via `use_transformed_coords`, [gh\\_get\\_transformations](#) for retrieving the transformation functions registered in a `GatingHierarchy`.

## Description

Extract FlowJo v11 Workspace

---

|                   |                                           |
|-------------------|-------------------------------------------|
| build_gating_tree | <i>Build Gating Tree for given Sample</i> |
|-------------------|-------------------------------------------|

---

**Description**

Build Gating Tree for given Sample

**Usage**

```
build_gating_tree(sample_uuid, populations, populationDefinitions, root_uuid)
```

---

|                       |                                                                             |
|-----------------------|-----------------------------------------------------------------------------|
| build_sample_keywords | <i>Build Sample Keywords from Original FCS Header and GatingSet Overlay</i> |
|-----------------------|-----------------------------------------------------------------------------|

---

**Description**

Build FlowJo v10 Workspace Keywords

**Usage**

```
build_sample_keywords(fcs_keywords, gs_keywords, final_filename)
```

**Arguments**

fcs\_keywords     Keywords from the original FCS header.  
gs\_keywords     Keywords from the GatingSet.  
final\_filename   The value to set for FILENAME.

**Value**

Named list of keywords for the exported workspace.

---

|                            |                                          |
|----------------------------|------------------------------------------|
| build_spillover_matrix_xml | <i>Build FlowJo Spillover Matrix XML</i> |
|----------------------------|------------------------------------------|

---

**Description**

Build FlowJo Spillover Matrix XML

**Usage**

```
build_spillover_matrix_xml(spill_matrix, matrix_id, indent = "    ")
```

**Arguments**

|                           |                                            |
|---------------------------|--------------------------------------------|
| <code>spill_matrix</code> | Compensation matrix with column/row names. |
| <code>matrix_id</code>    | UUID for the matrix.                       |
| <code>indent</code>       | Indentation string.                        |

**Value**

Character vector of XML lines.

---

`check_wsp_csv_coverage`

*Check Workspace-to-CSV Coverage*

---

**Description**

Reports which .wsp files have statistics that are missing from a mapping CSV. Useful for detecting mismatches before calling `extract_flowjo_stats()` in wide-table mode.

**Usage**

```
check_wsp_csv_coverage(wsp_files, csv_file)
```

**Arguments**

|                        |                                                                                                                                 |
|------------------------|---------------------------------------------------------------------------------------------------------------------------------|
| <code>wsp_files</code> | Character vector of paths to .wsp files.                                                                                        |
| <code>csv_file</code>  | Path to a mapping CSV with the columns <code>population_path</code> , <code>stat_name</code> , and <code>stat_ancestor</code> . |

**Value**

A tibble with one row per `wsp_file` and columns:

**file** The workspace file name.

**total\_signatures** Number of unique statistic signatures in the file.

**missing\_signatures** Number of signatures not present in the CSV.

**covered** Logical; TRUE if all signatures are in the CSV.

---

compensation

*Compensation Functions for FlowJo v11*

---

**Description**

Compensation Functions for FlowJo v11

---

|            |                                                  |
|------------|--------------------------------------------------|
| conversion | <i>Convert FlowJo v11 Workspace to GatingSet</i> |
|------------|--------------------------------------------------|

---

**Description**

Convert FlowJo v11 Workspace to GatingSet

---

|                      |                             |
|----------------------|-----------------------------|
| convert_boolean_gate | <i>Convert Boolean Gate</i> |
|----------------------|-----------------------------|

---

**Description**

Convert Boolean Gate

**Usage**

convert\_boolean\_gate(gate, pop\_name)

---

|                             |                                                  |
|-----------------------------|--------------------------------------------------|
| convert_boolean_to_flowjo10 | <i>Convert Boolean Gate to FlowJo v10 Format</i> |
|-----------------------------|--------------------------------------------------|

---

**Description**

Convert Boolean Gate to FlowJo v10 Format

**Usage**

convert\_boolean\_to\_flowjo10(gate, pop\_name, gh = NULL)

**Arguments**

- |          |                        |
|----------|------------------------|
| gate     | booleanFilter object   |
| pop_name | Population name        |
| gh       | GatingHierarchy object |

**Value**

List representing boolean gate in FlowJo v10 format

---

|                      |                             |
|----------------------|-----------------------------|
| convert_ellipse_gate | <i>Convert Ellipse Gate</i> |
|----------------------|-----------------------------|

---

**Description**

Convert Ellipse Gate

**Usage**

```
convert_ellipse_gate(  
  gate,  
  pop_name,  
  extend_val,  
  extend_to,  
  correct_faulty_gate = 0,  
  use_transformed_coords = FALSE  
)
```

---

|                               |                                                    |
|-------------------------------|----------------------------------------------------|
| convert_ellipsoid_to_flowjo10 | <i>Convert Ellipsoid Gate to FlowJo v10 Format</i> |
|-------------------------------|----------------------------------------------------|

---

**Description**

Convert Ellipsoid Gate to FlowJo v10 Format

**Usage**

```
convert_ellipsoid_to_flowjo10(gate, pop_name, gh = NULL)
```

**Arguments**

|          |                      |
|----------|----------------------|
| gate     | ellipsoidGate object |
| pop_name | Population name      |

**Value**

List representing ellipsoid gate in FlowJo v10 format

---

|                     |                                                    |
|---------------------|----------------------------------------------------|
| convert_flowjo_gate | <i>Convert FlowJo Gate to flowCore Gate Object</i> |
|---------------------|----------------------------------------------------|

---

**Description**

Convert FlowJo Gate to flowCore Gate Object

**Usage**

```
convert_flowjo_gate(  
  gate,  
  pop_name,  
  pop_type,  
  channel.ignore.case = FALSE,  
  extend_val = 0,  
  extend_to = -4000,  
  correct_faulty_gate = 0,  
  use_transformed_coords = FALSE  
)
```

---

|                                 |                                                   |
|---------------------------------|---------------------------------------------------|
| convert_gate_to_flowjo10_format | <i>Convert flowCore Gate to FlowJo v10 Format</i> |
|---------------------------------|---------------------------------------------------|

---

**Description**

Convert flowCore Gate to FlowJo v10 Format

**Usage**

```
convert_gate_to_flowjo10_format(gate, pop_name, gh = NULL)
```

**Arguments**

|          |                      |
|----------|----------------------|
| gate     | flowCore gate object |
| pop_name | Population name      |

**Value**

List representing gate in FlowJo v10 format

---

convert\_polygon\_gate    *Convert Polygon Gate*

---

### Description

Convert Polygon Gate

### Usage

```
convert_polygon_gate(
    gate,
    pop_name,
    extend_val,
    extend_to,
    correct_faulty_gate = 0,
    use_transformed_coords = FALSE
)
```

---

convert\_polygon\_to\_flowjo10  
                                   *Convert Polygon Gate to FlowJo v10 Format*

---

### Description

Convert Polygon Gate to FlowJo v10 Format

### Usage

```
convert_polygon_to_flowjo10(gate, pop_name, gh = NULL)
```

---

convert\_quadrant\_gate    *Convert Quadrant Gate*

---

### Description

Convert Quadrant Gate

### Usage

```
convert_quadrant_gate(
    gate,
    pop_name,
    extend_val,
    extend_to,
    correct_faulty_gate = 0,
    use_transformed_coords = FALSE
)
```

---

|                    |                                |
|--------------------|--------------------------------|
| convert_range_gate | <i>Convert Range Gate (1D)</i> |
|--------------------|--------------------------------|

---

**Description**

Convert Range Gate (1D)

**Usage**

```
convert_range_gate(  
    gate,  
    pop_name,  
    extend_val,  
    extend_to,  
    correct_faulty_gate = 0,  
    use_transformed_coords = FALSE  
)
```

---

|                        |                               |
|------------------------|-------------------------------|
| convert_rectangle_gate | <i>Convert Rectangle Gate</i> |
|------------------------|-------------------------------|

---

**Description**

Convert Rectangle Gate

**Usage**

```
convert_rectangle_gate(  
    gate,  
    pop_name,  
    extend_val,  
    extend_to,  
    correct_faulty_gate = 0,  
    use_transformed_coords = FALSE  
)
```

