

GATING-ML 2.0

International Society for Advancement of Cytometry (ISAC)
standard for representing gating descriptions in flow cytometry

Josef Spidlen, ISAC Data Standards Task Force, Ryan Brinkman*

Version 2.0 – 2015-03-16

Document Status

Gating-ML has undergone several revisions since the first public release in February 2006. This document is part of the Gating-ML version 2.0 specification. It is based on the Gating-ML version 1.5, which became an International Society for Advancement of Cytometry (ISAC) Candidate Recommendation in January 2008. Based on feedback gathered from the implementors and development in the field, the Gating-ML version 2.0 has been developed.

The current specification is an ISAC Recommendation that has been formally approved by the ISAC Data Standards Task Force and by ISAC Council on January 13, 2015. At this point, reference implementations have been developed and the specification has been formally tested to comply with the W3C XML schema version 1.0 specification, but no position is taken in respect to whether a particular software implementing this specification performs according to medical or other valid regulations.



The work may be used under the terms of the Creative Commons Attribution-ShareAlike 3.0 Unported license. You are free to share (copy, distribute and transmit), and adapt the work under the conditions specified at <http://creativecommons.org/licenses/by-sa/3.0/legalcode>.

Patent Disclaimer

Attention is called to the possibility that implementation of this standard may require use of subject matter covered by patent rights. By publication of this standard, no position is taken with respect to the existence or validity of any patent rights in connection therewith. ISAC shall not be responsible for identifying patents or patent applications for which a license may be required to implement an ISAC standard or for conducting inquiries into the legal validity or scope of those patents that are brought to its attention.

*List of ISAC Data Standards Task Force members provided in Annex C.

To assure the highest-level of transparency and neutrality ISAC requires that all members of the Flow Cytometry Data Standards Task Force adhere to the Patent Disclosure Policy set forth below. ISAC undertakes to establish standards that do not require the purchase of patent licenses for compliance. For this reason, Task Force members must immediately disclose any patents or patent applications held by them and which they know or have reason to believe have a likelihood of being infringed by compliance with any draft or approved ISAC Recommendation. Users and reviewers are also encouraged to disclose any procedures that require a user to purchase a patent license. In addition, by participating in any ISAC Task Force, members affirm that, with respect to any patent that may be infringed by compliance with any draft or approved ISAC Recommendation and is not disclosed, such patent shall either (1) not be enforced with respect to such compliance, or (2) shall be freely licensed to all users on a fair, reasonable and non-discriminatory basis and without imposing a licensing fee or other charge of any kind. Every Task Force member shall at all times act in good faith and in an open and honest manner.

Abstract

The Flow Cytometry Standard (FCS) specification has been adopted for the common representation of flow cytometry data, and this standard is supported by all analytical instrument and third party software suppliers. It includes data, metadata, and analysis components within the same file. However, metadata and analysis components, if included at all, are not recorded in a standardized fashion or in sufficient detail for use by independent parties. Also, the FCS standard is not sufficient to accommodate for novel flow cytometry data, such as streaming or correlated waveform data or data combining the visual power of microscopy with the statistical rigor of flow cytometry.

We propose to address these shortcomings via series of related data standards. This document represents a specification on how to form unambiguous XML-based gate definitions. Such a description of gates can facilitate the interchange and validation of data between different software packages with the potential for significant increase of hardware and software interoperability. The specification supports rectangular gates in n dimensions (*i.e.*, from one-dimensional range gates up to n -dimensional hyper-rectangular regions), quadrant gates in n dimensions, polygon gates, ellipsoid gates in n dimensions, and Boolean collections of any of the types of gates. Gates can be uniquely identified and may be ordered into a hierarchical structure to describe a gating strategy. Gates may be applied on list mode data files (*e.g.*, FCS files), which may be transformed as explicitly described. Transformations including compensation description are included as part of the Gating-ML specification.

Keywords

flow cytometry, analytical cytology, gating, classification, data transformation, compensation, scale, spillover, spectrum, XML

Contents

List of Figures	6
List of Tables	6
1 Overview	9
1.1 Introduction	9
1.2 Scope	9
1.3 Purpose	10
1.4 Normative references	10
1.5 The content of this specification	11
1.6 Namespaces and their prefixes within this document	11
2 Design principles and rationale	12
2.1 Supported gate and transformation types	12
2.2 Computationally simple determination of gate membership	13
2.3 Inclusion of transformations within Gating-ML	13
2.3.1 Motivation	13
2.3.2 Effects of nonlinear transformations	13
2.3.3 Transforming the whole gate description into the data space	14
2.4 Maximal utilization of XML schema validation	16
2.5 Significant XML sibling elements order	17
3 Terminology and requirements	18
3.1 Acronyms and abbreviations	18
3.2 Mathematical symbols used	18
3.3 Terminology	18
3.3.1 FCS dimension	18
3.3.2 Electronic event	20
3.3.3 List mode data file	20
3.3.4 Event value	20
3.3.5 Spillover, or spectrum, coefficient and matrix	21
3.3.6 Dependency	21
3.3.7 Dependency graph	21
3.3.8 Data transformation	21
3.3.9 Scale transformation	22
3.4 Conformance	22
3.4.1 File conformance	22
3.4.2 Applicability	22
3.4.3 Software and hardware conformance	22
4 General	23
4.1 Referencing dimensions	23
4.2 FCS dimensions and transformations	24
4.2.1 Gates applicable directly on uncompensated list mode data files	24
4.2.2 Gates applicable on compensated data	25

4.2.3	Gates applicable on data on transformed scales	26
4.2.4	Gates applicable on the ratio of two FCS dimensions	27
4.2.5	Gates applicable on scaled ratio of two FCS dimensions	29
4.3	Results of gating operations	29
4.4	Hierarchical gating and gating strategy description	30
4.5	Gating-ML file structure	30
4.6	Gates overview	31
4.7	Scale transformations overview	31
4.8	Additional custom information	32
5	Gate description	33
5.1	Rectangular and range gates	33
5.1.1	Definition	33
5.1.2	Syntax specification	35
5.1.3	Validity conditions	36
5.1.4	Examples	36
5.2	Polygon gates	38
5.2.1	Definition	38
5.2.2	Syntax specification	40
5.2.3	Validity conditions	41
5.2.4	Examples	41
5.3	Ellipsoid gates	44
5.3.1	Definition	44
5.3.2	Syntax specification	47
5.3.3	Validity conditions	48
5.3.4	Examples	49
5.4	Quadrant gates	49
5.4.1	Definition	49
5.4.2	Syntax specification	52
5.4.3	Validity conditions	54
5.4.4	Examples	55
5.5	Boolean gates	56
5.5.1	Definition	56
5.5.2	Syntax specification	58
5.5.3	Validity conditions	59
5.5.4	Examples	59
6	Scale transformation description	61
6.1	Bounding transformations	61
6.1.1	Definition	61
6.1.2	Use case description	62
6.1.3	Syntax specification	62
6.1.4	Validity conditions	62
6.1.5	Examples	62
6.2	Parametrized linear transformation – <i>flin</i>	64
6.2.1	Definition	64

6.2.2	Syntax specification	65
6.2.3	Validity conditions	65
6.2.4	Examples	65
6.3	Parametrized logarithmic transformation – <i>flog</i>	66
6.3.1	Definition	66
6.3.2	Syntax specification	67
6.3.3	Validity conditions	67
6.3.4	Examples	67
6.4	Parametrized inverse hyperbolic sine transformation – <i>fasinh</i>	69
6.4.1	Definition	69
6.4.2	Syntax specification	69
6.4.3	Validity conditions	70
6.4.4	Examples	70
6.5	Logicle transformation	71
6.5.1	Definition	71
6.5.2	Syntax specification	73
6.5.3	Validity conditions	74
6.5.4	Examples	74
6.6	Hyperlog transformation	76
6.6.1	Definition	76
6.6.2	Syntax specification	77
6.6.3	Validity conditions	78
6.6.4	Examples	78
7	Compensation description	80
7.1	Definition	80
7.2	Syntax specification	80
7.3	Semantic specification	81
7.4	Validity conditions	81
7.5	Examples	83
7.6	Compensation computing algorithm	84
7.6.1	Standard compensation based on square spectrum (spillover) matrices	84
7.6.2	Compensation based on non-square spectrum matrices	85
8	Additional transformations description	86
8.1	Parametrized ratio transformation – <i>fratio</i>	86
8.1.1	Definition	86
8.1.2	Syntax specification	86
8.1.3	Validity conditions	87
8.1.4	Examples	87
A	References	90
B	Summary of changes since Gating-ML 1.5	92
C	ISAC Data Standards Task Force Members	94

List of Figures

1	A polygon gate created in a transformed space.	14
2	The effect of a logarithmic transformation on the shape of the gate.	14
3	Example of a gate “changing dimensionality” in data space.	15
4	Example of a gate stored as a bitmap image.	16
5	Work flow of data transformations prior to the application of a gate.	25
6	Comparison of transformations with $T = 1000$, $M = 4.5$, $W = 1$, and $A = 0$. . .	33
7	Comparison of transformations with $T = 1000$, $M = 4$ or 5 , $W = 1$ and $A = 1$. .	34
8	Example of a range gate.	37
9	Example of a range gate.	37
10	Example of a rectangular gate.	38
11	Example of a 3-dimensional rectangular gate with one unbound dimension. . . .	39
12	Winding number method used to determine the interior of non-simple polygons. .	40
13	Example of a triangular polygon gate.	42
14	Example of a pentagon polygon gate.	43
15	Example of a polygon gate with crossing boundaries.	44
16	Ellipse with major and minor axes in line with the coordinate system.	45
17	Example of a two dimensional ellipsoid gate.	50
18	Example of a rotated two dimensional ellipsoid gate.	51
19	Demonstration of quadrant gates definitions.	53
20	Example of a one dimensional quadrant gate.	55
21	Example of a quadrant gate with “merged” quadrants.	58
22	Example of a Boolean OR gate.	61
23	The effect of transformation boundary as described section 6.1.5 (a).	64
24	The effect of increasing the W parameter in the Logicle transformation.	73
25	The effect of increasing the W parameter in the Hyperlog transformation.	77

List of Tables

1	XML namespaces used in this specification.	11
2	Space requirements for gates stored by bitmaps.	17
3	Acronyms and abbreviations used in this specification.	18
4	Mathematical symbols used in this specification.	19
5	Sample values of the <i>flin</i> transformation with various T and A	66
6	Sample values of the <i>flog</i> transformation with various T and M	68
7	Sample values of the <i>fasinh</i> transformation with various T , M and A	70
8	Sample values of the <i>Logicle</i> transformation with various T , W , M , and A	75
9	Sample values of the <i>Hyperlog</i> transformation with various T , W , M , and A . . .	79
10	Example of a simple 3×3 spillover matrix.	84
11	Example of a simple 2×3 spectrum matrix.	84
12	Sample values of the <i>fratio</i> transformation with various A , B , and C	87

List of Examples

1	Referencing FCS dimensions in gate definitions.	23
2	Referencing the ratio of two FCS dimensions in gate definitions.	24
3	Gate applicable directly on uncompensated scale values of a list mode data file.	26
4	Gate applicable on scale values compensated as defined in the FCS file.	27
5	Gate applicable on scale values compensated as defined in Gating-ML.	27
6	Gate applicable on scale values un-mixed as defined in Gating-ML.	28
7	Gate applicable on data on a Logicle scale.	28
8	Referencing scaled ratio of two FCS dimensions in a gate definition.	29
9	Gating-ML XML file demonstrating proper header definition.	31
10	Custom_info element at the top level of a Gating-ML file.	34
11	Range gate definition.	36
12	Open range gate definition.	37
13	Rectangular gate definition.	38
14	Three-dimensional hyper-rectangular gate definition.	39
15	Simple polygon gate definition.	42
16	Polygon gate definition with transformation.	43
17	Polygon gate definition with crossing boundaries.	44
18	Two-dimensional ellipsoid gate definition.	49
19	Rotated two-dimensional ellipsoid gate definition.	50
20	Three-dimensional ellipsoid gate definition.	52
21	Quadrant gate definition splitting the FSC-H dimension at multiple values.	56
22	One dimensional quadrant gate definition.	57
23	Quadrant gate definition demonstrating “merging” of quadrants.	57
24	Boolean OR gate definition combining a rectangle gate with an ellipse gate.	60
25	Boolean NOT gate definition.	60
26	Boolean AND gate definition with one operand used as complement.	61
27	Logicle transformation with a boundary restricting its result to the [0.1,0.9] interval.	62
28	Polygon gate referencing a transformation with a boundary.	63
29	Inverse hyperbolic sine transformation with a left boundary.	63
30	Ratio transformation with a boundary restricting its result to the [0.2, 5] interval.	64
31	Parametrized linear transformation definition with $T = 1000$ and $A = 0$	65
32	Parametrized linear transformation definition with $T = 1000$ and $A = 100$	66
33	Parametrized linear transformation definition with $T = 1024$ and $A = 256$	66
34	Parametrized linear transformation definition with a boundary.	66
35	Parametrized logarithmic transformation definition with $T = 10000$ and $M = 5$	67
36	Parametrized logarithmic transformation definition with $T = 1023$ and $M = 4.5$	68
37	Parametrized logarithmic transformation definition with $T = 262144$ and $M = 4.5$	68
38	Parametrized logarithmic transformation definition with a boundary.	68
39	Inverse hyperbolic sine transformation with $T = 1000$, $M = 4$ and $A = 1$	70
40	Inverse hyperbolic sine transformation with $T = 1000$, $M = 5$ and $A = 0$	71
41	Inverse hyperbolic sine transformation with $T = 1000$, $M = 3$ and $A = 2$	71
42	Logicle transformation definition with $T = 1000$, $W = 1$, $M = 4$ and $A = 0$	75
43	Logicle transformation definition with $T = 1000$, $W = 1$, $M = 4$, and $A = 1$	75
44	Logicle transformation definition with $T = 1000$, $W = 0$, $M = 4$, and $A = 1$	75

45 Hyperlog transformation definition with $T = 1000$, $W = 1$, $M = 4$ and $A = 0$ 78

46 Hyperlog transformation definition with $T = 1000$, $W = 1$, $M = 4$, and $A = 1$ 79

47 Hyperlog transformation definition with $T = 1000$, $W = 0$, $M = 4$, and $A = 1$ 79

48 Hyperlog transformation definition with a boundary. 79

49 Definition of a spectrum (spillover) matrix. 82

50 A simple 3×3 spillover matrix definition. 83

51 A simple 2×3 spectrum matrix definition. 84

52 Parametrized ratio transformation with $A = 1$, $B = 0$, and $C = 0$ 87

53 Parametrized ratio transformation with $A = 1$, $B = 5$ and $C = 5$ 88

54 Parametrized ratio transformation with $A = 0.5$, $B = -10$, and $C = 25$ 88

1 Overview

1.1 Introduction

In flow cytometry, gating is a well-known and highly important process for sorting and selecting populations of interests for further data acquisition or analysis. A standard formal way of exchanging unambiguous descriptions of gates is crucial for interoperability among analytical hardware and software applications. A gating description is also required by the Minimum Information for a Flow Cytometry Experiment [1] recommendation.

Within this document we demonstrate how to form unambiguous XML-based gate definitions. Gating-ML description of gates can facilitate the interchange and validation of data between different software packages with the potential for significant increase of hardware and software interoperability. The specification supports rectangular gates in n dimensions (*i.e.*, from one-dimensional range gates up to n -dimensional hyper-rectangular regions), quadrant gates in n dimensions, polygon gates, ellipsoid gates in n dimensions, and Boolean collections of any of the types of gates. Gates can be uniquely identified and may be ordered into a hierarchical structure to describe a gating strategy. Gates may be applied on events as in list mode data files (*e.g.*, scale values from the FCS files) or on transformed events as described by an explicit scale transformation. If a data analysis involves transformed values, the transformation needs to be exactly described in order to reconstruct the analysis. Therefore, scale transformation and compensation description are included as part of this specification.

The support for the Gating-ML specification as well as other related standards by software tools, journals, and scientists will significantly facilitate the reproduction of experiments and clinical measurements. Most importantly, these changes will allow scientists and software agents to search, automatically process, and in particular understand both flow cytometry data and metadata.

1.2 Scope

This document provides detailed specifications on how to form flow cytometry gate definitions using XML technology in order to computationally exchange details about post-acquisition analysis. The gate definitions are not primarily intended to define data acquisition or physical sorting gates. Methodology chosen to describe gates in XML may not be optimal for instrument acquisition or sorting settings.

Gates may be applied on raw data as in list mode data files or on data displayed using various scales. Data compensation as well as other transformations may be performed prior to applying gates. If analyses involve transformed data, then the transformation needs to be exactly described in order to reconstruct the analyses. For example, if a gate is created using nonlinearly transformed event values, the transformation needs to be described in order to reconstruct the gate. Therefore, scale transformation and compensation are included as part of this specification; see section 2.3 for more details.

This document does not cover guidelines (protocols, SOPs) on how gates ought to be formed in order to define specific populations, *i.e.*, how to create gates or what gating strategies ought to be used for particular assays.

1.3 Purpose

Gating in flow cytometry is a well-known and highly important process for selecting populations of interest by defining the characteristics of particles for further data acquisition or analysis. Although flow cytometry has a successful data format standard [2], it does not include a full representation of gates. This prevents a variety of collaborative opportunities to recreate experimental methods and results.

The purpose of this document is to standardize the description of flow cytometry gate definitions. It facilitates cross-platform sharing of gates, enables the interchange of gate definitions between different software packages, and provides the means to integrate methods with results in flow cytometry reporting.

1.4 Normative references

The following referenced documents are indispensable for the application of this standard. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments or corrigenda) applies.

- W3C Recommendation, Extensible Markup Language (XML) 1.0 (Fourth Edition) [3].
- W3C Recommendation, Namespaces in XML 1.0 (Third Edition) [4].
- W3C Recommendation, XML Schema 1.1 Part 1: Structures [5].
- W3C Recommendation, XML Schema 1.1 Part 2: Datatypes [6].

The following documents are useful for the application of this standard. They represent other standards and standard proposals relevant for understanding of this specification or intended to describe or store flow cytometry data and meta-data together with information about flow cytometry experiments and analyses.

- Spidlen J, Moore W, Parks D, Goldberg M, Bray C, Bierre P, Gorombey P, Hyun B, Hubbard M, Lange S, Lefebvre R, Leif R, Novo D, Ostruszka L, Treister A, Wood J, Murphy RF, Roederer M, Sudar D, Zigon R and Brinkman RR. Data File Standard for Flow Cytometry, version FCS 3.1 [2].
- Lee J, Spidlen J, Boyce K, Cai J, Dalphin M, Gasparetto M, Goldberg M, Jansen K, Kong M, Nikolich-Zugich J, Moloshok T, Parrish D, Qian Y, Selvaraj B, Smith C, Tchuvatkina O, Wilkinson P, Wilson C, Scheuermann R, and Brinkman R. MIFlowCyt: Minimum Information about a Flow Cytometry Experiment [1].
- Seamer LC, Bagwell CB, Barden L, Redelman D, Salzman GC, Wood JC, and Murphy RF. Proposed new data file standard for flow cytometry, version FCS 3.0 [7].

1.5 The content of this specification

This specification consists of the following parts:

- (a) **Normative:** This document providing a detailed description of the Gating-ML specification.
- (b) **Normative:** The XML schemas `Gating-ML.v2.0.xsd`, `Transformations.v2.0.xsd`, and `DataTypes.v2.0.xsd` defining syntax of Gating-ML compliant files and usable to validate Gating-ML XML documents. Within this document, the `Gating-ML.v2.0.xsd`, `Transformations.v2.0.xsd`, and `DataTypes.v2.0.xsd` files are referenced as the XML schemas or the schemas. The XML schemas can also be obtained from the following URLs:
 - <http://flowcyt.sourceforge.net/gating/2.0/xsd/Gating-ML.v2.0.xsd>
 - <http://flowcyt.sourceforge.net/gating/2.0/xsd/Transformations.v2.0.xsd>
 - <http://flowcyt.sourceforge.net/gating/2.0/xsd/DataTypes.v2.0.xsd>
- (c) **Informative:** Examples of Gating-ML files.
- (d) **Informative:** Gating-ML software compliance tests – a set of tests to determine Gating-ML compliance of third party software.
- (e) **Informative:** Gating-ML reference implementation.