---

|                               |                                                    |
|-------------------------------|----------------------------------------------------|
| convert_rectangle_to_flowjo10 | <i>Convert Rectangle Gate to FlowJo v10 Format</i> |
|-------------------------------|----------------------------------------------------|

---

**Description**

Convert Rectangle Gate to FlowJo v10 Format

**Usage**

```
convert_rectangle_to_flowjo10(gate, pop_name, gh = NULL)
```

---

`create_biexponential_transform`*Create Biexponential Transform (FlowJo version)*

---

**Description**

Create Biexponential Transform (FlowJo version)

**Usage**`create_biexponential_transform(spec)`

---

`create_default_groups_v10`*Create Default Groups for FlowJo v10*

---

**Description**

Create Default Groups for FlowJo v10

**Usage**`create_default_groups_v10(samples)`**Arguments**

|                      |                     |
|----------------------|---------------------|
| <code>samples</code> | List of sample data |
|----------------------|---------------------|

**Value**

List of group data in FlowJo v10 format

---

`create_gatingset_from_cytoset`*Create GatingSet from Components - Transform First Approach*

---

**Description**

Create GatingSet from Components - Transform First Approach

**Usage**

```
create_gatingset_from_cytoset(
  cytoset,
  gating_trees,
  gates,
  compensations,
  transformations,
  sample_uuids,
  dataSources,
  keywords,
  additional.keys,
  additional.sampleID,
  keyword.ignore.case,
  strip_comp_prefix = TRUE
)
```

**Arguments**

`strip_comp_prefix`  
 Logical. Strip "Comp-" prefix from gate parameter names when adding populations? Default TRUE. Set to FALSE if compensation has already been applied and gate names should match the compensated parameter names.

---

create\_linear\_transform

*Create Linear Transform*

---

**Description**

Create Linear Transform

**Usage**

```
create_linear_transform(spec)
```

---

create\_log\_transform    *Create Log Transform (FlowJo / flowWorkspace compatible)*

---

**Description**

Builds a `scales::trans_new` object that mirrors FlowJo's Log transform. Accepts two parameterisations:

- flowWorkspace / FlowJo v10 style: decade, offset, scale (plus optional shift)
- FlowJo v11 parsed style: numberDecades, decadesOffset, vectorLength (plus optional shift)

**Usage**

```
create_log_transform(spec)
```

**Details**

The transform maps a raw value  $r$  to display space  $d$  as  $d = (\log_{10}(\max(r + \text{shift}, \text{offset})) - \log_{10}(\text{offset})) * \text{scale} / \text{decade}$  and back as  $r = 10^{(d * \text{decade} / \text{scale} + \log_{10}(\text{offset}))} - \text{shift}$

---

|                     |                            |
|---------------------|----------------------------|
| create_sample_names | <i>Create Sample Names</i> |
|---------------------|----------------------------|

---

**Description**

Create Sample Names

**Usage**

```
create_sample_names(  
  sample_uuids,  
  dataSources,  
  additional.keys,  
  additional.sampleID  
)
```

---

|                            |                                   |
|----------------------------|-----------------------------------|
| create_transformation_list | <i>Create Transformation List</i> |
|----------------------------|-----------------------------------|

---

**Description**

Create Transformation List

**Usage**

```
create_transformation_list(trans_spec)
```

**Details**

Creates a list of flowCore transformation objects from transformation specifications. Supports bi-exponential transformations which are mapped to logicle transforms.

---

 derive\_cytometer\_attrs

*Determine Cytometer Attributes from FCS Header*


---

### Description

Determine Cytometer Attributes from FCS Header

### Usage

```
derive_cytometer_attrs(fcs_keywords)
```

### Arguments

fcs\_keywords      Named list of FCS header keywords.

### Value

Named list of Cytometer XML attributes.

---

 display\_to\_raw

*Transform coordinates from FlowJo display space to raw data space*


---

### Description

FlowJo stores gate coordinates in display space  $\backslash(0 \text{ to } \text{gateResolution}/\text{vectorLength})$ . This function converts them to raw data space for use with flowCore gates.

Transformation logic:

- Linear: scale from  $[0, \text{vectorLength}]$  to  $[\text{minRange}, \text{maxRange}]$
- Biex/Logicle: apply inverse transform (display is in transformed space)

### Usage

```
display_to_raw(
  display_coords,
  transform_spec,
  gate_resolution = NULL,
  correct_faulty_gate = 0,
  use_transformed_coords = FALSE
)
```

### Arguments

display\_coords    Numeric vector of coordinates in display space

transform\_spec    Transform specification from gate axis

gate\_resolution

Gate resolution (overrides vectorLength if provided)

correct\_faulty\_gate

Fallback maxRange value if maxRange=0

**Value**

Numeric vector of coordinates in raw data space

---

|                    |                                               |
|--------------------|-----------------------------------------------|
| emit_transform_xml | <i>Emit a Single Channel Transform as XML</i> |
|--------------------|-----------------------------------------------|

---

**Description**

Emit a Single Channel Transform as XML

**Usage**

```
emit_transform_xml(  
  type,  
  channel,  
  transform_obj,  
  atr_tr,  
  data_range,  
  indent = "      "  
)
```

**Arguments**

- type Transform type: "biex", "log", "fasinh", or "linear".
- channel Parameter name for the transform.
- transform\_obj The transform function/object.
- atr\_tr Attributes list from the transform object.
- data\_range Numeric vector of length 2 (min, max) for linear transforms.
- indent Indentation string.

**Value**

Character vector of XML lines.

---

|                 |                                    |
|-----------------|------------------------------------|
| export-flowjo10 | <i>Export FlowJo v10 Workspace</i> |
|-----------------|------------------------------------|

---

**Description**

Export FlowJo v10 Workspace

---

export\_flowjo10\_workspace

*Export GatingSet to FlowJo v10 Workspace*


---

## Description

Export GatingSet to FlowJo v10 Workspace

## Usage

```
export_flowjo10_workspace(
  gating_set,
  output_path,
  workspace_name = NULL,
  fcs_root = NULL,
  overwrite = FALSE
)
```

## Arguments

|                             |                                                                                                                                                                                                             |
|-----------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>gating_set</code>     | A GatingSet object to export. A named list of GatingSets, as returned by <a href="#">fj11_to_gatingset</a> , is also accepted and merged with <code>flowWorkspace::merge_list_to_gs()</code> before export. |
| <code>output_path</code>    | Path where the .xml file should be created                                                                                                                                                                  |
| <code>workspace_name</code> | Optional name for the workspace                                                                                                                                                                             |
| <code>fcs_root</code>       | Optional base directory for FCS files. If provided, FCS files will be written there and URIs in the WSP will reference that location. If NULL, uses the actual paths found in the GatingSet.                |
| <code>overwrite</code>      | Logical. When <code>fcs_root</code> is supplied and FCS files already exist in that directory, FALSE (default) aborts with an error; TRUE silently replaces existing files.                                 |

## Value

Logical indicating success

## Examples

```
# Export a GatingSet to FlowJo v10 XML format
ws_path <- system.file("extdata", "min_test.flowjo", package = "CyFj11")
fcs_path <- system.file("extdata", package = "CyFj11")
ws <- read_flowjo11_workspace(ws_path)
gs <- fj11_to_gatingset(ws, group_name = 1, path = fcs_path)
out_file <- tempfile(fileext = ".xml")
export_flowjo10_workspace(gs, out_file)
file.exists(out_file)
```

---

|                      |                                                                  |
|----------------------|------------------------------------------------------------------|
| extract-flowjo-stats | <i>Extract Statistics from FlowJo Workspace (.wsp) XML Files</i> |
|----------------------|------------------------------------------------------------------|

---

### Description

Extract Statistics from FlowJo Workspace (.wsp) XML Files

---

|                   |                                                    |
|-------------------|----------------------------------------------------|
| extract_all_gates | <i>Extract All Gates from FlowJo v11 Workspace</i> |
|-------------------|----------------------------------------------------|

---

### Description

Converts FlowJo v11 gate definitions to flowCore gate objects

### Usage

```
extract_all_gates(
  populationDefinitions,
  sample_uuids,
  channel.ignore.case = FALSE,
  extend_val = 0,
  extend_to = -4000,
  correct_faulty_gate = 0,
  use_transformed_coords = FALSE
)
```

### Arguments

|                       |                                                   |
|-----------------------|---------------------------------------------------|
| populationDefinitions | Population definitions from workspace             |
| sample_uuids          | Vector of sample UUIDs to extract gates for       |
| channel.ignore.case   | Logical. Case-insensitive channel matching?       |
| extend_val            | Numeric. Threshold for extending gate coordinates |
| extend_to             | Numeric. Value to extend gates to                 |

### Value

Named list of gates for each population-sample combination

---

|                      |                                                                |
|----------------------|----------------------------------------------------------------|
| extract_compensation | <i>Extract Compensation Matrices from FlowJo v11 Workspace</i> |
|----------------------|----------------------------------------------------------------|

---

**Description**

Extracts compensation matrices for each sample, with option to override with custom compensation.

**Usage**

```
extract_compensation(  
  dataSources,  
  sample_uuids,  
  custom_compensation = NULL,  
  platforms = NULL  
)
```

**Arguments**

|                     |                                                                                               |
|---------------------|-----------------------------------------------------------------------------------------------|
| dataSources         | Data sources from workspace                                                                   |
| sample_uuids        | Vector of sample UUIDs                                                                        |
| custom_compensation | NULL, or compensation object/matrix/data.frame, or named list of these (names = sample UUIDs) |

**Value**

Named list of compensation matrices (one per sample)

---

|                                     |                                                    |
|-------------------------------------|----------------------------------------------------|
| extract_compensation_from_platforms | <i>Extract Compensation from Platforms Section</i> |
|-------------------------------------|----------------------------------------------------|

---

**Description**

Extract Compensation from Platforms Section

**Usage**

```
extract_compensation_from_platforms(  
  spillover_matrices,  
  dataSources = NULL,  
  sample_uuids = NULL  
)
```

**Arguments**

|                    |                                             |
|--------------------|---------------------------------------------|
| spillover_matrices | List of spillover matrices from platforms   |
| dataSources        | Data sources to map samples to compensation |
| sample_uuids       | Vector of sample UUIDs                      |

**Value**

Named list of compensation matrices (one per sample)

---

|                      |                                                                  |
|----------------------|------------------------------------------------------------------|
| extract_flowjo_stats | <i>Extract Statistics from FlowJo Workspace (.wsp) XML Files</i> |
|----------------------|------------------------------------------------------------------|

---

**Description**

Parses FlowJo workspace XML files and extracts population counts (from the count attribute of <Population>) and all <Statistic> elements. The function implements a two-step workflow:

**Usage**

```
extract_flowjo_stats(
  wsp_files,
  csv_file = NULL,
  output_csv = "wsp_stat_mapping.csv",
  write_csv = TRUE,
  value_as_numeric = TRUE,
  preserve_slashes = FALSE
)
```

**Arguments**

|                  |                                                                                                                                                                                                                                            |
|------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| wsp_files        | Character vector of paths to .wsp files.                                                                                                                                                                                                   |
| csv_file         | Optional path to a mapping CSV. Must contain the columns population_path, stat_name, and stat_ancestor. The better_name column is optional; if missing it is auto-generated from population, a short form of stat_name, and stat_ancestor. |
| output_csv       | Path where the template CSV is written when csv_file is NULL. Default: "wsp_stat_mapping.csv".                                                                                                                                             |
| write_csv        | Logical. If TRUE (default) and csv_file is NULL, write the template CSV to disk.                                                                                                                                                           |
| value_as_numeric | Logical. If TRUE (default), coerce raw values to numeric where possible.                                                                                                                                                                   |
| preserve_slashes | Logical. If TRUE, "/" characters in population names are preserved instead of being replaced with "_". Default FALSE.                                                                                                                      |

**Details**

1. Run **without** csv\_file to generate a template CSV containing every unique statistic signature found in the workspaces. The better\_name column is pre-filled automatically; edit only rows that collide or need a more human-readable label.
2. Run **with** csv\_file to validate the mapping and return a wide-format table: one row per sample, one column per renamed statistic.

The matching signature is the combination of population\_path, stat\_name, and stat\_ancestor. population\_path is the full gating lineage (e.g., "CD45+ / T cells / CD4+"). The id column is included for reference but is *not* used for matching, so it can be left blank in the CSV if desired.

**Value**

If `csv_file` is NULL, a named list:

`long` A tibble with all extracted counts/statistics in long format.

`mapping` A tibble of unique statistic signatures, ready to be written to `output_csv`.

If `csv_file` is provided, a wide tibble with one row per sample and one column per renamed statistic.

**Examples**

```
## Not run:
# Step 1: discover all statistics and write a template CSV
result <- extract_flowjo_stats(wsp_files = c("panel1.wsp", "panel2.wsp"))

# Step 2: edit wsp_stat_mapping.csv, then:
wide <- extract_flowjo_stats(
  wsp_files = c("panel1.wsp", "panel2.wsp"),
  csv_file = "wsp_stat_mapping.csv"
)

## End(Not run)
```

---

```
extract_gates_from_gatingset_v10
```

*Extract Gates from GatingSet for FlowJo v10*

---

**Description**

Extract Gates from GatingSet for FlowJo v10

**Usage**

```
extract_gates_from_gatingset_v10(gating_set)
```

**Arguments**

`gating_set`      GatingSet object

**Value**

List of gate data in FlowJo v10 format

---

`extract_keywords_for_samples`*Extract Keywords for Samples*

---

**Description**

Extract Keywords for Samples

**Usage**

```
extract_keywords_for_samples(sample_uuids, dataSources, keywords)
```

---

`extract_pdata`*Extract pData from Keywords*

---

**Description**

Extract pData from Keywords

**Usage**

```
extract_pdata(sample_uuids, dataSources, keywords, ignore.case)
```

---

`extract_populations_from_gatingset_v10`*Extract Populations from GatingSet for FlowJo v10*

---

**Description**

Extract Populations from GatingSet for FlowJo v10

**Usage**

```
extract_populations_from_gatingset_v10(gating_set, samples_data, gates_data)
```

**Arguments**

|                           |                  |
|---------------------------|------------------|
| <code>gating_set</code>   | GatingSet object |
| <code>samples_data</code> | Sample data      |
| <code>gates_data</code>   | Gate data        |

**Value**

List of population data in FlowJo v10 format

---

`extract_samples_from_gatingset_v10`*Extract Samples from GatingSet for FlowJo v10*

---

**Description**

Extract Samples from GatingSet for FlowJo v10

**Usage**

```
extract_samples_from_gatingset_v10(gating_set, target_fcs_dir = NULL)
```

**Arguments**

`gating_set`      GatingSet object

`target_fcs_dir`   The base directory that the WSP should assume the FCS files are in.

**Value**

List of sample data

---

`extract_transformations`*Extract Transformations from FlowJo v11 Workspace*

---

**Description**

Extracts transformation functions for each channel and sample

**Usage**

```
extract_transformations(populationDefinitions, sample_uuids)
```

**Arguments**

`populationDefinitions`

Population definitions from workspace

`sample_uuids`      Vector of sample UUIDs

**Details**

Extracts transformations from parameter specifications in gate definitions. Supports biexponential (Biex) transformations which are mapped to logicle transforms. Transformations are extracted per channel/marker, not per gate.

**Value**

Named list of transformation lists (one per sample)

---

`extract_transformation_spec`*Extract Transformation Specification*

---

**Description**

Extract Transformation Specification

**Usage**

```
extract_transformation_spec(sample_uuid, populationDefinitions)
```

**Details**

Extracts transformation specifications from parameter specifications in gate definitions. Transformations are per marker (channel), not per gate. Supports biexponential (Biex) transformations which are mapped to logicle transforms with appropriate parameters.