All the components of this standard are available from World Wide Web at <http://www.isac-net.org/>. The specification may also be downloaded from <http://flowcyt.sourceforge.net/>.

1.6 Namespaces and their prefixes within this document

Table 1 lists XML namespaces [4] used within the XML schemas and XML examples in this specification. Please note that a valid Gating-ML document may use different prefixes for any of these namespaces. The XML schemas for the `gating`, `transforms` and `data-type` namespaces can also be downloaded from URLs as specified in section 1.5 (b) above.

Prefix	Namespace
<i>none</i> or <code>xs</code>	http://www.w3.org/2001/XMLSchema
<code>xsi</code>	http://www.w3.org/2001/XMLSchema-instance
<code>gating</code>	http://www.isac-net.org/std/Gating-ML/v2.0/gating
<code>transforms</code>	http://www.isac-net.org/std/Gating-ML/v2.0/transformations
<code>data-type</code>	http://www.isac-net.org/std/Gating-ML/v2.0/datatypes

Table 1: XML namespaces used in this specification.

2 Design principles and rationale

2.1 Supported gate and transformation types

This specification supports four types of gates: rectangular gates in n dimensions (*i.e.*, from one-dimensional range gates up to n -dimensional hyper-rectangular regions), quadrant gates in n dimensions, polygon gates (in 2 dimensions), ellipsoid gates in n dimensions, and Boolean collections of any of the types of gates. Supported gate types have been selected based on feedback on Gating-ML 1.5 in order to keep the specification simple while accommodate future innovations in automated multidimensional gating and clustering in a generic way.

Gates may be applied on data as in list mode data files or on transformed data. If gates are applied on transformed data then the exact description of data transformation needs to be provided in order to reconstruct the analysis. This specification supports open transformations (*i.e.*, published and free to use) which have been shown useful for display or analysis of cytometry data, such as *Logicle* and *Hyperlog*. In addition, transformations such as linear, logarithmic, and inverse hyperbolic sine are supported and have been extended to allow for additional parameterization and tweaking specifically for the display of flow cytometry data. These extensions are called *FLin*, *FLog* and *FASinH*, respectively. Finally, a parametrized ratio of two FCS dimensions (*i.e.*, *FRatio*) and fluorescence *compensation* complete the list of supported transformations.

Compared to Gating-ML 1.5, the list of supported transformations has been shortened by omitting transformations that have not been found particularly useful or are no longer necessary due to additional design changes. Inverse transformations are no longer automatically included. In addition, values from FCS files [2] are referenced as *scale values* (used to be *channel values* in Gating-ML 1.5), which eliminates the necessity to encode the channel-to-scale transformation in Gating-ML (this transformation is unambiguously captured by keywords in the FCS data file standard). Finally, Gating-ML 2.0 no longer supports compound transformations in general. Instead, each gate dimension can be defined referencing up to one “scale” transformation plus an optional fluorescence *compensation* description applied on an FCS dimension, which may be an FCS dimension from a list mode data file or the result of an additional transformation, *i.e.*, the ratio of two FCS dimensions. All these changes have been made based on community feedback in order to significantly simplify the Gating-ML specification, especially for Gating-ML consumers (readers). Consequently, the full implementation of Gating-ML 2.0 is significantly simpler than Gating-ML 1.5, which will facilitate interoperability among flow cytometry data analysis tools.

Also based on community feedback, this specification does not include any transformations or other components that may require use of subject matter covered by patents and may only be available under restrictive licensing conditions to some groups. However, by publication of this standard, no position is taken with respect to the existence or validity of any patent rights in connection therewith. ISAC shall not be responsible for identifying patents or patent applications for which a license may be required to implement an ISAC standard or for conducting inquiries into the legal validity or scope of those patents that are brought to its attention.

Originally, we have investigated the possibilities of reusing MathML [8] to universally describe any kind of mathematical transformation. While this would be a very flexible solution, it is generally impossible to evaluate expressions describable by MathML. Therefore, we have abandoned the MathML-based approach.

2.2 Computationally simple determination of gate membership

The specification has been designed with respect to computational simplicity of gate processing. Specifically, the description of gates has been designed to be able to make computationally simple decisions whether a particular event is in a particular gate. This aspect can especially be identified in the description of ellipsoid gates using covariance matrices (see section 5.3).

2.3 Inclusion of transformations within Gating-ML

This section describes the rationale for inclusion of transformations in Gating-ML and for description of gates in the space where they have been created rather than in the original data space.

2.3.1 Motivation

– *Why do we care about transformations when describing gates?*

In most flow cytometry applications, fluorescence signals of interest can range from high values down to essentially zero. Depending on the signal evaluation system, primary measurements may or may not include negative data values. After fluorescence compensation, some cell populations will have low means and will include events with negative data values. Logarithmic presentation has been very useful in providing informative displays of wide-ranging flow cytometry data, and several other transformations have been introduced to adequately display cell populations with low means and high variances and, in particular, negative data values. Typically, transformations are performed to store fluorescence data more efficiently, to provide more complete, appropriate, and readily interpretable data representations, or to achieve optimal display of that data.

If transformed data are used for analysis (*e.g.*, data are viewed on a transformed scale while creating a gate), the transformation needs to be exactly reported in order to unambiguously interpret or reconstruct the analysis. As gating description is highly dependent on being able to describe these transformations, we have placed these into a single specification. Below we state the rationale for the inclusion of transformations in Gating-ML and for designing the specification in a way that gates are described in the space where they have been created rather than transformed into the space where the acquired list mode data is stored.

2.3.2 Effects of nonlinear transformations

– *Why not to only transform the vertices of the gate into the data space and store these?*

Let us demonstrate the effects of a nonlinear transformation on the shape of a gate by a simple example, which shows the difference between recording a transformation versus transforming the coordinates of gate vertices and reporting these.

The example involves two variables: x and y ; the data are stored in a linear space. The variable x is transformed from linear to logarithmic scale and it is visualized using logarithmic scale; the variable y remains on the linear scale. Using this visualization (transformed space), a polygon gate has been created: vertices at (1, 10), (3, 30), and (1, 30), Figure 1.

However, if we would create a polygon gate in the original space using the corresponding coordinates of the gate vertices in the linear data space, *i.e.*, the points (10, 10), (1000, 30),

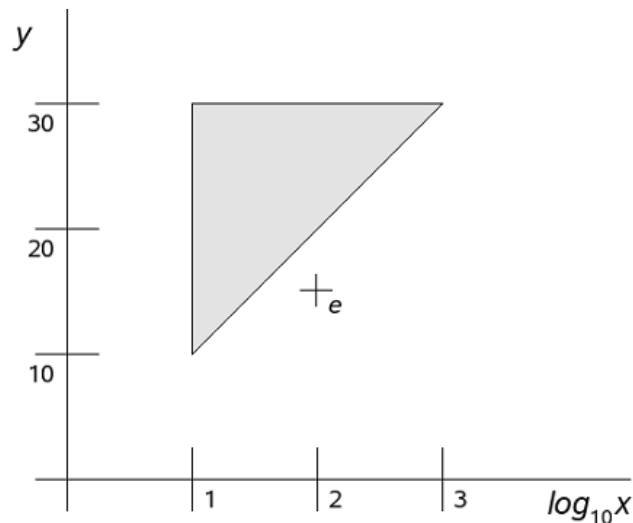


Figure 1: A polygon gate created in a transformed space.

and (10, 30), the event e with coordinates (100, 15) would fall in that gate (Figure 2). Such a naïve approach would create a significantly different gate. Figure 2 also demonstrates the effect of a logarithmic transformation on the shape of the gate.

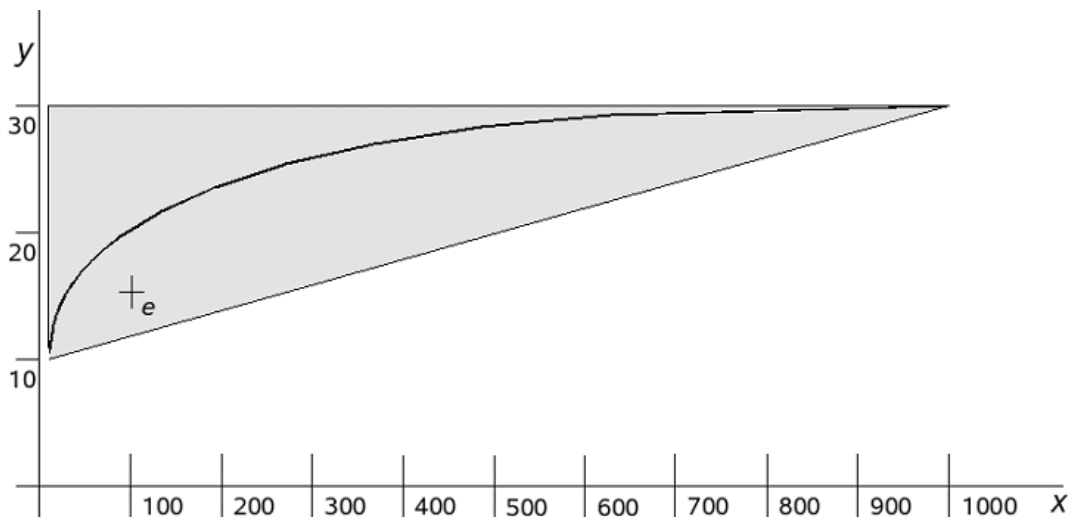


Figure 2: The effect of a logarithmic transformation on the shape of the gate.

2.3.3 Transforming the whole gate description into the data space

– *Why not to transform the whole gate into the data space?*

Above, we have shown why the naïve approach transforming only vertices does not work. Here we demonstrate two approaches of transforming “the whole” gate description into the data space and we explain why neither of these approaches has been used in Gating-ML.

Alternative Approach 1: Transform gate vertices into the data space and describe (using mathematical formulas) how edges “have been curved”.

In the example stated above, we would obtain a “polygon-like gate with vertices (10, 10), (1000, 30), and (10, 30), where the edge from (10, 10) to (1000, 30) is a logarithmic curve”.

Rationale for avoiding this approach

- This description (formulas) would be significantly difficult to create. Our example shows a two dimensional gate with only one dimension affected by the transformations. The “curving” gets complicated when a more complicated transformation is used (Logicle, Hyperlog, ...) or when both the dimensions are transformed, which is often the case. Moreover, multidimensional gates would have to be described by “curved” half-spaces, which would become challenging.
- Some transformations are irreversible (*i.e.*, they are not described by a uniquely invertible function). These include functions $f: \mathbb{R}^n \rightarrow \mathbb{R}$ where $n > 1$, *e.g.*, the ratio. These transformations can “change the dimensionality” of the gate. For example, a simple range gate “FL1-H/FL2-A ≥ 1 ” would translate into a gate as shown in Figure 3. While this is a very simple example chosen to graphically demonstrate the issue, description of these gates could become challenging in general cases.

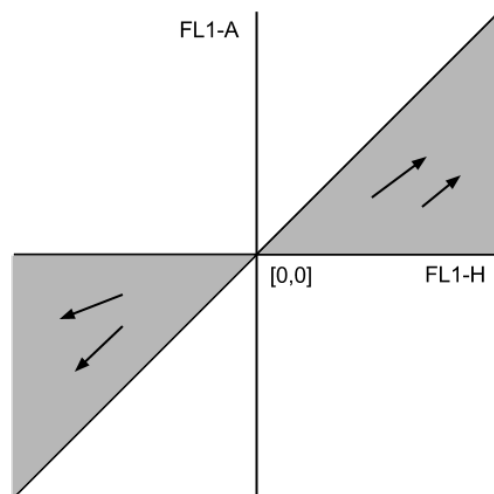


Figure 3: Example of a gate “changing dimensionality” in data space.

Alternative Approach 2: Storing a bitmap of the gate in data space

For each d -dimensional gate, we would create a d -dimensional hypercube; each dimension with the resolution corresponding to the range (*i.e.*, the value of the $\$PnR$ keyword [7]) of the corresponding FCS channel in the data file. For each combination of FCS values, we would store 1 bit of information (in/out), which unambiguously encodes the gate. Figure 4 shows a 2-dimensional example of a gate stored as a bitmap image.

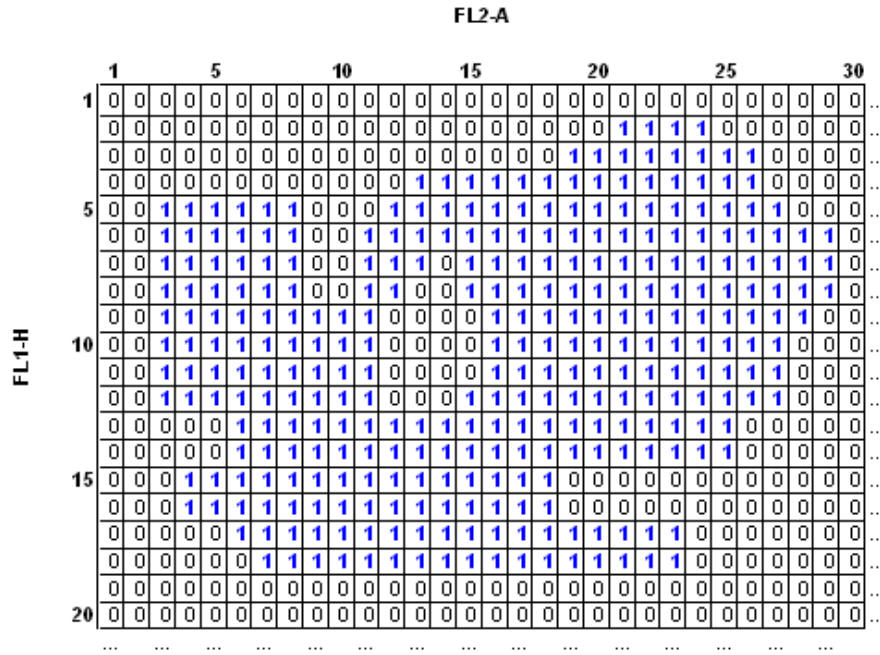


Figure 4: Example of a gate stored as a bitmap image.

Advantages of this approach

- Gates of any shape or dimensionality can be described in a uniform way in the data space (also independently on the space where these gates have been created).
- Event membership is very easy and fast to compute.

Disadvantages of this approach (*i.e.*, rationale for avoiding this approach)

- The approach is applicable for a discrete number of FCS values only. Contemporary instruments tend to provide fluorescence intensities as floating point values on linear scale. The bitmap approach would be difficult to use for floating point data. The data would have to be categorized into a discrete number of bins (*i.e.*, assign the closest “gate-bin”) and gated based on such an assignment. However, low vs. high values would have different precision (“bin-width”) requirements. This would again introduce the need of describing transformations to store gate bitmaps, which cancels out potential pros.
- The space requirements to store a gate are exponential in the number of dimensions involved in gate definitions. While this may be suitable for most traditional 2-dimensional gates, it does not allow for multidimensional gates, which are expected to become part of common gating strategies in the future. Table 2 shows the calculated space requirements.

2.4 Maximal utilization of XML schema validation

Ideally, a Gating-ML description would be valid if and only if the XML file was valid according to the Gating-ML XML schema. In most environments, platforms, and languages a developer

Resolution	Dimensions	Space required
256	2	8.19 kB
256	3	2.10 MB
256	4	536.87 MB
256	5	137.44 GB
1024	2	131.07 kB
1024	3	134.22 MB
1024	4	137.44 GB
65536	2	536.87 MB
65536	3	35.18 TB

Table 2: Space requirements for gates stored by bitmaps.

would just call an XML schema validator and would instantly know if the XML file is valid according to the Gating-ML standard. Unfortunately, this is not possible completely (see section 3.4.1) as for some necessary conditions that are too complicated to be expressed formally in XML schema (*e.g.*, not permitting circular definitions in Boolean gates). However, additional validity conditions have been kept on a minimal level.

2.5 Significant XML sibling elements order

Sibling elements within Gating-ML are explicitly considered to be ordered. This applies for example for the order of vertices within a polygon gate description. Sibling elements of different types are considered to be ordered within XML (*i.e.*, they are specified within the XML-schema sequence element and validation would fail if the order changes). Sibling elements of the same type are typically also considered to be ordered within XML; however, this cannot be verified by XML schema validation and XML specification [3] does not exactly specify interpretation by third party processing software.

According to our testing, this convention is maintained by the XML community, in support of the explicit convention adopted within Gating-ML. We have experimentally verified that the Java Architecture for XML Binding [9] implementation maintains the order of sibling elements. The Gating-ML 1.5 R/Bioconductor [10] reference implementation uses the XML package for R, which also keeps the element order as in the file. According to the DOM specification [11], one can access elements via the ordered NodeList interface. The elements are returned in the order in which they are encountered in a pre-order traversal of the document tree, *i.e.*, sibling elements keep the order when enumerating via the nextSibling method. Based on the nature of how SAX [12] parsers work, these keep the order of sibling elements and can as well be utilized for processing of Gating-ML. XPath [13] and therefore both XSLT [14] transformation and XPointer [15] addressing can be utilized for processing Gating-ML (*e.g.*, via XPath syntax such as following-sibling, preceding-sibling). The Qt XML parser (version 4.1) has also been confirmed as a compliant XML parser.

If other software is used for processing Gating-ML, the explicit requirement for maintenance of sibling order shall be verified (and eventually ensured, *e.g.*, by XSLT). This may apply to some XML-enabled databases or XML servers. For example, database rows are not considered to be ordered in a relational database model; thus, a naïve transformation of Gating-ML into a

database structure could eventually change the content of the Gating-ML document. However, based on our research, several database servers (*e.g.*, Oracle, IBM DB2) use “hidden columns” to store and maintain the sibling order automatically without any additional user effort.

3 Terminology and requirements

3.1 Acronyms and abbreviations

Table 3 lists acronyms and abbreviations used in this specification.

Acronym / Abbreviation	Description
DOM	Document Object Model
FCS	Flow Cytometry Standard
FLx	The signal from a fluorescence detector x ($x \in \mathbb{N}$)
FSC	Forward Scatter
DSTF	Data Standards Task Force
HTML	Hypertext Markup Language
ISAC	International Society for Advancement of Cytometry
JAXB	Java Architecture for XML Binding
RFC	Request for Comments
SAX	Simple API for XML
SSC	Side Scatter
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
W3C	World Wide Web Consortium
XML	Extensible Markup Language
XSL	Extensible Stylesheet Language
XSLT	XSL Transformations

Table 3: Acronyms and abbreviations used in this specification.

3.2 Mathematical symbols used

Table 4 lists the mathematical symbols used in this specification.