---

`extract_workspace_compensation`*Extract Compensation from Data Source*

---

**Description**

Extract Compensation from Data Source

**Usage**

```
extract_workspace_compensation(ds)
```

---

`filter_samples`*Filter Samples Based on Subset Argument*

---

**Description**

Filter Samples Based on Subset Argument

**Usage**

```
filter_samples(sample_uuids, subset, dataSources, keywords)
```

---

find\_root\_population     *Find Root Population*


---

**Description**

Find Root Population

**Usage**

```
find_root_population(populations, populationDefinitions, sample_uuid)
```

---



---

fj11\_to\_gatingset     *Convert FlowJo v11 Workspace to GatingSet*


---

**Description**

Parses a FlowJo v11 workspace and creates a GatingSet object with gating hierarchy, compensation, and transformations. Supports biexponential (Biex) transformations which are mapped to logicl transforms.

**Usage**

```
fj11_to_gatingset(
  fj11_workspace,
  group_name = NULL,
  subset = list(),
  execute = TRUE,
  path = NULL,
  cytoset = NULL,
  backend_dir = tempdir(),
  backend = c("h5", "tile"),
  include_gates = TRUE,
  compensation = NULL,
  additional.keys = "$TOT",
  additional.sampleID = FALSE,
  keywords = character(),
  keyword.ignore.case = FALSE,
  channel.ignore.case = FALSE,
  extend_val = 0,
  extend_to = -4000,
  leaf.bool = TRUE,
  include_empty_tree = FALSE,
  correct_faulty_gate = 0,
  transform = TRUE,
  use_transformed_coords = transform,
  max_search_depth = 5,
  stop_on_multiple = TRUE,
  mc.cores = 1,
  strip_comp_prefix = TRUE,
```

```

    sanitize_slashes = TRUE,
    ...
)

```

## Arguments

|                                     |                                                                                                                                                                     |
|-------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>fj11_workspace</code>         | List returned by <code>read_flowjo11_workspace()</code> containing the complete workspace structure                                                                 |
| <code>group_name</code>             | Character or numeric. Name or index of the sample group to import. If NULL, will list available groups interactively.                                               |
| <code>subset</code>                 | Numeric vector, character vector of FCS filenames, or expression to filter samples. See Details.                                                                    |
| <code>execute</code>                | Logical. Should gates be executed immediately? Default TRUE. If FALSE, creates GatingSet structure without computing cell counts.                                   |
| <code>path</code>                   | Character. Root directory to search for FCS files. Required if FCS files are not in workspace location.                                                             |
| <code>cytoset</code>                | A cytoset object providing alternative data source (instead of FCS files). Useful for preprocessed data.                                                            |
| <code>backend_dir</code>            | Directory for storing h5 backend data. Default is <code>tempdir()</code> .                                                                                          |
| <code>backend</code>                | Character. Storage backend, either "h5" or "tile". Default "h5".                                                                                                    |
| <code>include_gates</code>          | Logical. Should gates be imported? Default TRUE.                                                                                                                    |
| <code>compensation</code>           | Compensation object, matrix, data.frame, or named list of these. Overrides workspace compensation. Names should match sample GUIDs.                                 |
| <code>additional.keys</code>        | Character vector of FCS keywords to include in sample GUID (concatenated with filename). Default is "\$TOT".                                                        |
| <code>additional.sampleID</code>    | Logical. Include FlowJo sample ID in GUID? Default FALSE.                                                                                                           |
| <code>keywords</code>               | Character vector of keywords to extract as pData.                                                                                                                   |
| <code>keyword.ignore.case</code>    | Logical. Case-insensitive keyword matching? Default FALSE.                                                                                                          |
| <code>channel.ignore.case</code>    | Logical. Case-insensitive channel matching? Default FALSE.                                                                                                          |
| <code>extend_val</code>             | Numeric. Threshold for extending gate coordinates. Default 0.                                                                                                       |
| <code>extend_to</code>              | Numeric. Value to extend gates to. Default -4000.                                                                                                                   |
| <code>leaf.bool</code>              | Logical. Compute leaf boolean gates? Default TRUE.                                                                                                                  |
| <code>include_empty_tree</code>     | Logical. Include samples without gates? Default FALSE.                                                                                                              |
| <code>correct_faulty_gate</code>    | Numerical, zero = don't correct, otherwise use this value as transformation max value and Try to correct faulty gates instead of erroring? Default TRUE.            |
| <code>transform</code>              | Logical. Apply transformations? Default TRUE.                                                                                                                       |
| <code>use_transformed_coords</code> | Logical. Keep gate coordinates in FlowJo display (transformed) space rather than raw space? Default is the value of transform. Only relevant when transform = TRUE. |

`max_search_depth` Integer. Maximum directory depth when searching for FCS files. Default 5.

`stop_on_multiple` Logical. Stop if multiple FCS files match a sample? When TRUE, stops on both missing files and multiple matches. When FALSE, only stops on missing files and continues with multiple matches. Default TRUE.

`mc.cores` Integer. Number of cores for parallel processing. Default 1.

`strip_comp_prefix` Logical. Strip "Comp-" prefix from gate parameter names when adding populations? Default TRUE. Set to FALSE if compensation has already been applied and gate names should match the compensated parameter names.

`sanitize_slashes` Logical. Replace "/" with "\_" in parameter names? Default TRUE (matches flowCore behavior). Set to FALSE if you want to preserve "/" in marker names (e.g., "CD3/CD4" stays as-is).

... Additional arguments passed to FCS parser.

## Details

The subset argument can be:

**Numeric vector:** Sample indices to import

**Character vector:** FCS filenames to import

**Expression:** Logical expression to filter pData (requires keywords argument)

FlowJo v11 stores data differently than v10:

**populationDefinitions:** Master gate definitions

**desyncTable:** Per-sample gate variations (tailored gates)

**groups:** Sample groupings

**dataSources:** FCS file references

## Value

A named list of `GatingSet` objects, one per sample. Each element contains the full gating hierarchy, gates, compensation, and transformations for that sample. To obtain a single merged `GatingSet` for downstream analysis, call `flowWorkspace::merge_list_to_gs(gsList)` on the returned list.

## Examples

```
ws_path <- system.file("extdata", "min_test.flowjo", package = "CyFj11")
fcs_path <- system.file("extdata", package = "CyFj11")
ws <- read_flowjo11_workspace(ws_path)
gsList <- fj11_to_gatingset(ws, group_name = 1, path = fcs_path)
length(gsList) # Number of GatingSets created
```

---

|                 |                                                                   |
|-----------------|-------------------------------------------------------------------|
| format_gate_num | <i>Format a gate coordinate or dimension value for XML output</i> |
|-----------------|-------------------------------------------------------------------|

---

**Description**

Uses 15 significant figures and strips trailing zeros, matching the precision FlowJo stores gate boundaries in.

**Usage**

format\_gate\_num(x)

**Arguments**

x                      Numeric value.

**Value**

Character string suitable for embedding in an XML attribute.

---

|       |                                                 |
|-------|-------------------------------------------------|
| gates | <i>Gate Conversion Functions for FlowJo v11</i> |
|-------|-------------------------------------------------|

---

**Description**

Gate Conversion Functions for FlowJo v11

---

|                       |                                        |
|-----------------------|----------------------------------------|
| generate_flowjo10_xml | <i>Generate FlowJo v10 XML Content</i> |
|-----------------------|----------------------------------------|

---

**Description**

Generate FlowJo v10 XML Content

**Usage**

```
generate_flowjo10_xml(  
  gating_set,  
  samples,  
  gates,  
  populations,  
  groups,  
  workspace_name,  
  output_path,  
  force_XSC_linear = FALSE,  
  minimal_fj11 = FALSE  
)
```

**Arguments**

|                             |                         |
|-----------------------------|-------------------------|
| <code>samples</code>        | List of sample data     |
| <code>gates</code>          | List of gate data       |
| <code>populations</code>    | List of population data |
| <code>groups</code>         | List of group data      |
| <code>workspace_name</code> | Name of the workspace   |

**Value**

Character string containing XML content

---

`generate_group_subpopulations_xml`

*Generate Group Node Subpopulations XML*

---

**Description**

Recursively generates XML for group node population hierarchy

**Usage**

```
generate_group_subpopulations_xml(
  populations,
  gates,
  parent_path = "root",
  indent = "    ",
  visited_paths = NULL,
  gh = NULL
)
```

**Arguments**

|                            |                                                               |
|----------------------------|---------------------------------------------------------------|
| <code>populations</code>   | List of population data                                       |
| <code>gates</code>         | List of gate data                                             |
| <code>parent_path</code>   | Parent population path (default "root")                       |
| <code>indent</code>        | Current indentation level for XML formatting                  |
| <code>visited_paths</code> | Character vector to track visited paths (for cycle detection) |
| <code>gh</code>            | Optional GatingHierarchy object (for boolean gate processing) |

**Value**

Character vector of XML lines

---

```
generate_logical_node_xml
```

*Generate Logical Node XML (AndNode, OrNode, NotNode)*

---

### Description

Generate Logical Node XML (AndNode, OrNode, NotNode)

### Usage

```
generate_logical_node_xml(gate, pop_name, child_path, indent, gh, gates = NULL)
```

### Arguments

|            |                                                            |
|------------|------------------------------------------------------------|
| gate       | Gate data containing boolean definition                    |
| pop_name   | Original population name (for fallback)                    |
| child_path | Full path to the population                                |
| indent     | XML indentation string                                     |
| gh         | GatingHierarchy object                                     |
| gates      | Full gates list (for looking up dependent gates if needed) |

### Value

Character vector of XML lines

---

```
generate_sample_subpopulations_xml
```

*Generate Sample Subpopulations XML*

---

### Description

Recursively generates XML for sample-specific population hierarchy.

### Usage

```
generate_sample_subpopulations_xml(
  gating_hierarchy,
  gates,
  populations,
  parent_path = "root",
  indent = "      ",
  heat_map_param = ""
)
```

**Arguments**

|                  |                                                       |
|------------------|-------------------------------------------------------|
| gating_hierarchy | GatingHierarchy object                                |
| gates            | List of gate data                                     |
| populations      | List of population data (pre-filtered to this sample) |
| parent_path      | Parent population path (default "root")               |
| indent           | Current indentation string for XML formatting         |
| heat_map_param   | Channel name used for heatMapStatParameter attribute  |

**Value**

Character vector of XML lines

---

|                   |                                        |
|-------------------|----------------------------------------|
| get_display_range | <i>Get Display Range for Parameter</i> |
|-------------------|----------------------------------------|

---

**Description**

Determines the min/max range for a parameter that will be used in the XML

**Usage**

```
get_display_range(gh, param_name)
```

---

|                         |                                          |
|-------------------------|------------------------------------------|
| get_fcs_header_keywords | <i>Read Original FCS Header Keywords</i> |
|-------------------------|------------------------------------------|

---

**Description**

Read Original FCS Header Keywords

**Usage**

```
get_fcs_header_keywords(fcs_path)
```

**Arguments**

|          |                      |
|----------|----------------------|
| fcs_path | Path to an FCS file. |
|----------|----------------------|

**Value**

Named list of header keywords, or NULL if unavailable.

---

|                |                                                          |
|----------------|----------------------------------------------------------|
| get_graph_axes | <i>Safely get graph axis parameters for a population</i> |
|----------------|----------------------------------------------------------|

---

**Description**

Walks the hierarchy to find appropriate axes for the graph display. Handles boolean gates (no parameters) by checking children, self, parent.

**Usage**

```
get_graph_axes(gh, pop_path)
```

**Arguments**

|          |                 |
|----------|-----------------|
| gh       | GatingHierarchy |
| pop_path | Population path |

**Value**

Character vector of length 1-2 with channel names

---

|                |                              |
|----------------|------------------------------|
| get_group_info | <i>Get Group Information</i> |
|----------------|------------------------------|

---

**Description**

Get Group Information

**Usage**

```
get_group_info(groups)
```

**Arguments**

|        |                                       |
|--------|---------------------------------------|
| groups | Groups list from FlowJo v11 workspace |
|--------|---------------------------------------|

**Value**

Data frame with group info

---

`get_referenced_channels`*Collect all channel names referenced by gates in the workspace*

---

**Description**

Collect all channel names referenced by gates in the workspace

**Usage**

```
get_referenced_channels(gates)
```

**Arguments**

`gates` Gate data list as returned by `extract_gates_from_gatingset_v10`

**Value**

Character vector of unique channel names referenced by gates.

---

`get_sample_file_map` *Get Sample-to-File Mapping*

---

**Description**

Creates a simple lookup table mapping sample UUIDs to FCS file paths

**Usage**

```
get_sample_file_map(resolution_results, include_status = "FOUND")
```

**Arguments**

`resolution_results`

Output from `resolve_all_fcs_paths()`

`include_status` Include only samples with specific status (default: "FOUND")

**Value**

Named vector where names are `sample_ids` and values are `resolved_paths`

---

|                      |                                    |
|----------------------|------------------------------------|
| get_sanitize_slashes | <i>Get slash sanitization mode</i> |
|----------------------|------------------------------------|

---

**Description**

Check whether "/" characters in parameter names are being replaced with "\_".

**Usage**

```
get_sanitize_slashes()
```

**Value**

Logical. TRUE if "/" is replaced with "\_", FALSE if preserved

**Examples**

```
# Check current setting
is_sanitize_slashes <- get_sanitize_slashes()
```

---

|                    |                                               |
|--------------------|-----------------------------------------------|
| get_transform_spec | <i>Get Transform Specification for Export</i> |
|--------------------|-----------------------------------------------|

---

**Description**

Extracts transformation specification from gating hierarchy for export to FlowJo format. Handles biexponential, linear, log, logicle, and arcsinh.

**Usage**

```
get_transform_spec(gh, dim = "SSC-A")
```

**Arguments**

|     |                                                     |
|-----|-----------------------------------------------------|
| gh  | GatingHierarchy object                              |
| dim | Character string naming the channel (e.g. "FITC-A") |

**Value**

Named list representing FlowJo transformation specification, or NULL

---

|             |                         |
|-------------|-------------------------|
| get_verbose | <i>Get verbose mode</i> |
|-------------|-------------------------|

---

**Description**

Check whether verbose output is currently enabled.

**Usage**

```
get_verbose()
```

**Value**

Logical. TRUE if verbose output is enabled, FALSE otherwise

**Examples**

```
# Check current verbose setting
is_verbose <- get_verbose()
is_verbose  # Should be FALSE by default
```

---

|         |                                         |
|---------|-----------------------------------------|
| helpers | <i>Helper Functions for the Package</i> |
|---------|-----------------------------------------|

---

**Description**

Helper Functions for the Package

---

|                    |                                                   |
|--------------------|---------------------------------------------------|
| helpers-conversion | <i>Helper Functions for FlowJo v11 Conversion</i> |
|--------------------|---------------------------------------------------|

---

**Description**

Helper Functions for FlowJo v11 Conversion

---

```
load_example_workspace
```

*Load Example FlowJo v11 Workspace*

---

### Description

Loads the example FlowJo v11 workspace from inst/extdata/min\_test.flowjo for use in tests and examples. This workspace contains a simple gating hierarchy (Ungated -> Lymphocytes -> CD8, CD3 subset) and references sample04.fcs.

### Usage

```
load_example_workspace()
```

### Value

A flowjo11\_workspace object

### Examples

```
ws <- load_example_workspace()
length(ws$samples) # Number of samples
```

---

```
map_compensation_names
```

*Map Compensation Channel Names to Cytoframe Parameter Names*

---

### Description

flowCore's compensation() function sanitizes channel names (e.g., "/" -> "\_", spaces -> "."). This function creates a mapping between sanitized compensation names and original cytoframe parameter names to ensure proper compensation application.

### Usage

```
map_compensation_names(comp_matrix, param_names)
```

### Arguments

|             |                                                |
|-------------|------------------------------------------------|
| comp_matrix | Compensation matrix (may have sanitized names) |
| param_names | Original parameter names from cytoframe        |

### Value

Compensation matrix with names matching cytoframe parameters

---

map\_gate\_params\_to\_gh *Map gate parameter names to GatingSet parameter names*


---

### Description

Gate parameter names may have "Comp-" prefix or different sanitization. This function maps them to the actual GatingSet parameter names.