3.3 Terminology

3.3.1 FCS dimension

An FCS dimension within this document is understood as a type of measurement or other characteristic or quality, typically based on (or derived from) a measurement, which is captured (or capturable) in a list-mode data file (*e.g.*, an FCS file) on a per-event basis. These measurements are typically based on a signal (or signals) produced by a detector (or several detectors) of an analytical instrument (typically a flow cytometry analyzer, cell sorter, or mass cytometer). Please note that an FCS dimension is called a “parameter” in the FCS data file standard [2] as

Symbol	Description
\mathbb{N}	The set of natural numbers
\mathbb{R}	The set of real numbers
C^T	Transposition of a matrix C
C^{-1}	C to the power of -1 or the inverse of a matrix C
$[x, y]$	A closed interval from x to y , both x and y are in the interval
(x, y)	On ordered pair or an open interval from x to y , neither x nor y are in the interval
$[x, y)$	A one side open interval from x to y , x is the interval while y is not
$(x, y]$	A one side open interval from x to y , y is the interval while x is not
$\{x, y, \dots, z\}$	A set consisting of items x, y, \dots, z
$X \subset Y$	X is a subset of Y
$X \subseteq Y$	X is either a subset of or it is equal to Y
$X \cup Y$	The union of X and Y
$X \cap Y$	The intersection of X and Y
$x \in Y$	x is in Y , <i>e.g.</i> , $n \in \mathbb{N}$
\forall	The <i>for all</i> symbol specifying that something is true for all items
\exists	The <i>exists</i> symbol specifying that an item exists that something is true
$X \times Y$	The Cartesian product of X and Y
$X \Leftrightarrow Y$	X if and only if Y
$X \Rightarrow Y$	An implication arrow (if X then Y)
$f : X \rightarrow Y$	An indication that f is a Y -valued function of an X -valued variable

Table 4: Mathematical symbols used in this specification.

well as previous versions of Gating-ML [16]. For the purpose of this specification, we decided to abandon the term “parameter” due to the inconsistency of its meaning in the field of flow cytometry vs. common understanding in several other fields, such as mathematics, statistics, probability, logic, computer science, engineering, etc.

The area of the signal detected by the first photomultiplier tube (abbreviated as FL1-A) is an example of an FCS dimension. FCS dimensions may also encompass mathematically transformed measurements (*e.g.*, log fluorescence), or calculated measurements (*e.g.*, electronic opacity), or other normalized measurements (*e.g.*, fluorescence divided by DC impedance or low angle scatter), or other types of measurement (*e.g.*, the relative measurement of the time when an electronic event was detected).

Note that the description of an FCS dimension constructed as a concatenation of the detector type, a sequential number and how the signal was processed (*e.g.*, FL1-W) is appropriate for engineering studies only and it is not suitable for publishing experimental results, labeling graph axes, etc. FSC is a common FCS dimension used in examples within this document. It represents the measurement of light scattered usually less than 10 degrees as a particle passes through the light beam. The FSC measurement is usually related to particle size. SSC represents another FCS dimension commonly used in examples within this document. SSC, also called 90 degrees scatter or right angle scatter (light scattered at the 90 degree angle as a particle passes through the light beam) measurement is usually related to the internal

granularity or complexity of a particle.

Within this document, FCS dimensions correspond to dimensions creating the multidimensional space where gates are formed. Also, FCS dimension may be transformed as described further in this document.

3.3.2 Electronic event

An electronic event within this document is understood as a unit of data that has been generated as a response to a signal occurrence detected by an analytical instrument. An electronic event is a vector (record) that has a value (see section 3.3.4) for each FCS dimension. Electronic events typically describe characteristics of a single particle (*e.g.*, a cell) and they are typically stored in list mode data files (*i.e.*, FCS files).

3.3.3 List mode data file

A list mode data file is a file containing a sequential list of events. List mode data files are typically used for event-by-event data acquisition. Currently, FCS [2] is the most widely (and almost solely) used list mode data file format in flow cytometry, although the FCS specification supports both list mode data and histograms. Gates can only be applied on list mode data (not on histograms).

Although some data files may contain more than one data set, for the purpose of this document, we consider a list mode data file equal to a list mode data set, *i.e.*, each list mode data file containing a single sequential list of events. As of FCS version 3.1, the usage of multiple data sets within a single data file is deprecated unless the multiple data sets are derived one from another.

3.3.4 Event value

An event value is the value recorded for a particular event in a particular FCS dimension. If the used list mode data file format prescribes implicit transformations to decode data, then these shall be applied prior to any transformation that may be described by Gating-ML.

Specifically, values of events stored in FCS data files shall be decoded and converted to the form of so-called “scale values” as specified by the FCS [2] specification. This involves interpretation of several FCS keywords (*e.g.*, \$DATATYPE, \$BYTEORD, \$PnB, \$PnE, \$PnR, \$PnG). For example, let us assume that the value of the \$DATATYPE keyword is “I” (indicating that the integer data type is being used), the value of the \$BYTEORD keyword is “4,3,2,1” (indicating that the big endian byte order is used), the value of the \$MODE keyword is “L” (indicating that list mode data is being stored), the value of the \$PAR keyword is “8” (indicating that eight different FCS dimensions are captured), the value of the \$TOT keyword is “73965” (indicating that there are 73965 electronic events captured in the FCS data set), the value of all 8 \$PnB keywords (*i.e.*, for $n \in \mathbb{N}, n \leq 8$) is “16” (indicating that there are 2 bytes used to store each value), the value of the \$P3N keyword is “FL1-H” (indicating that the short name of the third FCS dimension is FL1-H), the value of the \$P3R keyword is “1024” (indicating that the “range” value for the third FCS dimension is 1024), the value of the \$P3E keyword is “4,1” (indicating that the third FCS dimension is stored on a logarithmic scale) and finally, that the \$P3G keyword is not present. Let us further assume that the data segment of our FCS data file starts with the following byte sequence (shown in hexadecimal notation): 00 C5 00 96

01 AF 00 FF 00 FC 01 14 ... Based on this example, we can decode (calculate) the (scale) value of the FL1-H dimension of the first event as $10^{(4*431/1024)} = 48.26071$ (the 01 AF byte sequence represents the value of 431 in the big endian binary integer encoding, which is further transformed as prescribed by the FCS specification).

3.3.5 Spillover, or spectrum, coefficient and matrix

Spillover or spectrum coefficient from FCS dimension X to FCS dimension Y is the ratio of the amount of signal in the Y channel to the amount of signal in the X channel for particles carrying only the X -dimension dye. Please note that this is not a symmetric relation, *i.e.*, the spillover or spectrum coefficient from X to Y is not the same as the spillover/spectrum coefficient from Y to X .

Historically, spillover coefficients formed square spillover matrices, which have been used to compensate fluorescence expression values, and there was a direct correspondence between fluorescence dyes and their “main” detectors. Compensated values were then obtained by multiplying the vector of uncompensated values by the inverse of the spillover matrix (see Section 7.6.1). Recently, novel instruments have been introduced measuring the fluorescence spectrum with greater detail by many more detectors than dyes in the sample. In these cases, the originally square spillover matrix became a non-square matrix describing the relative amount of fluorescence detected by each of the detectors for each of the dyes in the sample. The process of compensation turned hereby into the problem of spectral unmixing (see Section 7.6.2). In Gating-ML, the term “spectrum” replaced the originally used term “spillover” to be used for both the traditional as well as the recently introduced non-square matrix design.

3.3.6 Dependency

A gate G_a is directly dependent on a gate G_b if and only if G_b is referenced from the definition of G_a . Evaluation of G_b is essential for the evaluation of G_a . A gate G_x is dependent on a gate G_y if and only if there is a sequence of gates $G_x, G_{x+1}, G_{x+2}, \dots, G_{y-1}, G_y$, where G_n is directly dependent on G_{n+1} for each n from $\{x, x+1, \dots, y-1\}$. The following types of dependencies may occur:

- A gate is dependent on its ancestor gates within the gates hierarchy (see section 4.4). Referencing a parent gate as part of the gate definition produces the same data filtering results as creating an “AND” Boolean gate (see section 5.5).
- A Boolean gate is dependent on its operand gates (see section 5.5).

3.3.7 Dependency graph

Dependency graph of a Gating-ML XML file is a directed graph $D = (G, A)$, where the set of vertices G is the set of all gates of the Gating-ML XML file, and the set of arcs (directed edges) A contains an arc (G_a, G_b) if and only if the gate G_a is directly dependent on the gate G_b .

3.3.8 Data transformation

Within this document, data transformation is a procedure that decodes and converts event values from a list mode data file to values that gates are applicable to. The channel-to-scale

transformation (see section 3.3.4) is the first one that is implicitly applied after decoding the event vector from an FCS file. Gates may explicitly specify further data transformations to be performed prior the application of the gate. These may include compensation (see section 7), a scale transformation (see sections 3.3.9 and 6) or an additional transformation, such as the ratio of two FCS dimensions (see section 8). If multiple transformations are specified, then the compensation shall be applied first (if applicable), the ratio second (if applicable), and the scale transformation last (if applicable). Figure 5 demonstrates this work flow in more detail.

3.3.9 Scale transformation

Within this document, a scale transformation is a data transformation that is typically applied in order to bring the data to a scale that is more suitable for the visualization or analysis. Details about scale transformations supported by this specification are listed in section 6.

For the purpose of this specification, compensation (see section 7) and additional transformations (*i.e.*, the ratio of two FCS dimensions, see section 8) are not considered scale transformations. Each gate dimension definition may reference up to one compensation description and up to one scale transformation. If referenced, these transformations are supposed to be applied prior to applying the gate as described.

Scale transformations are functions $f : X \rightarrow Y$, where $X \subseteq \mathbb{R}, Y \subseteq \mathbb{R}$. In addition, these scale transformations are typically parametrized so that “reasonable” values of $x \in X$ are mapped to the $[0, 1]$ interval, *i.e.*, $f(x) \in [0, 1]$. The semantic of the term “reasonable” is dependent on the actual scale transformation. Typically, it means that $x \leq T$ where T is the specified top of the scale value. In addition, $x > 0$ for some transformation (*e.g.*, parametrized logarithmic) or not “too negative” for transformations that allow bringing negative data to scale by adjusting parameters such as W and A (*e.g.*, Logicle, Hyperlog).

3.4 Conformance

3.4.1 File conformance

To be conformant with this standard, a Gating-ML XML file shall pass validation according to the Gating-ML XML schema, all components (*i.e.*, gates, transformations, and spectrum matrices) shall meet additional requirements (validity conditions) if these are stated for the corresponding component type, and there shall be no circular dependency within the Gating-ML XML file, *i.e.*, the dependency graph shall be a directed acyclic graph (see also sections 3.3.6 and 3.3.7).

3.4.2 Applicability

A compliant Gating-ML XML file is applicable to a certain list mode data file if and only if all list mode data file dimensions (FCS dimensions) referenced from the Gating-ML file (see section 4.1) are present in the list mode data file.

3.4.3 Software and hardware conformance

To be compliant with this standard, a software application or hardware instrument shall be able to *read*, *write*, or *read and write* Gating-ML XML files and it shall meet the following

criteria:

- When Gating-ML XML files are produced (written), then these shall be valid Gating-ML files according to section 3.4.1.
- When Gating-ML XML files are read, then the software application or the hardware instrument shall be able to read any Gating-ML XML file that is valid according to section 3.4.1 and it should be able to process all components (*e.g.*, gates and transformations) of the Gating-ML file.

Gating-ML software compliance tests and a reference implementation are provided along with this specification.

4 General

4.1 Referencing dimensions

FCS dimensions from list mode data files are referenced by their names using the *fcs-dimension* element with the *name* attribute as shown in Example 1. An *fcs-dimension* element is typically included as a sub element of the *dimension* element of a gate definition or as a sub element of the *divider* element of a quadrant gate definition (see section 5.4). Two *fcs-dimension* elements are typically included as sub elements of the *fratio* transformation definition (see section 8.1). Multiple *fcs-dimension* elements are typically included as sub elements of the *fluorochromes* and *detectors* elements of a spectrum matrix definition (see section 7). Compensation (spectral unmixing) defined by a spectrum matrix within Gating-ML is a function defined on dimensions specified by detectors (the columns of the matrix) and resulting in dimensions specified by the fluorochromes (rows of the matrix). The sets of detector and fluorochrome names shall be different from each other, *i.e.*, no name from the list of detector names shall be included in the list of fluorochrome names and vice versa. Gates shall use the set of fluorochrome names to reference dimensions after a compensation (spectral unmixing) as per a spectrum matrix defined in the Gating-ML file. Please note that this is different from the case where compensation is specified within the data file directly, which does not create a new set of dimension names (see section 4.2.2).

```
<gating:RectangleGate gating:id="FSC_Range_Gate">
  <gating:dimension
    gating:min="10" gating:max="20" gating:compensation-ref="uncompensated">
    <data-type:fcs-dimension data-type:name="FSC-H" />
  </gating:dimension>
</gating:RectangleGate>
```

Example 1: Referencing FCS dimensions in gate definitions.

FCS dimensions created by additional transformations (*i.e.*, the ratio of two FCS dimensions) are referenced by placing the identifier of the transformation in the value of the *transformation-ref* attribute of the *new-dimension* element of a gating dimension definition. These FCS dimensions can only be used as gating dimensions with optional scale transformations applied on

top of these; however, they cannot be used within the ratio transformation, neither as detectors or fluorochromes of a spectrum matrix definition. Example 2 demonstrates how a ratio of two FCS dimensions can be reused in a range gate definition. Currently, a parametrized ratio of two FCS dimensions is the only supported “additional” transformation.

```
<gating:RectangleGate gating:id="Range_Gate_On_Ratio">
  <gating:dimension gating:min="5" gating:max="20" gating:compensation-ref="FCS">
    <data-type:new-dimension data-type:transformation-ref="myRatio" />
  </gating:dimension>
</gating:RectangleGate>
<transforms:transformation transforms:id="myRatio">
  <transforms:fratio transforms:A="1" transforms:B="0" transforms:C="0" >
    <data-type:fcs-dimension data-type:name="FL1-A" />
    <data-type:fcs-dimension data-type:name="FL2-A" />
  </transforms:fratio>
</transforms:transformation>
```

Example 2: Referencing the ratio of two FCS dimensions in gate definitions.

The dimension names (and transformation identifiers) are case sensitive (*i.e.*, “FSC-H” and “Fsc-H” are two different FCS dimension names). In case the corresponding list mode data file is an FCS [2] file, the name is the value of the \$PnN keywords. In case of other list mode data files, the name of the FCS dimension is format-specific identification of the dimension. Specifically, in case of list-mode data files stored in NetCDF [17] files, the name of a dimension is the “variable name” in the NetCDF file. In case of tabular-based list mode data files with rows corresponding to events and columns corresponding to FCS dimensions (*e.g.*, CSV files [18], MS Excel spreadsheets, etc.), the name of a dimension is the heading of the particular data column.

4.2 FCS dimensions and transformations

Gates are applicable on list mode data files either directly or after a transformation. These options are briefly introduced below and demonstrated in the work flow diagram shown in Figure 5.

4.2.1 Gates applicable directly on uncompensated list mode data files

A direct application means that the dimensions of the gate correspond to FCS dimensions of the list mode data file without additional transformations except for transformations implicitly defined by the list mode data file format to decode values in the file. Specifically for FCS, the channel-to-scale transformation shall always be applied (see section 3.3.4), but no other transformations (including fluorescence compensation) shall be applied unless explicitly specified by Gating-ML as shown in Sections 4.2.2, 4.2.3, 4.2.4, and 4.2.5. In this case, the gate definition in Gating-ML does not reference any additional transformation and it specifies that the data are uncompensated. Example 3 demonstrates a gate applicable on scale values of a list mode data file directly.

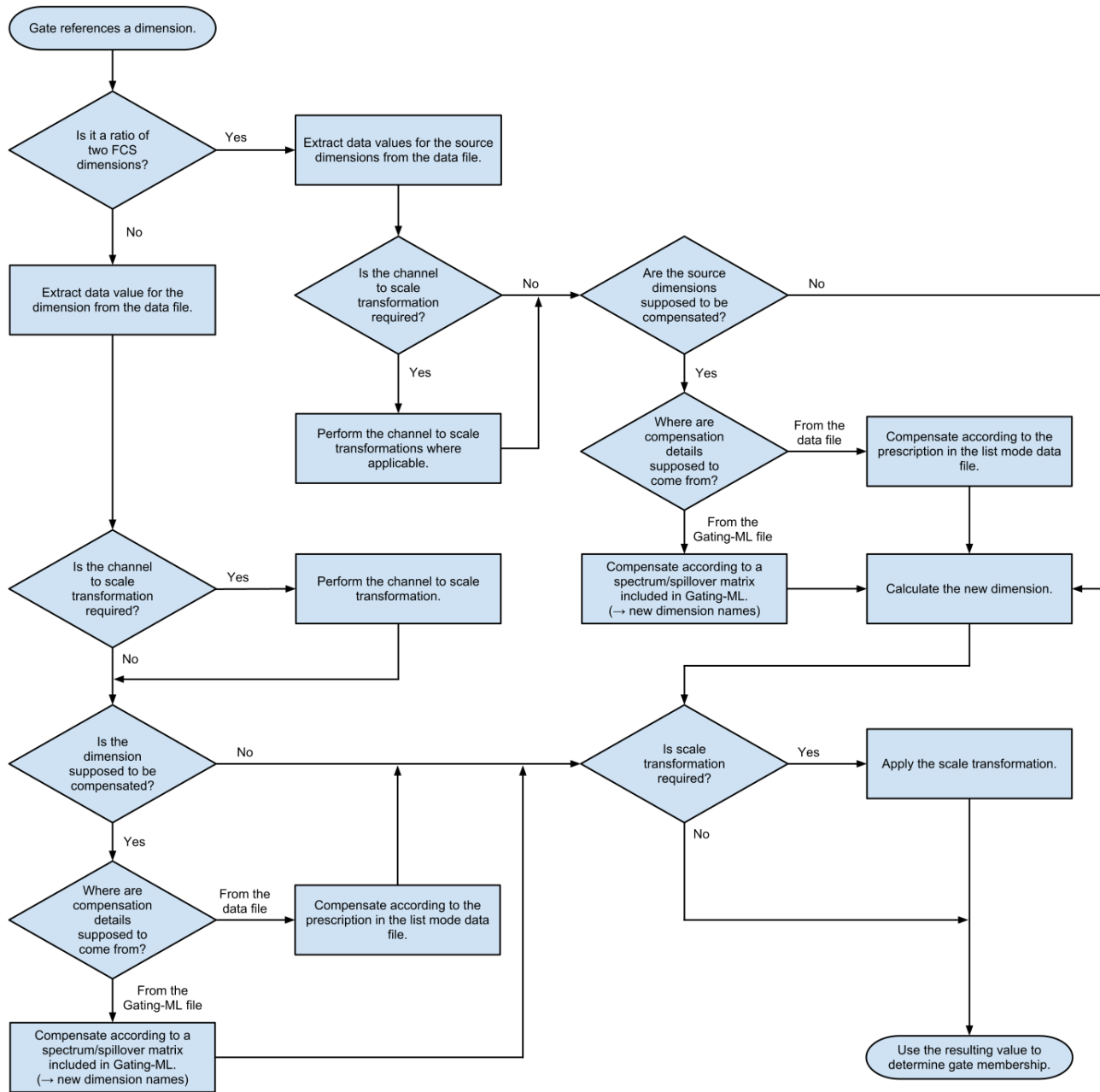


Figure 5: Work flow of data transformations prior to the application of a gate.

4.2.2 Gates applicable on compensated data

If a gate is supposed to be applied on compensated data, then the compensation shall be referenced as part of the gate definition. There are two options where the compensation definition may reside:

1. The compensation definition may be encoded directly in the same list mode data file as the data (event values). Typically, this may be by means of the `$SPILLOVER` keywords in FCS version 3.1 [2], `$COMP` keywords in FCS version 3.0 [7] (rarely used) or some custom keywords, such as the `SPILL` keywords as introduced by Becton, Dickinson and Company.

```

<gating:PolygonGate gating:id="Example_P1">
  <gating:dimension gating:compensation-ref="uncompensated">
    <data-type:fcs-dimension data-type:name="FSC-H" />
  </gating:dimension>
  <gating:dimension gating:compensation-ref="uncompensated">
    <data-type:fcs-dimension data-type:name="SSC-H" />
  </gating:dimension>
  <gating:vertex>
    <gating:coordinate data-type:value="100" />
    <gating:coordinate data-type:value="100" />
  </gating:vertex>
  <!-- Definition of other vertices would follow -->
  ...
</gating:PolygonGate>

```

Example 3: Gate applicable directly on uncompensated scale values of a list mode data file.