### Usage

```
map_gate_params_to_gh(
  gate_params,
  gh_param_names,
  strip_comp_prefix = TRUE,
  sanitize_slashes = TRUE
)
```

### Arguments

**gate\_params** Character vector of gate parameter names

**gh\_param\_names** Character vector of GatingSet parameter names

**strip\_comp\_prefix** Logical. Strip "Comp-" prefix from gate parameter names? Default TRUE. Set to FALSE if compensation has already been applied with matching parameter names.

**sanitize\_slashes** Logical. Replace "/" with "\_" in parameter names? Default TRUE (matches flowCore behavior). Set to FALSE if you want to preserve "/" in marker names (e.g., "CD3/CD4" stays as-is).

### Value

Named list mapping gate params to GatingSet params

---

map\_param\_names *Unified Parameter Name Mapping Function*


---

### Description

Creates a mapping between source parameter names (e.g., from compensation, gates, transformations) and target parameter names (e.g., cytoframe flowFrame column names).

### Usage

```
map_param_names(
  source_names,
  target_names,
  target_descriptions = NULL,
  strip_comp_prefix = FALSE,
  case_insensitive = FALSE,
  sanitize_slashes = TRUE
)
```

**Arguments**

|                     |                                                                                                                                                                                                  |
|---------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| source_names        | Character vector of source parameter names                                                                                                                                                       |
| target_names        | Character vector of target parameter names (e.g. flowFrame colnames)                                                                                                                             |
| target_descriptions | Optional named character vector of descriptions for each target name (e.g. the marker names returned by markernames()). When supplied, source names are also matched against these descriptions. |
| strip_comp_prefix   | Logical. Strip "Comp-" prefix from source names? Default FALSE.                                                                                                                                  |
| case_insensitive    | Logical. Case-insensitive matching? Default FALSE.                                                                                                                                               |
| sanitize_slashes    | Logical. Replace "/" with "_" in parameter names? Default TRUE (matches flowCore behavior). Set to FALSE if you want to preserve "/" in marker names (e.g., "CD3/CD4" stays as-is).              |

**Details**

This function handles common naming discrepancies:

- "Comp-" prefix (e.g., "Comp-APC-Ax700-A" vs "APC-Ax700-A")
- "/" vs "" *sanitization (flowCore converts "/" to "")*
- Case sensitivity (optional)
- Detector-to-marker description matching (optional, via target\_descriptions)

**Value**

Named list mapping source names to target names. Unmapped names have NULL values.

---

map\_transformation\_names

*Map Transformation Channel Names to flowFrame Parameter Names*

---

**Description**

Transformation names from FlowJo may not match flowFrame parameter names:

- They may have "Comp-" prefix (e.g., "Comp-APC-Ax700-A" vs "APC-Ax700-A")
- They may use different sanitization (e.g., "/" vs "\_") This function creates a mapping between transformation names and flowFrame parameters.

**Usage**

```
map_transformation_names(trans_list, param_names)
```

**Arguments**

|             |                                                       |
|-------------|-------------------------------------------------------|
| trans_list  | Named list of transformations (names = channel names) |
| param_names | Original parameter names from flowFrame/cytoframe     |

**Value**

Named list of transformations with names matching flowFrame parameters

---

|               |                                        |
|---------------|----------------------------------------|
| null-coalesce | <i>Custom Null Coalescing Operator</i> |
|---------------|----------------------------------------|

---

**Description**

Provides a null coalescing operator that returns the second argument if the first is NULL.

**Usage**

```
x %||% y
```

**Arguments**

|   |                                         |
|---|-----------------------------------------|
| x | First value to check                    |
| y | Second value to return if first is NULL |

**Value**

Either x if not NULL, or y

---

|                          |                                 |
|--------------------------|---------------------------------|
| parse_boolean_expression | <i>Parse Boolean Expression</i> |
|--------------------------|---------------------------------|

---

**Description**

Parse Boolean Expression

**Usage**

```
parse_boolean_expression(spec)
```

---

|                         |                                |
|-------------------------|--------------------------------|
| parse_compensation_data | <i>Parse Compensation Data</i> |
|-------------------------|--------------------------------|

---

**Description**

Parse Compensation Data

**Usage**

```
parse_compensation_data(comp_data)
```

---

|                     |                                        |
|---------------------|----------------------------------------|
| parse_spill_keyword | <i>Parse SPILL Keyword into Matrix</i> |
|---------------------|----------------------------------------|

---

**Description**

Parse SPILL Keyword into Matrix

**Usage**

```
parse_spill_keyword(keywords)
```

**Arguments**

|          |                             |
|----------|-----------------------------|
| keywords | Named list of FCS keywords. |
|----------|-----------------------------|

**Value**

A matrix with dimnames, or NULL if no SPILL keyword.

---

|                           |                                  |
|---------------------------|----------------------------------|
| parse_transformation_info | <i>Parse Transformation Info</i> |
|---------------------------|----------------------------------|

---

**Description**

Parse Transformation Info

**Usage**

```
parse_transformation_info(trans_info)
```

**Details**

Parses transformation information from FlowJo v11 transformation specifications. Supports normalization of various transformation type names including biexponential (Biex) which maps to the biexponential type for proper handling.

---

|             |                                            |
|-------------|--------------------------------------------|
| populations | <i>Population Functions for FlowJo v11</i> |
|-------------|--------------------------------------------|

---

**Description**

Population Functions for FlowJo v11

---

|                     |                                      |
|---------------------|--------------------------------------|
| pretty_print_flowjo | <i>Pretty Print FlowJo Workspace</i> |
|---------------------|--------------------------------------|

---

**Description**

This function takes a FlowJo v11 workspace file (.flowjo) and extracts the JSON content, then pretty prints it to a text file with "\_pretty.json" suffix.

**Usage**

```
pretty_print_flowjo(flowjo_file)
```

**Arguments**

flowjo\_file      Path to the FlowJo workspace file (.flowjo)

**Value**

Path to the created pretty printed JSON file

**Examples**

```
# Pretty print a FlowJo workspace to readable JSON
ws_path <- system.file("extdata", "min_test.flowjo", package = "CyFj11")
out_file <- pretty_print_flowjo(ws_path)
file.exists(out_file) # Verify output was created
# Clean up
unlink(out_file)
```

---

|                     |                                       |
|---------------------|---------------------------------------|
| process_zip_archive | <i>Process FlowJo v11 ZIP Archive</i> |
|---------------------|---------------------------------------|

---

**Description**

Extracts and parses the contents of a .flowjo file:

- manifest.txt: File listing
- analysis-\*.json: Workspace data (gates, populations, samples)

**Usage**

```
process_zip_archive(zip_path)
```

**Arguments**

zip\_path          Path to the .flowjo file

**Value**

List containing manifest and parsed JSON data

---

```
read_flowjo11_workspace
```

*Read FlowJo v11 Workspace*

---

### Description

Main wrapper function to read and parse FlowJo v11 workspace files.

### Usage

```
read_flowjo11_workspace(workspace_path)
```

### Arguments

`workspace_path` Path to the FlowJo workspace file (.flowjo)

### Value

Parsed workspace object containing manifest and JSON data

### Examples

```
ws_path <- system.file("extdata", "min_test.flowjo", package = "CyFj11")
ws <- read_flowjo11_workspace(ws_path)
length(ws$samples) # Number of samples in workspace
names(ws$groups)   # Group names
```

---

```
resolve_all_fcs_paths
```

*Resolve All FCS File Paths from FlowJo Workspace*

---

### Description

Resolves all FCS file paths from a FlowJo workspace, mapping FlowJo URIs to actual file locations on the current system.