The Gating-ML definition of a dimension that is supposed to reference data compensated as prescribed by related list mode data file shall contain a *compensation-ref* attribute set to the value “FCS” as shown in Example 4. Names of the FCS dimensions do not change with compensation according to the list mode data file, *i.e.*, in case the corresponding list mode data file is an FCS file then gates should still use the values of the \$P_nN keywords to reference gating dimensions.

2. The compensation definition may be part of the Gating-ML file. In this case, the compensation is defined by a spectrum matrix as specified in section 7. The Gating-ML definition of a dimension that is supposed to reference data compensated according to the spectrum matrix defined in Gating-ML shall contain a *compensation-ref* attribute set to the value of the identifier of the spectrum matrix as shown in Example 5. Compensation (spectral unmixing) defined by a spectrum matrix within Gating-ML is a function defined on dimensions specified by detectors (columns) of the matrix and resulting in dimensions specified by the fluorochromes (rows) of the matrix. The sets of detector and fluorochrome names shall be different from each other, *i.e.*, no name from the list of detector names shall be included in the list of fluorochrome names and vice versa. For these cases, gates shall use the set of fluorochrome (row) names to reference compensated (un-mixed) dimensions as per such a spectrum matrix (Example 6).

4.2.3 Gates applicable on data on transformed scales

As mentioned in section 2.3.1, further scale transformations are commonly performed to provide more complete, appropriate, and readily interpretable data representations, or to achieve optimal display of the data. If a gate uses a transformed scale (*e.g.*, the gate is drawn while data is displayed on a transformed scale), then the transformation shall be recorded in the Gating-ML (see section 6) and referenced in the appropriate gating dimension definition using the *transformation-ref* attribute as shown in Example 7. These transformations may be combined with both compensated and uncompensated data using the *compensation-ref* attribute

```

<gating:PolygonGate gating:id="Example_P2">
  <gating:dimension gating:compensation-ref="FCS">
    <data-type:fcs-dimension data-type:name="FL1-H" />
  </gating:dimension>
  <gating:dimension gating:compensation-ref="FCS">
    <data-type:fcs-dimension data-type:name="FL2-H" />
  </gating:dimension>
  <!-- Definition of gate vertices would follow -->
  ...
</gating:PolygonGate>

```

Example 4: Gate applicable on scale values compensated as defined in the FCS file.

```

<transforms:spectrumMatrix transforms:id="my_spill_1">
  <transforms:fluorochromes>
    <data-type:fcs-dimension data-type:name="FITC"/>
    <data-type:fcs-dimension data-type:name="PE"/>
  </transforms:fluorochromes>
  <transforms:detectors>
    <data-type:fcs-dimension data-type:name="FL1-H"/>
    <data-type:fcs-dimension data-type:name="FL2-H"/>
  </transforms:detectors>
  <!-- Definition of the matrix would continue here -->
  ...
</transforms:spectrumMatrix>

<gating:PolygonGate gating:id="Example_P3">
  <gating:dimension gating:compensation-ref="my_spill_1">
    <data-type:fcs-dimension data-type:name="FITC" />
  </gating:dimension>
  <gating:dimension gating:compensation-ref="my_spill_1">
    <data-type:fcs-dimension data-type:name="PE" />
  </gating:dimension>
  <!-- Definition of gate vertices would follow -->
  ...
</gating:PolygonGate>

```

Example 5: Gate applicable on scale values compensated as defined in Gating-ML.

as described in sections 4.2.1 and 4.2.2, and they may also be applied on dimensions created by “additional” transformations, *i.e.*, on the ratio of two FCS dimensions.

4.2.4 Gates applicable on the ratio of two FCS dimensions

Gates may be applied on new FCS dimensions created as the ratio of two FCS dimensions as previously described in section 4.1. These newly created FCS dimensions shall be referenced by placing the identifier of the source ratio transformation in the value of the *transformation-ref* attribute of the *new-dimension* element of a gating dimension definition (see also Example 2).

```

<transforms:spectrumMatrix transforms:id="my_spectrum_1">
  <transforms:fluorochromes>
    <data-type:fcs-dimension data-type:name="FITC"/>
    <data-type:fcs-dimension data-type:name="PE"/>
  </transforms:fluorochromes>
  <transforms:detectors>
    <data-type:fcs-dimension data-type:name="FL1-H"/>
    <data-type:fcs-dimension data-type:name="FL2-H"/>
    <data-type:fcs-dimension data-type:name="FL3-H"/>
  </transforms:detectors>
  <!-- Definition of the matrix would continue here -->
  ...
</transforms:spectrumMatrix>

<gating:PolygonGate gating:id="Example_P4">
  <gating:dimension gating:compensation-ref="my_spectrum_1">
    <data-type:fcs-dimension data-type:name="FITC" />
  </gating:dimension>
  <gating:dimension gating:compensation-ref="my_spectrum_1">
    <data-type:fcs-dimension data-type:name="PE" />
  </gating:dimension>
  <!-- Definition of gate vertices would follow -->
  ...
</gating:PolygonGate>

```

Example 6: Gate applicable on scale values un-mixed as defined in Gating-ML.

```

<transforms:transformation transforms:id="my_logicle_1">
  <transforms:logicle transforms:T="10000" transforms:M="4.5"
    transforms:W="0.5" transforms:A="0" />
</transforms:transformation>

<gating:PolygonGate gating:id="Example_P5">
  <gating:dimension gating:transformation-ref="my_logicle_1"
    gating:compensation-ref="FCS">
    <data-type:fcs-dimension data-type:name="FL4-H" />
  </gating:dimension>
  <gating:dimension gating:transformation-ref="my_logicle_1"
    gating:compensation-ref="FCS">
    <data-type:fcs-dimension data-type:name="FL5-H" />
  </gating:dimension>
  <!-- Definition of gate vertices would follow -->
  ...
</gating:PolygonGate>

```

Example 7: Gate applicable on data on a Logicle scale.

Similarly to referencing regular FCS dimensions, the ratio may be referenced as compen-

sated or uncompensated by specifying the *compensation-ref* attribute as described in sections 4.2.1 and 4.2.2. However, if compensation is specified, then it shall be applied on the source FCS dimensions (*i.e.*, dimensions from the list mode data file, such as FL1, FL2, etc.) prior to the calculation of the ratio as shown in Figure 5. In Example 2 above, one would first apply the implicit channel-to-scale transformation on the FL1-A and FL2-A FCS dimensions (if applicable based on how data are captured in the list mode data file, see section 3.3.4), then compensate these dimensions as prescribed by the list mode data file (*e.g.*, the `$SPILLOVER` or `$COMP` keywords, see section 4.2.2), then calculate the ratio of compensated FL1-A and FL2-H, and then apply the gate.

4.2.5 Gates applicable on scaled ratio of two FCS dimensions

As described in section 4.2.4, gates may be applied on the ratio of two FCS dimensions. This may be referenced as compensated or uncompensated. In addition, the ratio of two FCS dimensions may be scaled using one of the scale transformations listed in section 6.

In the situation described in Example 8, one would first apply the implicit channel-to-scale transformation on the FL3-A and FL4-A FCS dimensions (if applicable based on how data are captured in the list mode data file, see section 3.3.4), then compensate these dimensions as prescribed by the list mode data file (*e.g.*, the `$SPILLOVER` or `$COMP` keywords, see section 4.2.2), then calculate the ratio of compensated FL3-A and FL4-A, then scale the result using the prescribed logarithmic transformation, and then finally apply the gate.

```
<gating:RectangleGate gating:id="Range_Gate_On_Scaled_Ratio">
  <gating:dimension gating:min="0.5"
    gating:compensation-ref="FCS" gating:transformation-ref="myRatioLog">
    <data-type:new-dimension data-type:transformation-ref="myRatio" />
  </gating:dimension>
</gating:RectangleGate>
<transforms:transformation transforms:id="myRatio">
  <transforms:fratio transforms:A="1" transforms:B="0" transforms:C="0" >
    <data-type:fcs-dimension data-type:name="FL3-A" />
    <data-type:fcs-dimension data-type:name="FL4-A" />
  </transforms:fratio>
</transforms:transformation>
<transforms:transformation transforms:id="myRatioLog">
  <transforms:flog transforms:T="100" transforms:M="4" />
</transforms:transformation>
```

Example 8: Referencing scaled ratio of two FCS dimensions in a gate definition.

4.3 Results of gating operations

The gating operation returns the set of events within the gate. The actual values of these events shall not change as a side effect of any gating operation. If a gate definition includes compensation, then the compensation should be used if the gating results are to be visualized graphically. Similarly, if a gate definition includes a scale transformation, then scale transformation should

be used if the gating results are to be visualized graphically. Internally, the analytical software may also want to keep compensated and scaled values for performance purposes.

4.4 Hierarchical gating and gating strategy description

A gate definition can include an optional *parent_id* attribute that can be used to reference a parent gate by its identifier. A gate is supposed to be applied on the whole population unless the *parent_id* attribute is used. If the *parent_id* attribute is specified, then the gate is supposed to be applied on a population defined by its parent gate, which may be used to encode the gating strategy (hierarchy). Let G be a gate with a specified parent gate G_p , and let G' be a gate defined the same way as gate G except for the *parent_id* attribute – gate G' does not define any parent gate. The result of applying a gate G is the intersection of the result of applying the parent gate G_p with the gate G' . Same results (in terms of events filtered by the gate) may be obtained by creating a Boolean *And* gate. However, a Boolean *And* gate does not capture the semantic of a parent gate or the gating hierarchy.

4.5 Gating-ML file structure

A Gating-ML file is a valid XML [3] file that corresponds to the Gating-ML.v2.0.xsd XML schema and conforms to the validity conditions as described in Section 3.4.1. The Gating-ML XML schema specifies the `http://www.isac-net.org/std/Gating-ML/v2.0/gating` target namespace. Gating-ML is the main element of a Gating-ML file. Child elements describe individual gates, transformations, and spectrum matrices. Example 9 shows an example of the header of a Gating-ML XML file.

Please note that you may also use a local path (formatted as URI/URL) to reference XML schema files in the `schemaLocation` attribute in order to accelerate the XML validation process. Definition of gates, transformations, spectrum matrices, and custom elements is included within the Gating-ML element as described further in this specification. The main *Gating-ML* element may have one or more of the following sub elements in any order:

RectangleGate – description of an n dimensional rectangular gate (including a range gate and hyper-rectangular regions) as described in section 5.1.

PolygonGate – description of a two dimensional polygon gate as described in section 5.2.

EllipsoidGate – description of an n dimensional ellipsoid gate (including 2 dimensional ellipse gate) as described in section 5.3.

QuadrantGate – description of a quadrant gate as described in section 5.4.

BooleanGate – description of a Boolean gate as described in section 5.5.

transformation – description of a scale transformation as described in section 6 or a new parameter creation as described in section 8.

spectrumMatrix – description of a spectrum (spillover) matrix as described in section 7.

custom_info – additional custom information as described in section 4.8.

```

<?xml version="1.0" encoding="UTF-8"?>
<gating:Gating-ML xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:gating="http://www.isac-net.org/std/Gating-ML/v2.0/gating"
  xmlns:transforms="http://www.isac-net.org/std/Gating-ML/v2.0/transformations"
  xmlns:data-type="http://www.isac-net.org/std/Gating-ML/v2.0/datatypes"
  xsi:schemaLocation="
    http://www.isac-net.org/std/Gating-ML/v2.0/gating
    http://flowcyt.sourceforge.net/gating/2.0/xsd/Gating-ML.v2.0.xsd
    http://www.isac-net.org/std/Gating-ML/v2.0/transformations
    http://flowcyt.sourceforge.net/gating/2.0/xsd/Transformations.v2.0.xsd
    http://www.isac-net.org/std/Gating-ML/v2.0/datatypes
    http://flowcyt.sourceforge.net/gating/2.0/xsd/DataTypes.v2.0.xsd">

  <!-- Description of gates, transformations, spectrum matrices, etc. -->
  <gating:RectangleGate gating:id="Example_R1">
    <gating:dimension gating:min="100" gating:compensation-ref="uncompensated">
      <data-type:fcs-dimension data-type:name="FSC-H"/>
    </gating:dimension>
  </gating:RectangleGate>

</gating:Gating-ML>

```

Example 9: Gating-ML XML file demonstrating proper header definition.

4.6 Gates overview

Gating-ML element is the main element of a Gating-ML file. The following sub elements may be included to define various types of gates: *RectangleGate*, *PolygonGate*, *EllipsoidGate*, *QuadrantGate* and *BooleanGate*.

The XML schemas define an abstract gate type *AbstractGate_Type* that is used as an extension base for all the other types of gates. This design allows all gate types to inherit the *id* attribute that is used for identification and reference purposes, and the *parent_id* attribute that allows for description of hierarchical gating and gating strategies as described in section 4.4.

4.7 Scale transformations overview

Gating-ML allows for a scale transformation description within a *transformation* element (sub element of the main *Gating-ML* element). The *transformation* element shall include an *id* attribute to assign a unique identifier to the transformation. In addition, the *boundMin* and *boundMax* attributes may be present. If present, these attributes define the boundaries of the transformation as specified in section 6.1. The transformation description follows within the transformation element as one of the *flin* (parametrized linear), *flog* (parametrized logarithmic), *fasinh* (parametrized inverse hyperbolic sine), *logicle* and *hyperlog*.

Design goals of the specification of the logarithmic and log-like transformations are to make these comprehensible to the end users, *i.e.*, scientists, and also to ease the burden to implementers. To achieve the latter, these transformations are designed so that they map an appropriate range of data values onto a standard unit display interval [0,1].

To make the transformations comprehensible, they are defined in terms of a consistent parameterization that is closely related to the user experience. The top of scale value T is always mapped to the value 1. In addition, the logarithmic and log-like transforms are parametrized by M , the number of decades of data range mapped onto the unit display interval by the Logarithmic transform. The log-like scales are also parametrized by W and A , which are commensurate with decades on the scale although they do not represent ten fold changes in data value. W controls the degree of linearization for the Logicle and Hyperlog transforms. The parameter A specifies an additional range of negative data values that are to be brought on scale. For the Logicle and Hyperlog transforms, this is in addition to what is already brought on scale by W and should generally not be needed. The Logicle, Hyperlog, and parametrized inverse hyperbolic sine transforms with $A = 0$ will all behave like the logarithmic transform with the same values of T and M for large data values.

This choice of parameters also leads to a sensible fall back strategy when software does not implement a particular transform. For example, if the Logicle transform is not available, then a Hyperlog transform with the same parameters should be a reasonable alternative, and vice versa (see tables 8 and 9). Figure 6 shows a comparison of the parametrized logarithmic and the supported log-like scale transformations with $T = 1000$, $M = 4.5$, $W = 1$ and $A = 0$ (W and A set where applicable only). As you can see, all transformations are very close to each other for large data values. In the low data range (zoomed-in in the two sub figures), the parametrized inverse hyperbolic sine, Logicle, and Hyperlog transforms show a “linear-like” behavior around zero and “extend” the scale to the negative data range. Unlike the parametrized inverse hyperbolic sine, the Logicle and Hyperlog transforms allow the width of the linearization region to be controlled independently from the logarithmic character for large data values.

The effect of increasing A is shown in Figure 7, which demonstrates the same transformations with $T = 1000$, $W = 1$, and $M = 5$ for the parametrized logarithmic, while $M = 4$ for the other transformations since these also include an additional negative decade ($A = 1$).

4.8 Additional custom information

Gating-ML allows for a *custom_info* element to capture additional vendor specific information within Gating-ML. This element may be placed at the top level as a sub element of the main *Gating-ML* element as demonstrated in Example 10, or as the first sub element of the *spectrumMatrix* element (for information related to the compensation definition), *transformation* element (for information related to a scale transformation), or any of the gate definition elements (for information related to a particular gate definition, see section 5).

In addition, any gate, transformation or spectrum (spillover) matrix element may include custom attributes as permitted by the XML schemas. This is also demonstrated by examples accompanying this specification. Custom information may be used to encode additional optional information (*e.g.*, visual properties of a drawn gate), but shall not be used as proprietary alternative to communicate aspects standardized by this specification.

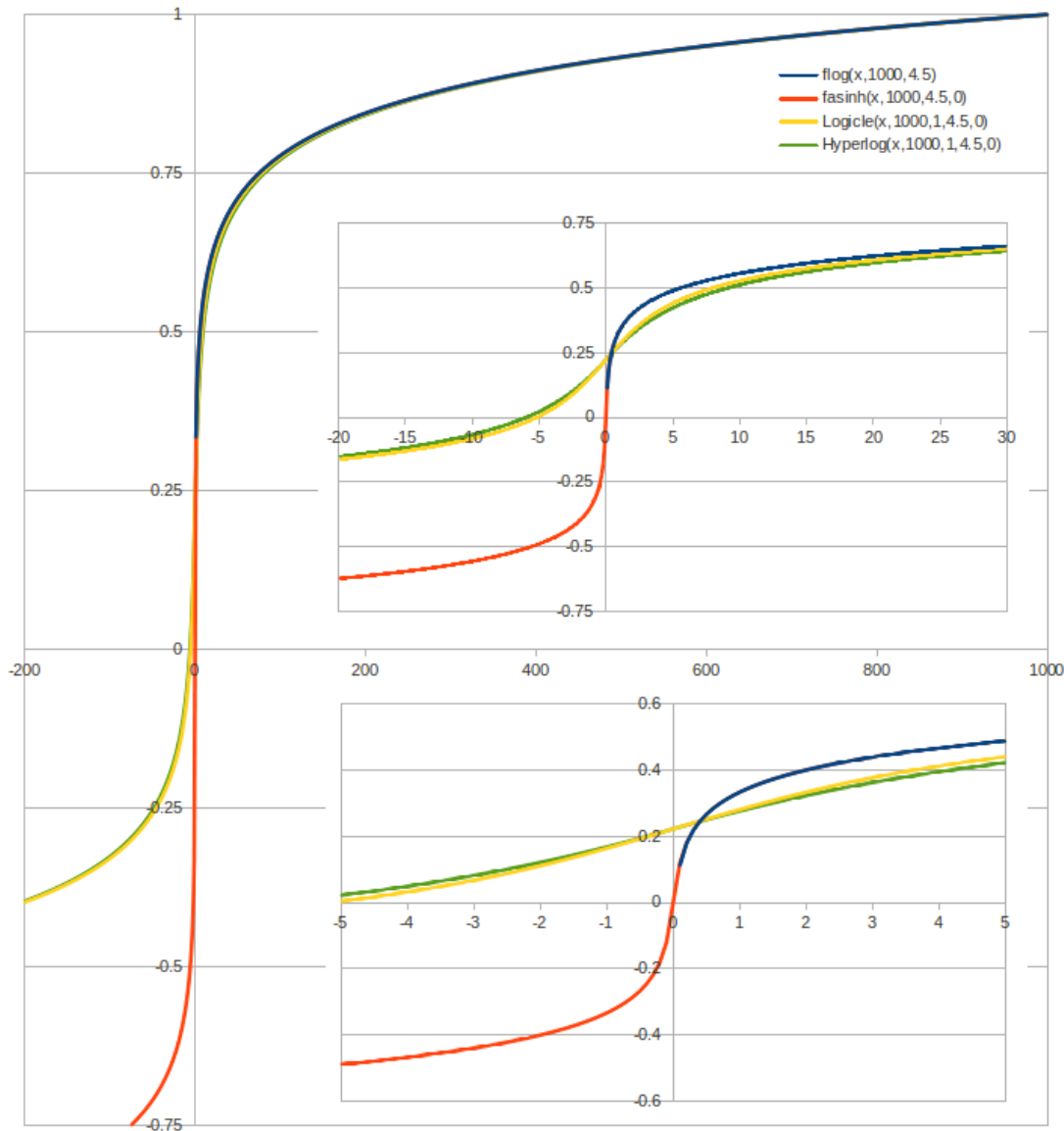


Figure 6: Comparison of transformations with $T = 1000$, $M = 4.5$, $W = 1$, and $A = 0$.

5 Gate description

5.1 Rectangular and range gates

5.1.1 Definition

A rectangular gate in n dimensions is a Cartesian product of n orthogonal intervals in these dimensions. The dimensions represent FCS dimensions that may be compensated and/or transformed with a scale transformation as defined in section 6. Rectangular gates are used to express range gates ($n = 1$, *i.e.*, one dimension), rectangle gates ($n = 2$, *i.e.*, two dimensions), box regions ($n = 3$, *i.e.*, three dimensions), and hyper-rectangular regions ($n > 3$, *i.e.*, more than three dimensions).

An event e is considered to be in the rectangular gate $G(d_{1_{min}}, d_{1_{max}}, \dots, d_{n_{min}}, d_{n_{max}})$ if and

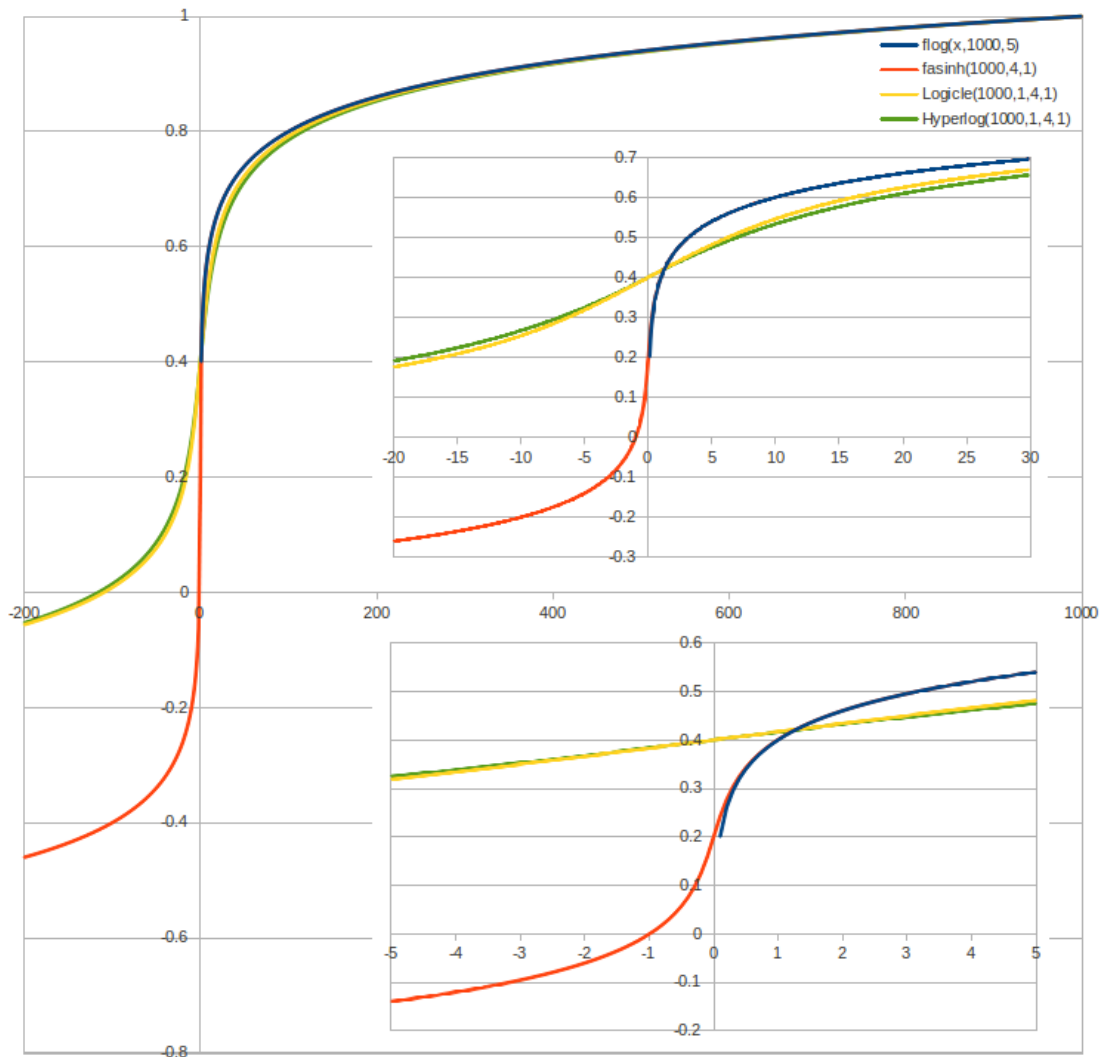


Figure 7: Comparison of transformations with $T = 1000$, $M = 4$ or 5 , $W = 1$ and $A = 1$.

```
<?xml version="1.0" encoding="UTF-8"?>
<gating:Gating-ML ...">
  ...
  <data-type:custom_info custom_attributes="also allowed">
    Custom information is allowed at the top level of Gating-ML
    description within the custom_info element.
    <subelements>Also are allowed.</subelements>
  </data-type:custom_info>
  ...
  <!-- Description of gates, transformations, spectrum matrices etc. follows -->
  ...
</gating:Gating-ML>
```

Example 10: Custom_info element at the top level of a Gating-ML file.

only if for each dimension d_i , $i \in \mathbb{N}$, $1 \leq i \leq n$ of the Gate G $d_{i_{min}} \leq e(d_i) < d_{i_{max}}$, where $d_{i_{min}}$ is the value of the *min* attribute of the dimension d_i , $d_{i_{max}}$ is the value of the *max* attribute of the dimension d_i , and $e(d_i)$ is the scale value of the event e in FCS dimension d_i (which may be compensated or additionally transformed). The intervals are treated as half open with weak inequality of the lower bound and strict inequality of the upper bound so that a set of rectangle gates that covers the data space will sum properly with no missing or duplicated events.

5.1.2 Syntax specification

Rectangular gates shall be specified by the *RectangleGate* element as follows:

- (a) The *id* attribute shall be used to identify the gate for further referencing purposes, *e.g.*, as described in sections 4.4 and 5.5.
- (b) The *parent_id* attribute may be used to reference a parent gate as described in section 4.4.
- (c) A *custom_info* element may be placed at the beginning of the *RectangleGate* element. The *custom_info* element may contain any additional custom information related to this gate; the format of this element (*i.e.*, its attributes, sub elements, etc.) is not constrained by this specification as long as the document remains a valid XML.
- (d) At least one dimension shall be described by a *dimension* element placed either as the first sub element of the *RectangleGate* element or right after the *custom_info* element if the *custom_info* element is used. The maximum number of *dimension* elements is unbounded.
- (e) Each *dimension* element shall specify compensation by including the *compensation-ref* attribute as specified in section 4.2. The value of this attribute may be one of the following: *FCS*, *uncompensated*, or the identifier of a spectrum matrix defined within the Gating-ML file.
- (f) Each *dimension* element may use the *transformation-ref* attribute to reference a scale transformation. If used, the value of this attribute shall contain the identifier of a scale transformation defined within this Gating-ML file (see section 4.2.3).
- (g) Each *dimension* element shall specify a range by the *min* or *max* attributes of the dimension element; at least one of the *min* or *max* attributes shall be present. If they are both specified, the value of the *min* attribute shall be less than the value of the *max* attribute. If the *max* attribute is absent, then the range in this dimension is considered as $[min, \infty)$. If the *min* attribute is absent, then the range in this dimension is considered as $[-\infty, max)$. If both the *min* and *max* attributes are present, then the range in this dimension is considered as $[min, max)$.
- (h) Each *dimension* element shall reference a dimension as described in section 4.1, *i.e.*, by including an *fcs-dimension* element or a *new-dimension* element. The same FCS dimension or new dimension may not be used more than once within the gate definition.

5.1.3 Validity conditions

A rectangular gate definition is valid if and only if:

- (a) It is valid according to the XML schema.
- (b) If used, the *parent_id* attribute does not introduce any circular dependency (see also sections 3.3.6, 3.3.7 and 3.4.1).
- (c) The gating dimensions are different from one another, *i.e.*, the values of the *name* attributes are unique among all *fcs-dimension* elements (if used) in the gate definition, and the values of the *transformation-ref* attributes are unique among all *new-dimension* elements (if used) in the gate definition.
- (d) The value of the *compensation-ref* attribute of each *dimension* element equals to either *FCS*, *uncompensated*, or an identifier of a spectrum matrix defined within the Gating-ML file.
- (e) If used, the value of the *transformation-ref* attribute of the *dimension* element shall contain an identifier of a scale transformation defined within the same Gating-ML file.
- (f) At least one of the *min* and *max* attributes is specified for each *dimension* element.
- (g) If both the *min* and the *max* attributes are specified, then the value of the *min* attribute shall be less than the value of the *max* attribute.
- (h) If used, the value of the *transformation-ref* attribute of the *new-dimension* element shall contain an identifier of a transformation that creates a new dimension, *i.e.*, an identifier of a parametrized ratio transformation.

5.1.4 Examples

- (a) Example 11 demonstrates the definition of a range gate that includes events with scale values of the FCS-H dimension from the [100, 250) interval. Visual representation of this gate is shown in Figure 8. Uncompensated values are used for the purpose of this gate.

```
<gating:RectangleGate gating:id="Range_FCS_b_100_250">
  <gating:dimension gating:min="100" gating:max="250"
    gating:compensation-ref="uncompensated">
    <data-type:fcs-dimension data-type:name="FSC-H" />
  </gating:dimension>
</gating:RectangleGate>
```

Example 11: Range gate definition.

- (b) Example 12 demonstrates the definition of an open range gate that includes events with scale values of the FCS-H dimension greater than or equal to 100. Visual representation of this gate is shown in Figure 9. Values compensated as prescribed by the list mode data file (see section 4.2.2 for more details) are used for the purpose of this gate. In this particular

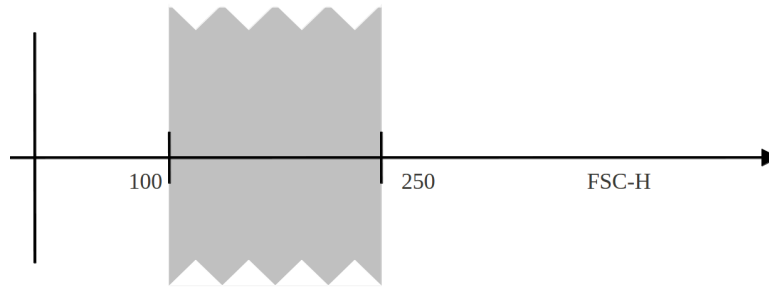


Figure 8: Example of a range gate.

example, the value of FSC-H won't likely be affected by compensation; however, it is still legal to specify that `compensation-ref="FCS"` even for FCS dimensions not included in the spillover matrix of the FCS file. If `compensation-ref="FCS"` but no valid compensation description is present in the FCS file for that particular dimension, then the value shall be used uncompensated.

```
<gating:RectangleGate gating:id="Range_FCS_o100">
  <gating:dimension gating:min="100"
    gating:compensation-ref="FCS">
    <data-type:fcs-dimension data-type:name="FSC-H" />
  </gating:dimension>
</gating:RectangleGate>
```

Example 12: Open range gate definition.

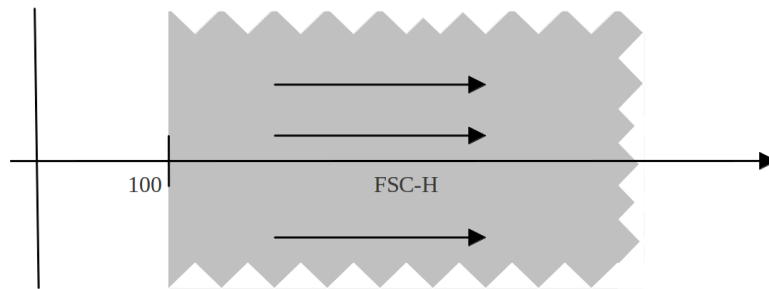


Figure 9: Example of a range gate.

- (c) Example 13 demonstrates a two-dimensional rectangular gate specified on a logicle scale (see section 6.5, same transformation reused for both dimensions) with compensation applied as prescribed by the list mode data file. The gate is applied on a parent gate (see section 4.4). Visual representation of this gate is shown in Figure 10.
- (d) Example 14 demonstrates a three-dimensional hyper-rectangular gate with one unbounded dimension. It also demonstrates the use of the `custom_info` element. Visual representation of this gate is shown in Figure 11.

```

<transforms:transformation transforms:id="myLogicle">
  <transforms:logicle transforms:T="10000" transforms:M="4.5"
    transforms:W="0.5" transforms:A="0" />
</transforms:transformation>
<gating:RectangleGate gating:id="myRect" gating:parent_id="Range_FCS_o100">
  <gating:dimension gating:min="0.2" gating:max="0.5"
    gating:compensation-ref="FCS" gating:transformation-ref="myLogicle">
    <data-type:fcs-dimension data-type:name="FL1-H" />
  </gating:dimension>
  <gating:dimension gating:min="0.25" gating:max="0.4"
    gating:compensation-ref="FCS" gating:transformation-ref="myLogicle">
    <data-type:fcs-dimension data-type:name="FL2-H" />
  </gating:dimension>
</gating:RectangleGate>

```

Example 13: Rectangular gate definition.

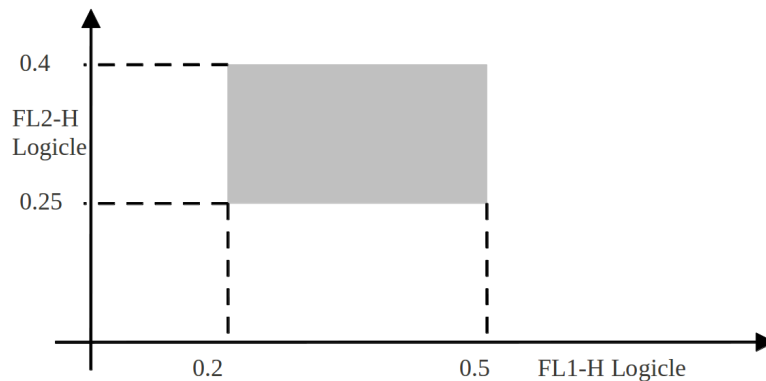


Figure 10: Example of a rectangular gate.

5.2 Polygon gates

5.2.1 Definition

A polygon gate is a region in the plane with a closed piecewise linear boundary. That is, it is bounded by a closed path, composed of a finite sequence of straight line segments. A point joining two consecutive line segments (sides) is a vertex of the polygon.

For simple polygons (polygons whose boundaries do not cross), the set of points in the plane enclosed by the polygon forms the interior of the polygon, the set of points on the polygon itself forms its boundary, and the set of points surrounding the polygon forms its exterior.

For non-simple polygons, (*i.e.*, polygons that have crossing boundaries), the winding number method (also known as the parity rule method, the even/odd winding rule method or the alternate filling method) is used to define the interior and the exterior of a polygon. For any point a ray is drawn to infinity in any direction. The number of times this ray crosses the boundary is equal to the winding number. If the winding number is odd then the point is taken to be inside the polygon, otherwise it is outside. In practice, this approach needs to be used with care to avoid or address issues resulting from the ray crossing at a vertex. Figure

```

<gating:RectangleGate gating:id="Rect3D" customAttributes="allowed">
  <data-type:custom_info>
    SSC-H is an unbound dimension
  </data-type:custom_info>
  <gating:dimension gating:min="100" gating:max="250"
    gating:compensation-ref="uncompensated">
    <data-type:fcs-dimension data-type:name="FSC-H"/>
  </gating:dimension>
  <gating:dimension gating:min="50" gating:compensation-ref="FCS">
    <data-type:fcs-dimension data-type:name="SSC-H"/>
  </gating:dimension>
  <gating:dimension gating:min="0" gating:max="300"
    gating:compensation-ref="uncompensated">
    <data-type:fcs-dimension data-type:name="FL1-H"/>
  </gating:dimension>
</gating:RectangleGate>

```

Example 14: Three-dimensional hyper-rectangular gate definition.

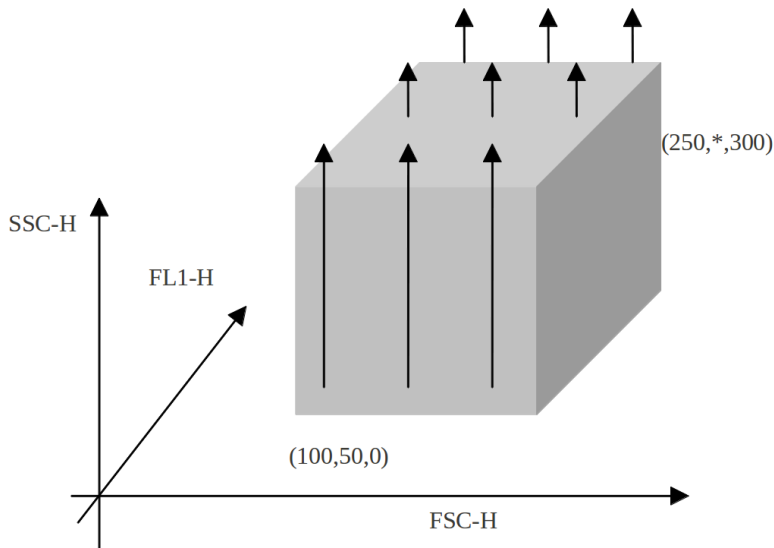


Figure 11: Example of a 3-dimensional rectangular gate with one unbound dimension.

12 shows the mentioned algorithm, which is only demonstrated to define the interior of non-simple polygons, not to standardize the membership decision algorithm itself. An arbitrary algorithm giving the same results may be used to compute polygon gates in compliant software or hardware implementations. Please note that there are also other polygon filling algorithms that are not compatible with this specification, *e.g.*, the non-zero winding rule or the non-exterior method [19–21].

A polygon gate is defined by a sequence of at least three vertices. The boundaries are drawn between consequent vertices in the ordered sequence indicated by the order of *vertex* elements. The polygon is automatically closed, *i.e.*, the last boundary is drawn from the last vertex back to the first one. The events in the interior of the polygon and the events on the boundary are

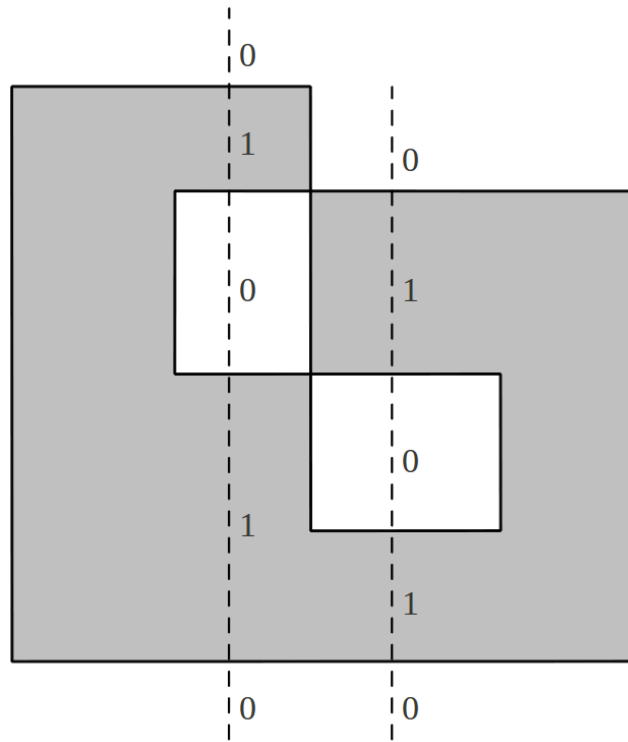


Figure 12: Winding number method used to determine the interior of non-simple polygons.

considered to be in the gate (*i.e.*, the boundary is considered as inclusive).