### Usage

```
resolve_all_fcs_paths(
  dataSources,
  root_dir,
  stop_on_multiple = FALSE,
  stop_on_missing = TRUE
)
```

### Arguments

`dataSources` Data sources from FlowJo workspace (from JSON)

`root_dir` Root directory to search for FCS files

`stop_on_multiple` Stop if any duplicate filenames found (default: FALSE)

`stop_on_missing` Stop if any files not found (default: TRUE)

**Value**

Data frame with columns: sample\_id, flowjo\_uri, filename, resolved\_path, status

---

|               |                                                       |
|---------------|-------------------------------------------------------|
| safe-coercion | <i>Safe Numeric/Integer Coercion Without Warnings</i> |
|---------------|-------------------------------------------------------|

---

**Description**

BiocCheck discourages `suppressWarnings(as.numeric(...))`; these helpers validate the string form with a regular expression first so that `as.numeric/as.integer` never produce a coercion warning while producing exactly the same result as the suppressed call.

**Usage**

```
as_num_quiet(x)
```

```
as_int_quiet(x)
```

**Arguments**

`x` Character vector to coerce

**Value**

Numeric/integer vector, NA where `x` does not denote a number

---

|                              |                                                               |
|------------------------------|---------------------------------------------------------------|
| sanitize_path_with_separator | <i>Sanitize Path While Preserving flowWorkspace Separator</i> |
|------------------------------|---------------------------------------------------------------|

---

**Description**

Population paths use `'/'` as the separator, but individual population names may contain `'/'` which must be replaced by `':'`. This helper sanitizes only the name components, leaving the path separator `'/'` intact.

**Usage**

```
sanitize_path_with_separator(path)
```

**Arguments**

`path` Character vector of path strings (e.g. "Ungated/Lymphocytes/CD45, l/d subset").

**Value**

Path strings with name-level `'/'` replaced by `':'`.

---

`sanitize_population_name`*Sanitize Population Name for flowWorkspace*

---

**Description**

flowWorkspace uses '/' as the path separator in population paths. Population names imported from FlowJo may contain '/' (e.g. "CD45, l/d subset"), which causes flowWorkspace to rewrite them inconsistently. This helper normalizes such names consistently before nodes are added to a GatingSet.

**Usage**

```
sanitize_population_name(name)
```

**Arguments**

|      |                                               |
|------|-----------------------------------------------|
| name | Character vector or list of population names. |
|------|-----------------------------------------------|

**Value**

Sanitized name of the same structure as the input.

---

`search_fcs_files`*Search for FCS files in directory tree*

---

**Description**

Builds an index of all FCS files in a directory tree.

**Usage**

```
search_fcs_files(root_dir, pattern = "\\*.fcs$")
```

**Arguments**

|          |                                                         |
|----------|---------------------------------------------------------|
| root_dir | Character vector of root directories to search          |
| pattern  | Character. File pattern to match (default: "\\*.fcs\$") |

**Value**

Data frame with columns: filename, full\_path, size\_bytes, mtime

---

|                      |                                                       |
|----------------------|-------------------------------------------------------|
| set_sanitize_slashes | <i>Set parameter name slash sanitization behavior</i> |
|----------------------|-------------------------------------------------------|

---

### Description

Control whether "/" characters in parameter/marker names are automatically replaced with "\_". *This is important because flowCore sanitizes parameter names by replacing "/" with "\_" when reading FCS files.*

### Usage

```
set_sanitize_slashes(v)
```

### Arguments

|   |                                                                        |
|---|------------------------------------------------------------------------|
| v | Logical. TRUE to replace "/" with "_" (default), FALSE to preserve "/" |
|---|------------------------------------------------------------------------|

### Details

By default, this is TRUE to match flowCore's behavior, ensuring that gate parameter names match the flowFrame column names. Set to FALSE if you want to preserve "/" in marker names (e.g., "CD3/CD4" stays as-is).

### Value

Invisible NULL. Called for side effects.

### Examples

```
# Preserve "/" in marker names
set_sanitize_slashes(FALSE)
# Use default behavior (replace "/" with "_")
set_sanitize_slashes(TRUE)
```

---

|             |                         |
|-------------|-------------------------|
| set_verbose | <i>Set verbose mode</i> |
|-------------|-------------------------|

---

### Description

Enable or disable verbose output for debugging and monitoring package operations. When enabled, the package will print detailed information about its operations, including file processing, gate conversion, and data extraction steps.

### Usage

```
set_verbose(v)
```

### Arguments

|   |                                                             |
|---|-------------------------------------------------------------|
| v | Logical. TRUE to enable verbose output, FALSE to disable it |
|---|-------------------------------------------------------------|

**Value**

Invisible NULL. Called for side effects.

**Examples**

```
set_verbose(TRUE)
set_verbose(FALSE)
```

---

|                 |                                                |
|-----------------|------------------------------------------------|
| transformations | <i>Transformation Functions for FlowJo v11</i> |
|-----------------|------------------------------------------------|

---

**Description**

Transformation Functions for FlowJo v11

---

|                         |                                                       |
|-------------------------|-------------------------------------------------------|
| update_gate_param_names | <i>Update gate parameter names to match flowFrame</i> |
|-------------------------|-------------------------------------------------------|

---

**Description**

After compensation, flowFrame parameter names may have "Comp-" prefix. This function updates gate parameter names to match.

**Usage**

```
update_gate_param_names(gate_obj, mapped_params)
```

**Arguments**

- gate\_obj            flowCore gate object
- mapped\_params    Named list mapping old param names to new param names

**Value**

Gate object with updated parameter names

---

|                       |                                     |
|-----------------------|-------------------------------------|
| validate_compensation | <i>Validate Compensation Object</i> |
|-----------------------|-------------------------------------|

---

**Description**

Validate Compensation Object

**Usage**

```
validate_compensation(comp)
```

---

verify\_gate\_marker\_names

*Verify Marker Name Consistency After Gate Creation*


---

### Description

This function checks that marker names in gates match the parameter names in the GatingSet/flowFrame after compensation and transformations have been applied. It helps detect issues where automated routines may have changed marker names.

### Usage

```
verify_gate_marker_names(gate_obj, flowframe_params, gate_source = "")
```

### Arguments

|                  |                                                                                                                                                                                        |
|------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| gate_obj         | A flowCore gate object (rectangleGate, polygonGate, etc.)                                                                                                                              |
| flowframe_params | Named character vector of parameter names from the flowFrame. Typically the output of markernames(gh), where names are detector/channel names and values are descriptive marker names. |
| gate_source      | Character. Description of gate source for error messages                                                                                                                               |

### Value

List with components:

**valid** Logical. Are all gate parameters mappable to flowFrame params?

**mapping** Named list of parameter name mappings

**warnings** Character vector of warning messages for unmapped params

---

write\_fcs\_files\_to\_dir

*Write FCS Files from a GatingSet to a Directory*


---

### Description

Copies (or re-exports) the FCS files backing a GatingSet to target\_dir. When the original file is accessible on disk it is copied verbatim so that all acquisition keywords are preserved exactly. If the original cannot be found, the flowFrame is extracted from the GatingSet with inverse transforms applied and written as a new FCS file.

### Usage

```
write_fcs_files_to_dir(gating_set, target_dir, overwrite = FALSE)
```

**Arguments**

|            |                                                                                                                      |
|------------|----------------------------------------------------------------------------------------------------------------------|
| gating_set | GatingSet object.                                                                                                    |
| target_dir | Destination directory (must already exist).                                                                          |
| overwrite  | Logical. FALSE (default) stops if any destination file already exists; TRUE replaces existing files after a warning. |

**Value**

Invisible character vector of file paths written successfully.