5.2.2 Syntax specification

Polygon gates shall be specified by the *PolygonGate* element as follows:

- (a) The *id* attribute shall be used to identify the gate for further referencing purposes, *e.g.*, as described in sections 4.4 and 5.5.
- (b) The *parent_id* attribute may be used to reference a parent gate as described in section 4.4.
- (c) A *custom_info* element may be placed at the beginning of the *PolygonGate* element. The *custom_info* element may contain any additional custom information related to this gate; the format of this element (*i.e.*, its attributes, sub elements, etc.) is not constrained by this specification as long as the document remains a valid XML.
- (d) Two dimensions shall be described by two *dimension* elements placed as at the beginning of the *PolygonGate* element or right after the *custom_info* element if the *custom_info* element is used.
- (e) Each *dimension* element shall specify compensation by including the *compensation-ref* attribute as specified in section 4.2. The value of this attribute may be one of the following: *FCS*, *uncompensated*, or the identifier of a spectrum matrix defined within the Gating-ML file.

- (f) Each *dimension* element may use the *transformation-ref* attribute to reference a scale transformation. If used, the value of this attribute shall contain the identifier of a scale transformation defined within this Gating-ML file (see section 4.2.3).
- (g) Each *dimension* element shall reference a dimension as described in section 4.1, *i.e.*, by including an *fcs-dimension* element or a *new-dimension* element. The same FCS dimension or new dimension may not be used more than once within the gate definition.
- (h) At least three vertices, each defined by a *vertex* element, shall follow after the *dimension* elements.
- (i) Each *vertex* element shall include two *coordinate* sub elements, each of which shall specify a coordinate of the vertex by its *value* attribute. The first coordinate corresponds to the first dimension and the second one to the second dimension as defined at the beginning of the polygon gate definition.

5.2.3 Validity conditions

A polygon gate definition is valid if and only if:

- (a) It is valid according to the XML schema.
- (b) If used, the *parent_id* attribute does not introduce any circular dependency (see also sections 3.3.6, 3.3.7, and 3.4.1).
- (c) The gating dimensions are different from one another, *i.e.*, the values of the *name* attributes are unique among all *fcs-dimension* elements (if used) in the gate definition, and the values of the *transformation-ref* attributes are unique among all *new-dimension* elements (if used) in the gate definition.
- (d) The value of the *compensation-ref* attribute of each *dimension* element equals to either *FCS*, *uncompensated*, or an identifier of a spectrum matrix defined within the Gating-ML file.
- (e) If used, the value of the *transformation-ref* attribute of the *dimension* element shall contain an identifier of a scale transformation defined within the same Gating-ML file.
- (f) If used, the value of the *transformation-ref* attribute of the *new-dimension* element shall contain an identifier of a transformation that creates a new dimension, *i.e.*, an identifier of a parametrized ratio transformation.

5.2.4 Examples

- (a) Example 15 demonstrates a simple triangular polygon gate definition in the FCS-H and SSC-H dimensions. Visual representation of this gate is shown in Figure 13.
- (b) Example 16 demonstrates a pentagon gate (*i.e.*, a polygon gate with 5 vertices) created in log-transformed (see section 6.3) FL1-H and untransformed SSC-H FCS dimensions. Visual representation of this gate is shown in Figure 14. Please note that most of the parametrized transformations will transform scale values to values with the maximum of 1. Specifically,

```

<gating:PolygonGate gating:id="Triangle" customAttributes="also allowed 123">
  <gating:dimension gating:compensation-ref="uncompensated">
    <data-type:fcs-dimension data-type:name="FSC-H" />
  </gating:dimension>
  <gating:dimension gating:compensation-ref="uncompensated">
    <data-type:fcs-dimension data-type:name="SSC-H" />
  </gating:dimension>
  <gating:vertex>
    <gating:coordinate data-type:value="0" />
    <gating:coordinate data-type:value="0" />
  </gating:vertex>
  <gating:vertex>
    <gating:coordinate data-type:value="400" />
    <gating:coordinate data-type:value="0" />
  </gating:vertex>
  <gating:vertex>
    <gating:coordinate data-type:value="400" />
    <gating:coordinate data-type:value="300" />
  </gating:vertex>
</gating:PolygonGate>

```

Example 15: Simple polygon gate definition.

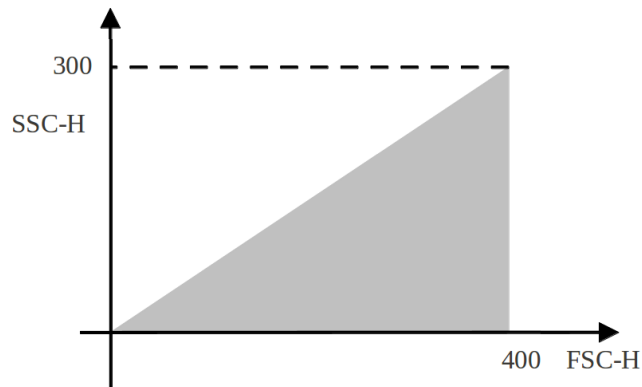


Figure 13: Example of a triangular polygon gate.

a 5-decade *flog* with $T = 10000$ transforms values from $(0, 10000]$ to values $(-, 1]$, where $flog(1) = 0.2$, $flog(10) = 0.4$, $flog(100) = 0.6$, $flog(1000) = 0.8$ and $flog(10000) = 1.0$. In Figure 14, we have included axis labels (tick marks) as they will be returned by the *flog* transformation (*i.e.*, 0.2, 0.4, ...).

- (c) Example 17 demonstrates a non-simple polygon gate (*i.e.*, a polygon with crossing boundaries) created in the FSC-H and SSC-H dimensions. The gate also references a parent gate by specifying the *parent_id* attribute. Visual representation of this gate is shown in Figure 15.

```

<transforms:transformation transforms:id="log" customAtt="allowed here">
  <transforms:flog transforms:T="10000" transforms:M="5" customA="also allowed" />
</transforms:transformation>
<gating:PolygonGate gating:id="Pentagon">
  <gating:dimension gating:compensation-ref="FCS"
    gating:transformation-ref="log">
    <data-type:fcs-dimension data-type:name="FL1-H" />
  </gating:dimension>
  <gating:dimension gating:compensation-ref="uncompensated">
    <data-type:fcs-dimension data-type:name="SSC-H" />
  </gating:dimension>
  <gating:vertex>
    <gating:coordinate data-type:value="0.2" />
    <gating:coordinate data-type:value="50" />
  </gating:vertex>
  <gating:vertex>
    <gating:coordinate data-type:value="0.6" />
    <gating:coordinate data-type:value="50" />
  </gating:vertex>
  <gating:vertex>
    <gating:coordinate data-type:value="0.6" />
    <gating:coordinate data-type:value="150" />
  </gating:vertex>
  <gating:vertex>
    <gating:coordinate data-type:value="0.2" />
    <gating:coordinate data-type:value="150" />
  </gating:vertex>
  <gating:vertex>
    <gating:coordinate data-type:value="0.4" />
    <gating:coordinate data-type:value="100" />
  </gating:vertex>
</gating:PolygonGate>

```

Example 16: Polygon gate definition with transformation.

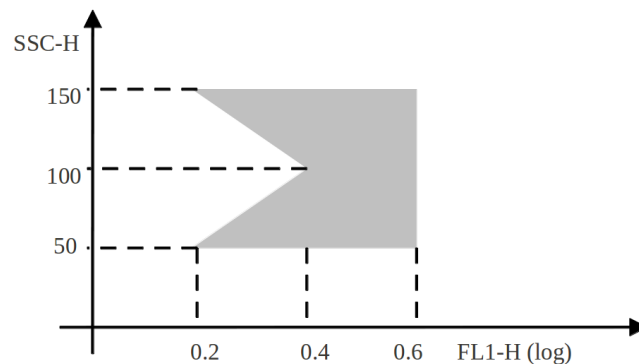


Figure 14: Example of a pentagon polygon gate.

```

<gating:PolygonGate gating:id="NonSimplePolygon" gating:parent_id="Pentagon">
  <gating:dimension gating:compensation-ref="uncompensated">
    <data-type:fcs-dimension data-type:name="FSC-H" />
  </gating:dimension>
  <gating:dimension gating:compensation-ref="uncompensated">
    <data-type:fcs-dimension data-type:name="SSC-H" />
  </gating:dimension>
  <gating:vertex>
    <gating:coordinate data-type:value="50" />
    <gating:coordinate data-type:value="50" />
  </gating:vertex>
  <gating:vertex>
    <gating:coordinate data-type:value="150" />
    <gating:coordinate data-type:value="50" />
  </gating:vertex>
  <gating:vertex>
    <gating:coordinate data-type:value="50" />
    <gating:coordinate data-type:value="150" />
  </gating:vertex>
  <gating:vertex>
    <gating:coordinate data-type:value="150" />
    <gating:coordinate data-type:value="150" />
  </gating:vertex>
</gating:PolygonGate>

```

Example 17: Polygon gate definition with crossing boundaries.

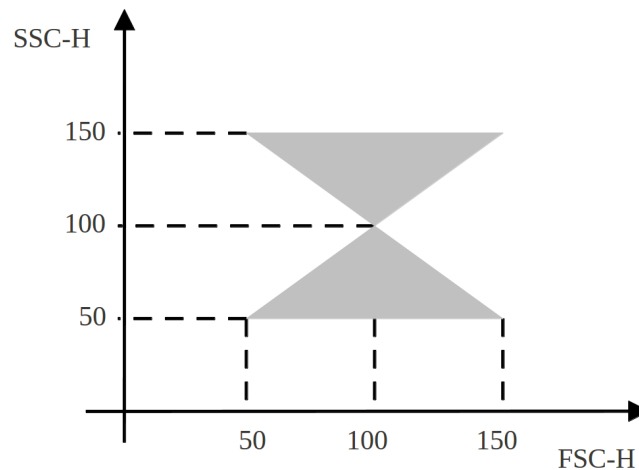


Figure 15: Example of a polygon gate with crossing boundaries.

5.3 Ellipsoid gates

5.3.1 Definition

Elliptical gates are specified by arbitrary ellipsoids in two or more dimensions. Elliptical gates in more than two dimensions are mainly expected to be created by (semi)automated computa-

tional methods. For multidimensional inputs, one can compute the variance for each dimension separately as well as the correlation between the dimensions (covariance). The resulting covariance matrix represents a description of the shape and orientation of the input, taking the first and second order moments of the input into account. Also, if the distribution is known as multivariate normal, then the covariance matrix completely specifies the distribution (except for the position of its means). Specification of elliptical gates by covariance matrices and vectors of means is useful for both creating gates by computational means as well as computing gate membership. From the statistical perspective, the ellipsoid represents concentration of the multivariate normal distribution, which is constant on the surface of the ellipsoid (and equals the square of the Mahalanobis distance). The ellipsoid gate $G(\mu, C, D^2)$ is defined as

$$G(\mu, C, D^2) = \{x : (x - \mu)^T C^{-1} (x - \mu) \leq D^2\}$$

where μ is a column vector specifying the center of the ellipsoid, C^{-1} is the inverse of a covariance matrix, D^2 is the square of the Mahalanobis distance, and T stands for transposition (*i.e.*, converting a row vector to the corresponding column vector). For any $n \geq 2$, μ is a $n \times 1$ vector and C is a $n \times n$ symmetric, positive-definite matrix. The vector μ and matrix C shall be defined in corresponding spaces (*i.e.*, referring to the same list of FCS dimensions).

Let x be the representation of an event e in the space (FCS dimensions) where μ is defined. An event e is considered to be in the ellipsoid gate $G(\mu, C, D^2)$ if and only if x is in $G(\mu, C, D^2)$. As implied by the defining equation, the boundaries are considered to be inclusive, *i.e.*, an event falling on the boundary of an ellipsoid describing a gate is considered to be in the gate.

Let us demonstrate how the representation by covariance matrix relates to the representation by ellipsoid axes in a two-dimensional example. Without restriction of generality, we consider ellipses in the center of the coordinate system (which is for simplicity reasons only, using x rather than $x - x_0$ and y rather than $y - y_0$ for an ellipsoid placed at point $[x_0, y_0]$) and with $D^2 = 1$ (this does not restrict generality as any covariance matrix can be appropriately rescaled). The inside of an ellipse in the center of the coordinate system and aligned with the coordinate axes can be described as $x^2/a^2 + y^2/b^2 \leq 1$ (Figure 16).

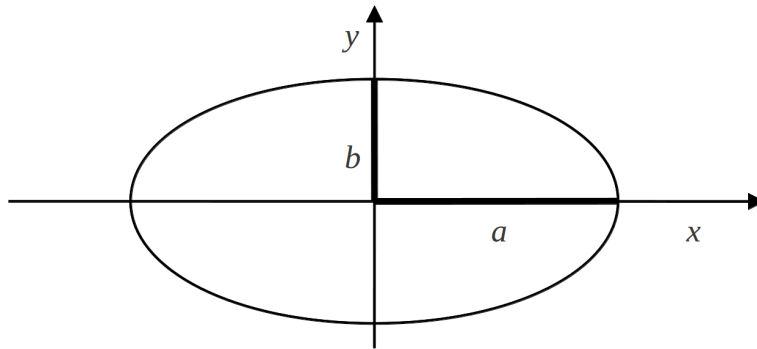


Figure 16: Ellipse with major and minor axes in line with the coordinate system.

Rotating the ellipse by an angle θ assigns $x \rightarrow x \cos(\theta) + y \sin(\theta)$ and $y \rightarrow y \cos(\theta) - x \sin(\theta)$. The equation for points within the rotated ellipse is:

$$x^2 \left(\frac{\cos^2(\theta)}{a^2} + \frac{\sin^2(\theta)}{b^2} \right) + 2xy \cos(\theta) \sin(\theta) \left(\frac{1}{a^2} - \frac{1}{b^2} \right) + y^2 \left(\frac{\sin^2(\theta)}{a^2} + \frac{\cos^2(\theta)}{b^2} \right) \leq 1$$

On the other hand, if we consider the covariance-based ellipsoid definition equation with $D^2 = 1$ and we place the ellipsoid in the center of the coordinate system (*i.e.*, μ becomes a zero vector), then we receive:

$$x^T C^{-1} x \leq 1$$

For a two-dimensional case and considering the symmetry of the covariance (and its inverse) matrix, this equation is specifically:

$$(x, y) \begin{bmatrix} C_{xx}^{-1} & C_{xy}^{-1} \\ C_{xy}^{-1} & C_{yy}^{-1} \end{bmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \leq 1$$

which means:

$$(xC_{xx}^{-1} + yC_{xy}^{-1}, xC_{xy}^{-1} + yC_{yy}^{-1}) \begin{pmatrix} x \\ y \end{pmatrix} \leq 1$$

which results in:

$$x^2 C_{xx}^{-1} + 2xy C_{xy}^{-1} + y^2 C_{yy}^{-1} \leq 1$$

Therefore, we see that the inverse of the covariance matrix describes an ellipse where the major and minor axis and the rotation angle map directly onto its components, specifically:

$$C_{xx}^{-1} = \frac{\cos^2(\theta)}{a^2} + \frac{\sin^2(\theta)}{b^2}$$

$$C_{yy}^{-1} = \frac{\sin^2(\theta)}{a^2} + \frac{\cos^2(\theta)}{b^2}$$

$$C_{xy}^{-1} = \sin(\theta) \cos(\theta) \left(\frac{1}{a^2} - \frac{1}{b^2} \right)$$

In the two-dimensional case, an ellipse described by μ , C , and D^2 may be converted to a description by a center, half axes, and rotation using the calculations described below. First, we calculate the eigenvalues λ_1 and λ_2 of the covariance matrix C as follows:

$$\lambda_{1,2} = \frac{(C_{xx} + C_{yy}) \pm \sqrt{(C_{xx} - C_{yy})^2 + 4C_{xy}^2}}{2}$$

Having λ_1 and λ_2 , the length of the axes a and b can be calculated as

$$a = \sqrt{\lambda_1 D^2}$$

$$b = \sqrt{\lambda_2 D^2}$$

Next, we calculate the eigenvectors X_1 and X_2 of the covariance matrix C . If $C_{xy} \neq 0$, then the eigenvectors $X_{1,2}$ are

$$X_{1,2} = \begin{pmatrix} \lambda_{1,2} - C_{yy} \\ C_{xy} \end{pmatrix}$$

If $C_{xy} = 0$, then the eigenvectors $X_{1,2}$ are

$$X_1 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

$$X_2 = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

The eigenvectors of the covariance matrix specify the direction in which each eigenvalue is achieved, which determines the rotation of the ellipse. For example, we can use the first eigenvector X_1 as follows: Let $X_{1,1}$ be the first component of vector X_1 , and let $X_{1,2}$ be the second component of vector X_1 . If $X_{1,1} \neq 0$, then the rotation θ may be calculated as

$$\theta = \tan^{-1}\left(\frac{X_{1,2}}{X_{1,1}}\right)$$

If $X_{1,1} = 0$, then the rotation θ may be calculated as

$$\theta = \lim_{x \rightarrow +\infty} \tan^{-1}(x) = \pi/2$$

Finally, the vector of means corresponds directly to the center position of the ellipsoid.

5.3.2 Syntax specification

Ellipsoid gates in n dimensions ($n \geq 2$) shall be specified by a $nx1$ vector of means and a nxn covariance matrix (symmetric, positive-definite). To specify an ellipsoid gate the *EllipsoidGate* element shall be used as follows:

- (a) The *id* attribute shall be used to identify the gate for further referencing purposes, *e.g.*, as described in sections 4.4 and 5.5.
- (b) The *parent_id* attribute may be used to reference a parent gate as described in section 4.4.
- (c) A *custom_info* element may be placed at the beginning of the *EllipsoidGate* element. The *custom_info* element may contain any additional custom information related to this gate; the format of this element (*i.e.*, its attributes, sub elements, etc.) is not constrained by this specification as long as the document remains a valid XML.
- (d) Two or more dimensions shall be described by two or more *dimension* elements placed at the beginning of the *EllipsoidGate* element or right after the *custom_info* element if the *custom_info* element is used. For further syntax description purposes, let n be the number of dimensions specified.
- (e) Each *dimension* element shall specify compensation by including the *compensation-ref* attribute as specified in section 4.2. The value of this attribute may be one of the following: *FCS*, *uncompensated*, or the identifier of a spectrum matrix defined within the Gating-ML file.
- (f) Each *dimension* element may use the *transformation-ref* attribute to reference a scale transformation. If used, the value of this attribute shall contain the identifier of a scale transformation defined within this Gating-ML file (see section 4.2.3).
- (g) Each *dimension* element shall reference a dimension as described in section 4.1, *i.e.*, by including an *fcs-dimension* element or a *new-dimension* element. The same FCS dimension or new dimension may not be used more than once within the gate definition.

- (h) The *mean* element shall follow after the *dimension* elements to describe the mean vector (center of the ellipsoid). It shall include *n coordinate* elements, each one specifying a coordinate of the mean point in space by a *value* attribute. The first coordinate corresponds to the first dimension, the second coordinate to the second dimension, etc. The number of *coordinate* elements specified for the *mean* element shall correspond to the number of *dimension* elements specified at the beginning of the ellipsoid gate definition.
- (i) The *covarianceMatrix* element shall follow after the *mean* element.
- (j) The *covarianceMatrix* element shall include *n row* sub elements, each of them including *n entry* sub elements (where *n* is the number of *dimension* elements specified at the beginning of the ellipsoid gate definition).
- (k) Each *entry* element shall specify its value by the *value* attribute.
- (l) A *distanceSquare* element shall follow after the *covarianceMatrix* element and it shall specify the square of the Mahalanobis distance (*i.e.*, D^2) using the *value* attribute.

5.3.3 Validity conditions

An ellipsoid gate definition is valid if and only if:

- (a) It is valid according to the XML schema.
- (b) If used, the *parent_id* attribute does not introduce any circular dependency (see also sections 3.3.6, 3.3.7, and 3.4.1).
- (c) There are *n dimension* elements describing *n* dimensions where $n \geq 2$.
- (d) The gating dimensions are different from one another, *i.e.*, the values of the *name* attributes are unique among all *fcs-dimension* elements (if used) in the gate definition, and the values of the *transformation-ref* attributes are unique among all *new-dimension* elements (if used) in the gate definition.
- (e) The value of the *compensation-ref* attribute of each *dimension* element equals to either *FCS*, *uncompensated*, or an identifier of a spectrum matrix defined within the Gating-ML file.
- (f) If used, the value of the *transformation-ref* attribute of the *dimension* element shall contain an identifier of a scale transformation defined within the same Gating-ML file.
- (g) If used, the value of the *transformation-ref* attribute of the *new-dimension* element shall contain an identifier of a transformation that creates a new dimension, *i.e.*, an identifier of a parametrized ratio transformation.
- (h) The *covarianceMatrix* element includes *n row* sub elements.
- (i) Each *row* element includes *n entry* elements.
- (j) The *covarianceMatrix* element defines a valid covariance matrix, *i.e.*, a symmetric, positive-definite matrix.

5.3.4 Examples

- (a) Example 18 demonstrates a two-dimensional ellipsoid gate in the FCS-H and SSC-H dimensions. Visual representation of this gate is shown in Figure 17.

```
<gating:EllipsoidGate gating:id="myEllipse">
  <gating:dimension gating:compensation-ref="FCS">
    <data-type:fcs-dimension data-type:name="FCS-H" />
  </gating:dimension>
  <gating:dimension gating:compensation-ref="FCS">
    <data-type:fcs-dimension data-type:name="SSC-H" />
  </gating:dimension>
  <gating:mean>
    <gating:coordinate data-type:value="400" />
    <gating:coordinate data-type:value="200" />
  </gating:mean>
  <gating:covarianceMatrix>
    <!-- a=200, b=175, rotation=0dg -->
    <gating:row>
      <gating:entry data-type:value="40000" />
      <gating:entry data-type:value="0" />
    </gating:row>
    <gating:row>
      <gating:entry data-type:value="0" />
      <gating:entry data-type:value="30625" />
    </gating:row>
  </gating:covarianceMatrix>
  <gating:distanceSquare data-type:value="1" />
</gating:EllipsoidGate>
```

Example 18: Two-dimensional ellipsoid gate definition.

- (b) Example 19 demonstrates a rotated two-dimensional ellipsoid gate in the FCS-H and SSC-H dimensions. A *custom.info* element is included to demonstrate the option of including customized information. Visual representation of this gate is shown in Figure 18.
- (c) Example 20 demonstrates a three-dimensional ellipsoid gate in FL1-H, FL2-H and FL3-H with the center point at coordinates [0.2, 0.3, 0.25] specified on a Logicle scale (see section 6.5).

5.4 Quadrant gates

5.4.1 Definition

A quadrant gate is a collection of n -dimensional rectangular gates, which are non-overlapping and fill up the whole event space. While this includes typical quadrant gates, note that it is more generic than what the term “quadrant” is typically used for. A quadrant gate is defined by a set of dividers, which split the event space at specified value(s). Specific quadrants are further defined by referencing dividers and stating locations of representative values as shown

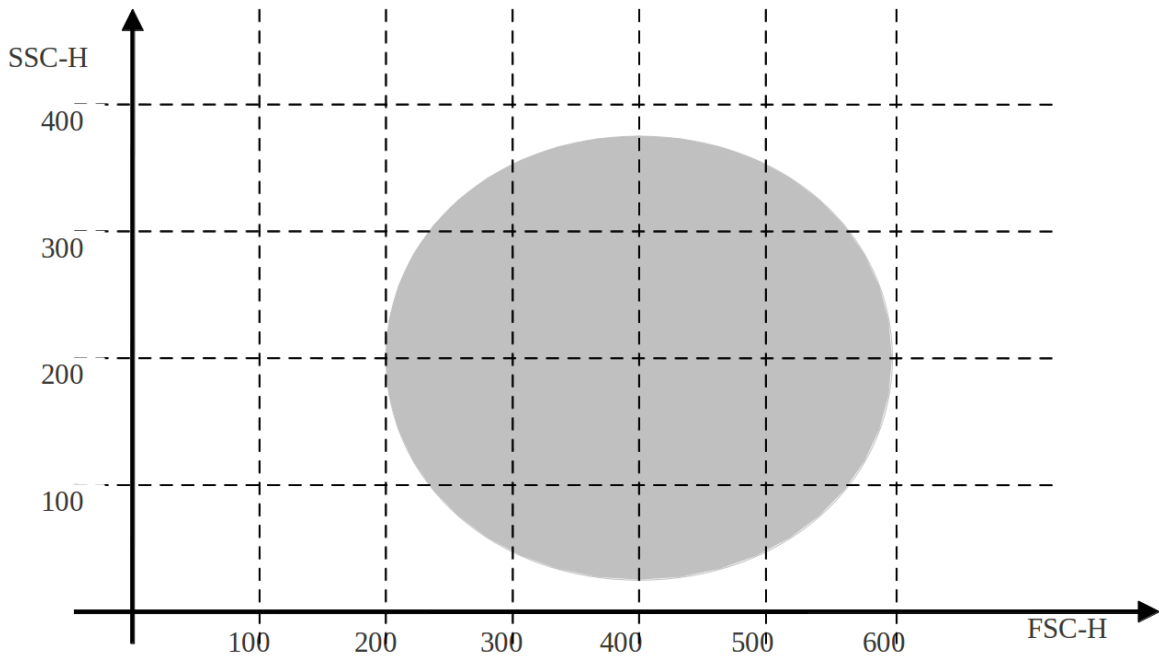


Figure 17: Example of a two dimensional ellipsoid gate.

```

<gating:EllipsoidGate gating:id="myEllipse2">
  <data-type:custom_info a="37" b="30" rotation="-45dg" />
  <gating:dimension gating:compensation-ref="uncompensated">
    <data-type:fcs-dimension data-type:name="FSC-H" />
  </gating:dimension>
  <gating:dimension gating:compensation-ref="uncompensated">
    <data-type:fcs-dimension data-type:name="SSC-H" />
  </gating:dimension>
  <gating:mean>
    <gating:coordinate data-type:value="40" />
    <gating:coordinate data-type:value="40" />
  </gating:mean>
  <gating:covarianceMatrix>
    <gating:row>
      <gating:entry data-type:value="1134.5" />
      <gating:entry data-type:value="-234.5" />
    </gating:row>
    <gating:row>
      <gating:entry data-type:value="-234.5" />
      <gating:entry data-type:value="1134.5" />
    </gating:row>
  </gating:covarianceMatrix>
  <gating:distanceSquare data-type:value="1" />
</gating:EllipsoidGate>

```

Example 19: Rotated two-dimensional ellipsoid gate definition.

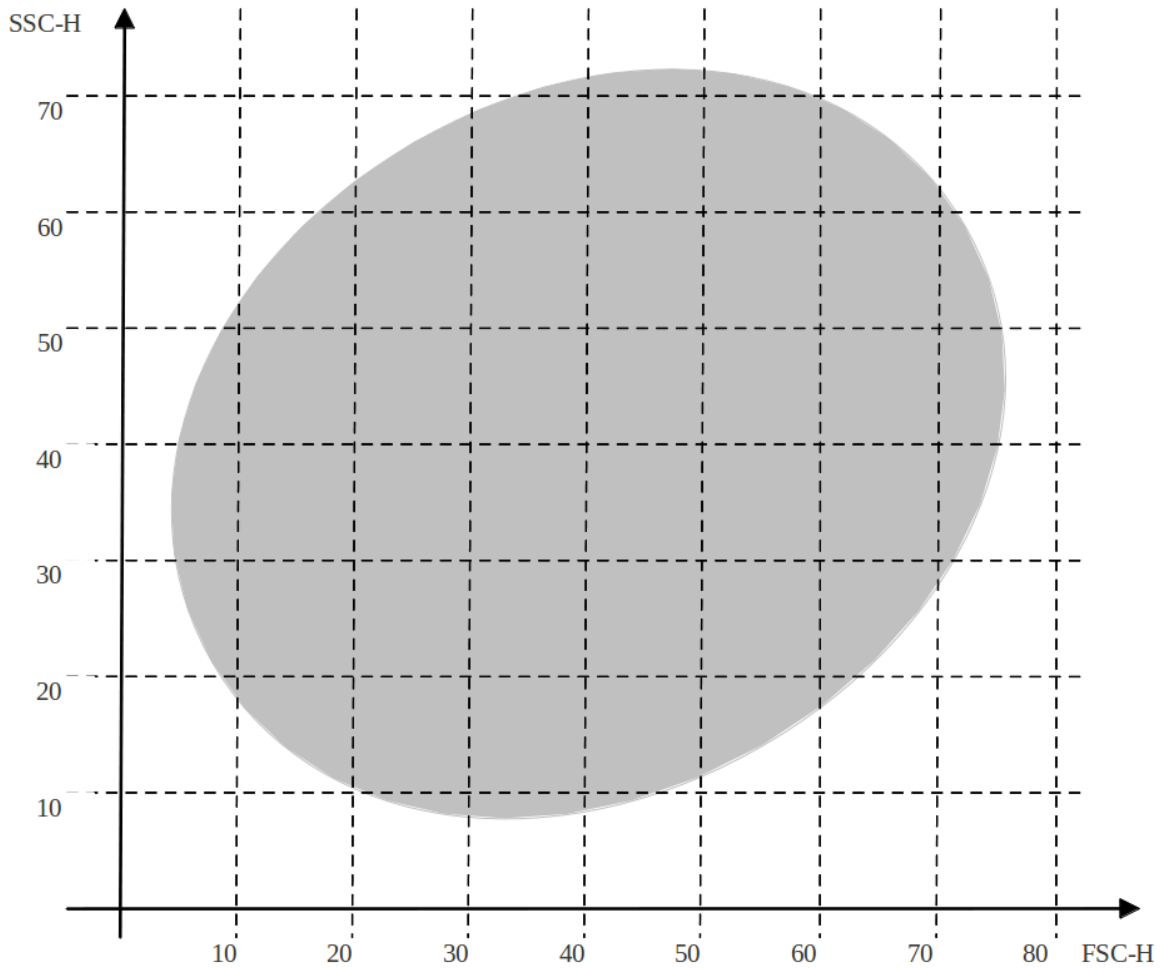


Figure 18: Example of a rotated two dimensional ellipsoid gate.

in example Figure 19, which demonstrates a quadrant gate splitting the events based on the FSC-H and SSC-H dimensions in 6 quadrants: Q1, Q2, Q3, Q4, Q5, and Q6. These quadrants are defined by means of two dividers – one for the FSC-H dimension at two values: 500 and 1000, and another one for the SSC-H dimension at the value of 400.

Specific quadrants are defined by referencing dividers and stating locations of representative values. For example, the Q5 quadrant can be defined by stating 600 as the value for the FSC-H divider (any value between 500 and 1000 would defined the same quadrant) and 200 for the SSC-H divider (any value smaller than 400 would define the same quadrant).

In this example, an event e is considered to be in the quadrant Q5 if and only if its FSC-H value $e(d_{FSC-H})$ is greater than or equal to 500 while being less than a thousand, *i.e.*, $500 \leq e(d_{FSC-H}) < 1000$, and its SSC-H value $e(d_{SSC-H})$ is less than 400, *i.e.*, $e(d_{SSC-H}) < 400$. Please note that the axes do not serve as implicit dividers; however, an explicit divider may be placed at the value zero if required. Technically, this would split the FCS dimension into two groups: (i) less than zero and (ii) greater than or equal to zero.

```

<transforms:transformation transforms:id="adjustedLogicle">
  <transforms:logicle transforms:T="10000" transforms:W="0.7"
    transforms:M="4.5" transforms:A="0" />
</transforms:transformation>
<gating:EllipsoidGate gating:id="myEllipse3">
  <gating:dimension gating:compensation-ref="FCS"
    gating:transformation-ref="adjustedLogicle">
    <data-type:fcs-dimension data-type:name="FL1-H" />
  </gating:dimension>
  <gating:dimension gating:compensation-ref="FCS"
    gating:transformation-ref="adjustedLogicle">
    <data-type:fcs-dimension data-type:name="FL2-H" />
  </gating:dimension>
  <gating:dimension gating:compensation-ref="FCS"
    gating:transformation-ref="adjustedLogicle">
    <data-type:fcs-dimension data-type:name="FL3-H" />
  </gating:dimension>
  <gating:mean>
    <gating:coordinate data-type:value="0.2" />
    <gating:coordinate data-type:value="0.3" />
    <gating:coordinate data-type:value="0.25" />
  </gating:mean>
  <gating:covarianceMatrix>
    <gating:row>
      <gating:entry data-type:value="3" />
      <gating:entry data-type:value="-2" />
      <gating:entry data-type:value="11" />
    </gating:row>
    <gating:row>
      <gating:entry data-type:value="-2" />
      <gating:entry data-type:value="0.3" />
      <gating:entry data-type:value="-1" />
    </gating:row>
    <gating:row>
      <gating:entry data-type:value="11" />
      <gating:entry data-type:value="-1" />
      <gating:entry data-type:value="2" />
    </gating:row>
  </gating:covarianceMatrix>
  <gating:distanceSquare data-type:value="1" />
</gating:EllipsoidGate>

```

Example 20: Three-dimensional ellipsoid gate definition.

5.4.2 Syntax specification

In order to define a quadrant gate the *QuadrantGate* element shall be used as follows:

(a) The *id* attribute shall be used to identify the quadrant gate. However, please note that

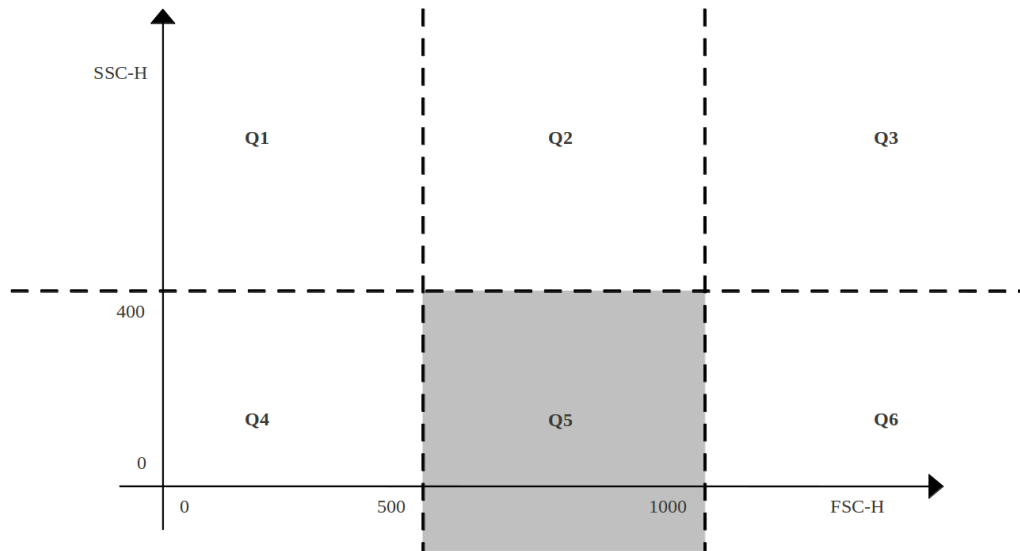


Figure 19: Demonstration of quadrant gates definitions.

this identifier shall not be used for further referencing purposes (quadrant gate is in fact a set of gates). Only specific quadrants may be referenced as parent gates (section 4.4) or parts of Boolean collections of gates (section 5.5).

- (b) The *parent_id* attribute may be used to reference a parent gate as described in section 4.4. If this is the case, the parent gate is applicable to all quadrants defined by the quadrant gate.
- (c) A *custom_info* element may be placed at the beginning of the *QuadrantGate* element. The *custom_info* element may contain any additional custom information related to this gate; the format of this element (*i.e.*, its attributes, sub elements, etc.) is not constrained by this specification as long as the document remains a valid XML.
- (d) One or more dividers shall be specified by one or more *divider* elements placed at the beginning of the *QuadrantGate* element or right after the *custom_info* element if the *custom_info* element is used. For further syntax description purposes, let n be the number of dividers specified.
- (e) Each divider element shall specify its identifier as the value of its *id* attribute. This identifier shall further be used to reference the divider from a quadrant definition as specified below.
- (f) Similarly to a description of a dimension, each *divider* element shall specify compensation by including the *compensation-ref* attribute as specified in section 4.2. The value of this attribute may be one of the following: *FCS*, *uncompensated*, or the identifier of a spectrum matrix defined within the Gating-ML file.
- (g) Similarly to a description of a dimension, each *divider* element may use the *transformation-ref* attribute to reference a scale transformation. If used, the value of this attribute shall contain the identifier of a scale transformation defined within this Gating-ML file (see section 4.2.3).

- (h) Each *divider* element shall reference a dimension as described in section 4.1, *i.e.*, by including an *fcs-dimension* element or a *new-dimension* element. The same FCS dimension or new dimension may not be used more than once within the gate definition.
- (i) Each *divider* element shall specify at least one divider value by including one or more *value* sub elements. Including multiple *value* elements results in “splitting” a particular FCS dimension at multiple values such as shown for the FCS-H dimension in Figure 19.
- (j) The *value* element shall contain a numerical value, *i.e.*, the contents of the *value* element shall be a numerical value. In addition, values shall be different from each other and arranged in an increasing order if multiple values (*i.e.*, *value* elements) are used for a single divider.
- (k) The divider definition(s) shall be followed by one or more quadrants defined by one or more *Quadrant* elements. Please note that not every quadrant needs to be defined (*e.g.*, example 23).
- (l) Each *Quadrant* element shall use the *id* attribute to identify the quadrant for further referencing purposes, *i.e.*, to be referenceable as a parent gate (section 4.4) or in a Boolean collections of gates (section 5.5).
- (m) Each *Quadrant* element shall specify *m position* sub elements where *m* is greater or equal to 1 while being less or equal to *n* where *n* is the number of dividers specified at the beginning of the quadrant gate definition. Specifying less positions than the number of dividers means that the omitted dividers are not part of the definition of a particular quadrant, *i.e.*, the quadrant reaches to both sides of the omitted dividers in that dimension (*e.g.*, example 23).
- (n) Each *position* element shall include a *divider_ref* attribute referencing a divider by its identifier. No two *position* elements within the same *Quadrant* element shall reference the same divider.
- (o) Each *position* element shall include a *location* attribute giving a representative value for events in this quadrant considering the divider values specified earlier. The value of the location attribute shall be different from the value(s) specified within the referenced divider element.

5.4.3 Validity conditions

A quadrant gate definition is valid if and only if:

- (a) It is valid according to the XML schema.
- (b) If used, the *parent_id* attribute does not introduce any circular dependency (see also sections 3.3.6, 3.3.7 and 3.4.1).
- (c) The dimensions are different from one another, *i.e.*, the values of the *name* attributes are unique among all *fcs-dimension* elements (if used) across all *divider* elements in the gate definition, and the values of the *transformation-ref* attributes are unique among all *new-dimension* elements (if used) across all *divider* elements in the gate definition.