---

|            |                                      |
|------------|--------------------------------------|
| xml_encode | <i>XML Encode Special Characters</i> |
|------------|--------------------------------------|

---

**Description**

XML Encode Special Characters

**Usage**

```
xml_encode(text)
```

**Arguments**

|      |                |
|------|----------------|
| text | Text to encode |
|------|----------------|

**Value**

Encoded text

# Index

## \* internal

- add\_population\_node, [5](#)
- add\_populations\_to\_gatingset, [5](#)
- adjust\_gate\_transformations, [7](#)
- apply\_extension, [8](#)
- apply\_param\_mapping, [9](#)
- apply\_transforms\_to\_gate, [9](#)
- archive, [10](#)
- build\_gating\_tree, [11](#)
- build\_sample\_keywords, [11](#)
- build\_spillover\_matrix\_xml, [11](#)
- check\_wsp\_csv\_coverage, [12](#)
- compensation, [12](#)
- conversion, [13](#)
- convert\_boolean\_gate, [13](#)
- convert\_boolean\_to\_flowjo10, [13](#)
- convert\_ellipse\_gate, [14](#)
- convert\_ellipsoid\_to\_flowjo10, [14](#)
- convert\_flowjo\_gate, [15](#)
- convert\_gate\_to\_flowjo10\_format, [15](#)
- convert\_polygon\_gate, [16](#)
- convert\_polygon\_to\_flowjo10, [16](#)
- convert\_quadrant\_gate, [16](#)
- convert\_range\_gate, [17](#)
- convert\_rectangle\_gate, [17](#)
- convert\_rectangle\_to\_flowjo10, [17](#)
- create\_bisexponential\_transform, [18](#)
- create\_default\_groups\_v10, [18](#)
- create\_gatingset\_from\_cytoset, [18](#)
- create\_linear\_transform, [19](#)
- create\_log\_transform, [19](#)
- create\_sample\_names, [20](#)
- create\_transformation\_list, [20](#)
- CyFj11-package, [4](#)
- derive\_cytometer\_attrs, [21](#)
- display\_to\_raw, [21](#)
- emit\_transform\_xml, [22](#)
- export-flowjo10, [22](#)
- extract-flowjo-stats, [24](#)
- extract\_all\_gates, [24](#)
- extract\_compensation, [25](#)
- extract\_compensation\_from\_platforms, [25](#)
- extract\_flowjo\_stats, [26](#)
- extract\_gates\_from\_gatingset\_v10, [27](#)
- extract\_keywords\_for\_samples, [28](#)
- extract\_pdata, [28](#)
- extract\_populations\_from\_gatingset\_v10, [28](#)
- extract\_samples\_from\_gatingset\_v10, [29](#)
- extract\_transformation\_spec, [30](#)
- extract\_transformations, [29](#)
- extract\_workspace\_compensation, [30](#)
- filter\_samples, [30](#)
- find\_root\_population, [31](#)
- format\_gate\_num, [34](#)
- gates, [34](#)
- generate\_flowjo10\_xml, [34](#)
- generate\_group\_subpopulations\_xml, [35](#)
- generate\_logical\_node\_xml, [36](#)
- generate\_sample\_subpopulations\_xml, [36](#)
- get\_display\_range, [37](#)
- get\_fcs\_header\_keywords, [37](#)
- get\_graph\_axes, [38](#)
- get\_group\_info, [38](#)
- get\_referenced\_channels, [39](#)
- get\_sample\_file\_map, [39](#)
- get\_transform\_spec, [40](#)
- helpers, [41](#)
- helpers-conversion, [41](#)
- map\_compensation\_names, [42](#)
- map\_gate\_params\_to\_gh, [43](#)
- map\_param\_names, [43](#)
- map\_transformation\_names, [44](#)
- null-coalesce, [45](#)
- parse\_boolean\_expression, [45](#)
- parse\_compensation\_data, [45](#)
- parse\_spill\_keyword, [46](#)
- parse\_transformation\_info, [46](#)
- populations, [46](#)
- process\_zip\_archive, [47](#)

- resolve\_all\_fcs\_paths, 48
  - safe-coercion, 49
  - sanitize\_path\_with\_separator, 49
  - sanitize\_population\_name, 50
  - search\_fcs\_files, 50
  - transformations, 52
  - update\_gate\_param\_names, 52
  - validate\_compensation, 52
  - verify\_gate\_marker\_names, 53
  - write\_fcs\_files\_to\_dir, 53
  - xml\_encode, 54
- add\_population\_node, 5, 8
- add\_populations\_to\_gatingset, 5, 6, 7
- adjust\_gate\_transformations, 6, 7, 7, 10
- apply\_extension, 8
- apply\_param\_mapping, 9
- apply\_transforms\_to\_gate, 8, 9
- archive, 10
- as\_int\_quiet (safe-coercion), 49
- as\_num\_quiet (safe-coercion), 49
- build\_gating\_tree, 6, 7, 11
- build\_sample\_keywords, 11
- build\_spillover\_matrix\_xml, 11
- check\_wsp\_csv\_coverage, 12
- compensation, 12
- conversion, 13
- convert\_boolean\_gate, 13
- convert\_boolean\_to\_flowjo10, 13
- convert\_ellipse\_gate, 14
- convert\_ellipsoid\_to\_flowjo10, 14
- convert\_flowjo\_gate, 15
- convert\_gate\_to\_flowjo10\_format, 15
- convert\_polygon\_gate, 16
- convert\_polygon\_to\_flowjo10, 16
- convert\_quadrant\_gate, 16
- convert\_range\_gate, 17
- convert\_rectangle\_gate, 17
- convert\_rectangle\_to\_flowjo10, 17
- create\_biexponential\_transform, 18
- create\_default\_groups\_v10, 18
- create\_gatingset\_from\_cytoiset, 18
- create\_linear\_transform, 19
- create\_log\_transform, 19
- create\_sample\_names, 20
- create\_transformation\_list, 20
- CyFj11 (CyFj11-package), 4
- CyFj11-package, 4
- derive\_cytometer\_attrs, 21
- display\_to\_raw, 21
- emit\_transform\_xml, 22
- export-flowjo10, 22
- export\_flowjo10\_workspace, 23
- extract-flowjo-stats, 24
- extract\_all\_gates, 6–8, 10, 24
- extract\_compensation, 25
- extract\_compensation\_from\_platforms, 25
- extract\_flowjo\_stats, 26
- extract\_gates\_from\_gatingset\_v10, 27
- extract\_keywords\_for\_samples, 28
- extract\_pdata, 28
- extract\_populations\_from\_gatingset\_v10, 28
- extract\_samples\_from\_gatingset\_v10, 29
- extract\_transformation\_spec, 30
- extract\_transformations, 29
- extract\_workspace\_compensation, 30
- filter\_samples, 30
- find\_root\_population, 31
- fj11\_to\_gatingset, 23, 31
- format\_gate\_num, 34
- gates, 34
- generate\_flowjo10\_xml, 34
- generate\_group\_subpopulations\_xml, 35
- generate\_logical\_node\_xml, 36
- generate\_sample\_subpopulations\_xml, 36
- get\_display\_range, 37
- get\_fcs\_header\_keywords, 37
- get\_graph\_axes, 38
- get\_group\_info, 38
- get\_referenced\_channels, 39
- get\_sample\_file\_map, 39
- get\_sanitize\_slashes, 40
- get\_transform\_spec, 40
- get\_verbose, 41
- gh\_get\_transformations, 8–10
- gs\_pop\_add, 6–8
- helpers, 41
- helpers-conversion, 41
- load\_example\_workspace, 42
- map\_compensation\_names, 42
- map\_gate\_params\_to\_gh, 8, 43
- map\_param\_names, 43
- map\_transformation\_names, 44
- message, 6
- null-coalesce, 45

- parse\_boolean\_expression, [45](#)
- parse\_compensation\_data, [45](#)
- parse\_spill\_keyword, [46](#)
- parse\_transformation\_info, [46](#)
- populations, [46](#)
- pretty\_print\_flowjo, [47](#)
- process\_zip\_archive, [47](#)
  
- read\_flowjo11\_workspace, [48](#)
- resolve\_all\_fcs\_paths, [48](#)
  
- safe-coercion, [49](#)
- sanitize\_path\_with\_separator, [49](#)
- sanitize\_population\_name, [50](#)
- search\_fcs\_files, [50](#)
- set\_sanitize\_slashes, [51](#)
- set\_verbose, [51](#)
  
- transformations, [52](#)
  
- update\_gate\_param\_names, [8](#), [52](#)
  
- validate\_compensation, [52](#)
- verify\_gate\_marker\_names, [53](#)
  
- write\_fcs\_files\_to\_dir, [53](#)
  
- xml\_encode, [54](#)