- (d) The value of the *compensation-ref* attribute of each *divider* element equals to either *FCS*, *uncompensated*, or an identifier of a spectrum matrix defined within the Gating-ML file.
- (e) If used, the value of the *transformation-ref* attribute of the *divider* element shall contain an identifier of a scale transformation defined within the same Gating-ML file.
- (f) If used, the value of the *transformation-ref* attribute of the *new-dimension* element shall contain an identifier of a transformation that creates a new dimension, *i.e.*, an identifier of a parametrized ratio transformation.
- (g) Numerical values in the *value* elements are different from each other and sorted in an increasing order within a single *divider* element (applicable if multiple *value* elements for a single *divider* element are used).
- (h) The value of the *location* attribute is different from the value(s) specified within the referenced *divider* element.
- (i) Dividers referenced from different *position* elements that are sub elements of the same *Quadrant* element shall be different from each other.

5.4.4 Examples

- (a) Example 21 demonstrates a quadrant gate definition that creates 6 quadrants by splitting the FSC-H dimension at two distinct values and the SSC-H dimension at one value. It defines the quadrant gate introduced and shown earlier in Figure 19.
- (b) Example 22 demonstrates a simple one dimensional quadrant gate definition that splits the FL4-W dimension at two values to create 3 quadrants in total. Visual representation of this quadrant gate is shown in Figure 20.

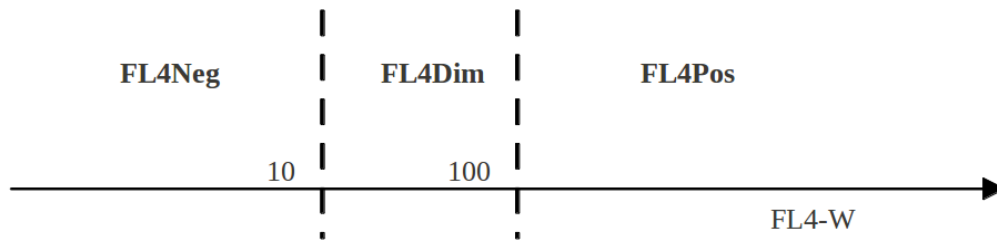


Figure 20: Example of a one dimensional quadrant gate.

- (c) Example 23 demonstrates a quadrant gate definition that defines 3 quadrants by splitting the Hyperlog scale (section 6.6) of FL1-A at 0.5 and the Logicle scale (section 6.5) at 0 and 0.25. The CS1 quadrant does not reference the FL1-A divider and therefore, it extends across all FL1-H values. The CS2 and CS3 quadrants reference both, the FL1-A and the FL2-H dividers. In addition, the CS2 and CS3 quadrants are enclosed from the top as well as the bottom (*i.e.*, $0 \leq \text{Logicle}(\text{FL2-H}) < 0.25$) since the FL2-H divider their position falls between the two values defined by the FL2-H divider. Additional quadrants (*i.e.*, $\text{Logicle}(\text{FL2-H}) < 0$) are not defined. Visual representation of this quadrant gate is shown in Figure 21.

```

<gating:QuadrantGate gating:id="myQuadrantGate">
  <data-type:custom_info>FSC-H is split at 500 and 1000</data-type:custom_info>
  <gating:divider gating:id="FSC" gating:compensation-ref="uncompensated">
    <data-type:fcs-dimension data-type:name="FSC-H" />
    <gating:value>500</gating:value>
    <gating:value>1000</gating:value>
  </gating:divider>
  <gating:divider gating:id="SSC" gating:compensation-ref="uncompensated">
    <data-type:fcs-dimension data-type:name="SSC-H" />
    <gating:value>400</gating:value>
  </gating:divider>
  <gating:Quadrant gating:id="Q1">
    <gating:position gating:divider_ref="FSC" gating:location="200" />
    <gating:position gating:divider_ref="SSC" gating:location="600" />
  </gating:Quadrant>
  <gating:Quadrant gating:id="Q2">
    <gating:position gating:divider_ref="FSC" gating:location="600" />
    <gating:position gating:divider_ref="SSC" gating:location="600" />
  </gating:Quadrant>
  <gating:Quadrant gating:id="Q3">
    <gating:position gating:divider_ref="FSC" gating:location="1700" />
    <gating:position gating:divider_ref="SSC" gating:location="600" />
  </gating:Quadrant>
  <gating:Quadrant gating:id="Q4">
    <gating:position gating:divider_ref="FSC" gating:location="200" />
    <gating:position gating:divider_ref="SSC" gating:location="200" />
  </gating:Quadrant>
  <gating:Quadrant gating:id="Q5">
    <gating:position gating:divider_ref="FSC" gating:location="600" />
    <gating:position gating:divider_ref="SSC" gating:location="200" />
  </gating:Quadrant>
  <gating:Quadrant gating:id="Q6">
    <gating:position gating:divider_ref="FSC" gating:location="1700" />
    <gating:position gating:divider_ref="SSC" gating:location="200" />
  </gating:Quadrant>
</gating:QuadrantGate>

```

Example 21: Quadrant gate definition splitting the FSC-H dimension at multiple values.

5.5 Boolean gates

5.5.1 Definition

A Boolean gate is a gate created based on other gates using Boolean operators – AND, OR, or NOT. NOT is a unary operator. The AND and OR operators are extended to allow for two or more operands. The operators have the following semantic:

- (a) Let gate G be created as $G = NOT(G_0)$. An event e is in the gate G if and only if e is not in the gate G_0 . Formally, $e \in G \Leftrightarrow e \notin G_0$.


```

<gating:QuadrantGate gating:id="myQuadrantGate2">
  <gating:divider gating:id="FL4" gating:compensation-ref="FCS">
    <data-type:fcs-dimension data-type:name="FL4-W" />
    <gating:value>10</gating:value>
    <gating:value>100</gating:value>
  </gating:divider>
  <gating:Quadrant gating:id="FL4Neg">
    <gating:position gating:divider_ref="FL4" gating:location="0" />
  </gating:Quadrant>
  <gating:Quadrant gating:id="FL4Dim">
    <gating:position gating:divider_ref="FL4" gating:location="50" />
  </gating:Quadrant>
  <gating:Quadrant gating:id="FL4Pos">
    <gating:position gating:divider_ref="FL4" gating:location="500" />
  </gating:Quadrant>
</gating:QuadrantGate>

```

Example 22: One dimensional quadrant gate definition.

```

<gating:QuadrantGate gating:id="myQuadrantGate3">
  <data-type:custom_info note="Assuming transformations defined elsewhere." />
  <gating:divider gating:id="FL1" gating:compensation-ref="FCS"
    gating:transformation-ref="myHyperlog1">
    <data-type:fcs-dimension data-type:name="FL1-A"/>
    <gating:value>0.5</gating:value>
  </gating:divider>
  <gating:divider gating:id="FL2" gating:compensation-ref="FCS"
    gating:transformation-ref="myLogicle1">
    <data-type:fcs-dimension data-type:name="FL2-A"/>
    <gating:value>0</gating:value>
    <gating:value>0.25</gating:value>
  </gating:divider>
  <gating:Quadrant gating:id="CS1">
    <gating:position gating:divider_ref="FL2" gating:location="1" />
  </gating:Quadrant>
  <gating:Quadrant gating:id="CS2">
    <gating:position gating:divider_ref="FL1" gating:location="0.1" />
    <gating:position gating:divider_ref="FL2" gating:location="0.1" />
  </gating:Quadrant>
  <gating:Quadrant gating:id="CS3">
    <gating:position gating:divider_ref="FL1" gating:location="0.6" />
    <gating:position gating:divider_ref="FL2" gating:location="0.1" />
  </gating:Quadrant>
</gating:QuadrantGate>

```

Example 23: Quadrant gate definition demonstrating “merging” of quadrants.

(b) Let gate G be created as $G = AND(G_1, G_2, \dots, G_n), n \in \mathbb{N}, n \geq 2$. An event e is in the

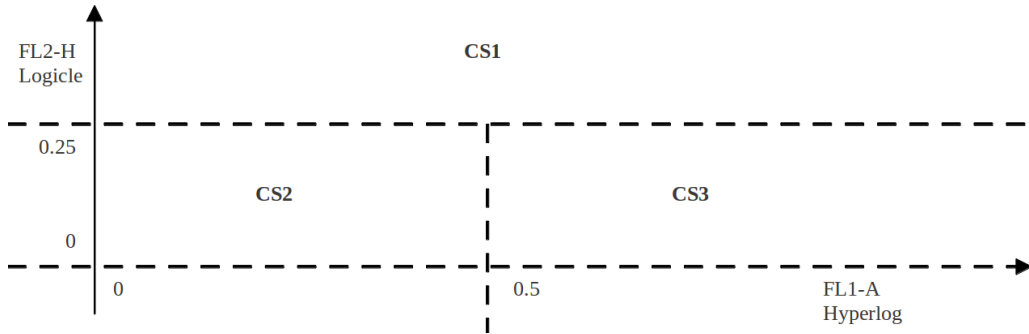


Figure 21: Example of a quadrant gate with “merged” quadrants.

gate G if and only if e is in all gates G_i for all $i \in \{1, 2, \dots, n\}$. Formally, $e \in G \Leftrightarrow \forall i \in \{1, 2, \dots, n\}, e \in G_i$.

- (c) Let gate G be created as $G = OR(G_1, G_2, \dots, G_n), n \in \mathbb{N}, n \geq 2$. An event e is in the gate G if and only if e is at least one of the gates G_i for $i \in \{1, 2, \dots, n\}$. Formally, $e \in G \Leftrightarrow \exists i \in \{1, 2, \dots, n\}, e \in G_i$.

Note that the XOR gate is not supported; however, a XOR gate may be created by combining the supported operators, *e.g.*, $XOR(A, B) = OR(AND(A, NOT(B)), AND(NOT(A), B))$. Note that the in-line definition of the operands of Boolean gates is not supported; however, a complement of a defined gate may be directly referenced (see the *use-as-complement* attribute), which allows for the creation of some simple expressions, such as $AND(A, NOT(B))$. More complicated expressions may be constructed step-wise by defining the sub-operands as separate gates that can be further combined.

5.5.2 Syntax specification

In order to define a Boolean gate the *BooleanGate* element shall be used as follows:

- The *id* attribute shall be used to identify the gate for further referencing purposes, *e.g.*, as described in sections 4.4 and 5.5.
- The *parent_id* attribute may be used to reference a parent gate as described in section 4.4.
- A *custom_info* element may be placed at the beginning of the *BooleanGate* element. The *custom_info* element may contain any additional custom information related to this gate; the format of this element (*i.e.*, its attributes, sub elements, etc.) is not constrained by this specification as long as the document remains a valid XML.
- Either one of the *and*, *or* or *not* elements shall be placed at the beginning of the *BooleanGate* element or right after the *custom_info* element if the *custom_info* element is used.
- If either the *and* or the *or* element is used, then it shall reference two or more operands by including two or more *gateReference* elements. If the *not* element is used, then it shall reference a single operand by including a single *gateReference* element. Operands of Boolean gates represent gates that Boolean operations are applied to. A Boolean AND

operation shall be computed for the and operator so that an event is in the “and gate” if and only if it is in all the referenced operand gates. A Boolean OR operation shall be computed for the or operator so that an event is in the “or gate” if and only if it is in at least one of the referenced operand gates. A Boolean NOT operation shall be computed for the not operator so that an event is in the “not gate” if and only if it is not in the referenced operand gate.

- (f) The *gateReference* element shall use the *ref* attribute to reference another rectangle, polygon, ellipsoid or Boolean gate, or a specific quadrant from a quadrant gate definition by its identifier, *i.e.*, by the *id* attribute of a *RectangleGate* element, *PolygonGate* element, *EllipsoidGate* element, *BooleanGate* element or *Quadrant* element defined elsewhere (*i.e.*, outside of this Boolean gate definition) in the Gating-ML file.
- (g) The *gateReference* element may use the optional boolean *use-as-complement* attribute, which may be set to either *false* (default) or *true*. Omitting this attribute has the same effect as setting it to *false*. Setting this attribute to *true* indicates that the complement of the particular operand (*i.e.*, NOT(...)) shall be used instead of the operand itself.

5.5.3 Validity conditions

A Boolean gate definition is valid if and only if:

- (a) It is valid according to the XML schema.
- (b) If used, the *parent_id* attribute does not introduce any circular dependency (see also sections 3.3.6, 3.3.7 and 3.4.1).
- (c) The values of the *ref* attribute of all included *gateReference* elements reference a valid rectangle, polygon, ellipsoid or Boolean gate, or a specific quadrant from a quadrant gate definition.
- (d) The Boolean gate does not introduce any circular dependency, *i.e.*, none of the operands are dependent (directly or indirectly) on this Boolean gate (see also sections 3.3.6, 3.3.7 and 3.4.1).

5.5.4 Examples

- (a) Example 24 demonstrates an “OR” Boolean gate combining a rectangle gate with an ellipsoid gate. Visual representation of this Boolean gate is shown in Figure 22.
- (b) Example 25 demonstrates a “NOT” Boolean gate applied on the Boolean gate defined in example 25. This “NOT” gate contains all the events that are outside of the Boolean gate shown in Figure 22.
- (c) Example 26 demonstrates an “AND” Boolean gate that creates the intersection of the ellipsoid gate “myEllipse” defined in example 18, the polygon gate “Pentagon” defined in example 16 and the complement of the “FL4Dim” quadrant from the quadrant gate defined in example 22. This can also be expressed as AND(“myEllipse”, “Pentagon”, NOT(“FL4Dim”)). This “AND” gate is supposed to be applied on the population defined

```

<gating:BooleanGate gating:id="myBoolean">
  <gating:or>
    <gating:gateReference gating:ref="R1" />
    <gating:gateReference gating:ref="E1" />
  </gating:or>
</gating:BooleanGate>
<gating:RectangleGate gating:id="R1">
  <gating:dimension gating:compensation-ref="uncompensated"
    gating:min="10" gating:max="30">
    <data-type:fcs-dimension data-type:name="FSC-H" />
  </gating:dimension>
  <gating:dimension gating:compensation-ref="uncompensated"
    gating:min="10" gating:max="20">
    <data-type:fcs-dimension data-type:name="SSC-H" />
  </gating:dimension>
</gating:RectangleGate>
<gating:EllipsoidGate gating:id="E1">
  <gating:dimension gating:compensation-ref="uncompensated">
    <data-type:fcs-dimension data-type:name="FSC-H" />
  </gating:dimension>
  <gating:dimension gating:compensation-ref="uncompensated">
    <data-type:fcs-dimension data-type:name="SSC-H" />
  </gating:dimension>
  <gating:mean>
    <gating:coordinate data-type:value="30" />
    <gating:coordinate data-type:value="15" />
  </gating:mean>
  <gating:covarianceMatrix>
    <gating:row>
      <gating:entry data-type:value="100" />
      <gating:entry data-type:value="0" />
    </gating:row>
    <gating:row>
      <gating:entry data-type:value="0" />
      <gating:entry data-type:value="25" />
    </gating:row>
  </gating:covarianceMatrix>
  <gating:distanceSquare data-type:value="1" />
</gating:EllipsoidGate>

```

Example 24: Boolean OR gate definition combining a rectangle gate with an ellipse gate.

```

<gating:BooleanGate gating:id="myBoolean2">
  <gating:not><gating:gateReference gating:ref="myBoolean" /></gating:not>
</gating:BooleanGate>

```

Example 25: Boolean NOT gate definition.

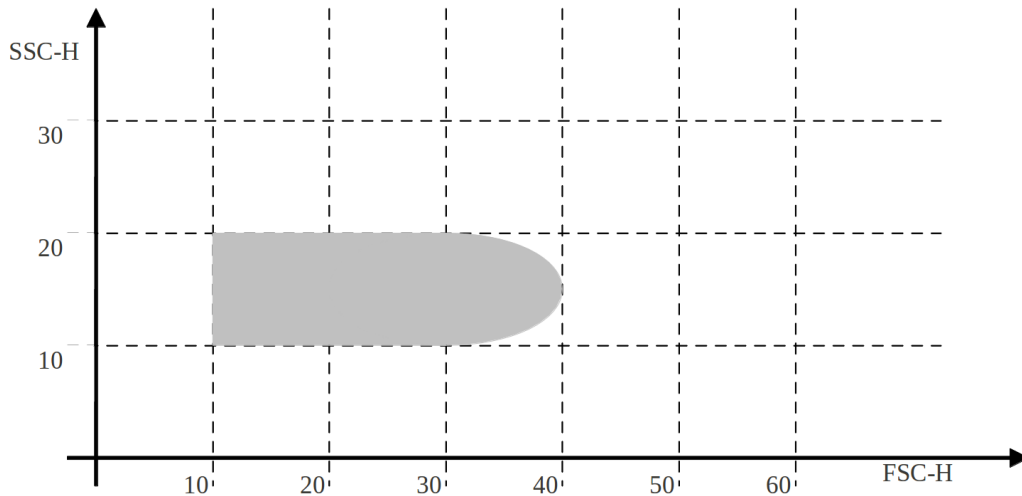


Figure 22: Example of a Boolean OR gate.

by the “Range_FCS_o100” parent gate defined in example 12. In terms of the gating (event filtering) operation, including the “Range_FCS_o100” gate as another operand of the “AND” gate rather than listing it as a parent gate would produce exactly the same results.

```
<gating:BooleanGate gating:id="myBoolean3" gating:parent_id="Range_FCS_o100">
  <gating:and>
    <gating:gateReference gating:ref="myEllipse" />
    <gating:gateReference gating:ref="Pentagon" />
    <gating:gateReference gating:ref="FL4Dim" gating:use-as-complement="true" />
  </gating:and>
</gating:BooleanGate>
```

Example 26: Boolean AND gate definition with one operand used as complement.

6 Scale transformation description

6.1 Bounding transformations

6.1.1 Definition

A boundary may be defined for any scaling transformation as described further in this section, or for any additional transformation as described in section 8. A boundary is defined by the *bound* function as follows:

$$\text{bound}(x, \text{bound}_{\min}, \text{bound}_{\max}) = \begin{cases} \text{bound}_{\min} & \text{if } x < \text{bound}_{\min} \\ \text{bound}_{\max} & \text{if } x > \text{bound}_{\max} \\ x & \text{otherwise} \end{cases}$$

- $x \in \mathbb{R}$ is the value that the boundary is applied to, such as the result of a scaling transformation, or the result of an additional transformation as described in section 8.
- $bound_{min} \in \mathbb{R}$ is the boundary lower bound.
- $bound_{max} \in \mathbb{R}, bound_{max} \geq bound_{min}$ is the boundary upper bound.

6.1.2 Use case description

A boundary restricts a value x (*i.e.* the result of a transformation that the boundary is defined for) to the $[bound_{min}, bound_{max}]$ interval. Using a boundary allows for simple unambiguous encoding of gating performed by software tools that pile “off-scale” events on the “graph axes”. In these cases, if the selected visualization (scaling transformation) is not quite appropriate, certain events could “fall of the graph”. However, instead of “losing” these events, some software tools prefer to shift them to a predefined minimum or maximum, which may affect gate membership of these events. A Gating-ML boundary may be used in order to mimic such behavior in Gating-ML and encode these gates in a reproducible manner.

6.1.3 Syntax specification

A boundary shall be specified by including the *boundMin* attribute, the *boundMax* attribute, or both, in any *transformation* element. If the *boundMin* attribute is not included, then $-\infty$ (negative infinity) shall be considered as the *boundMin* value. If the *boundMax* attribute is not included, then ∞ (infinity) shall be considered as the *boundMax* value.

6.1.4 Validity conditions

A transformation with a boundary is valid if and only if

- It is valid according to the XML schema.
- If both the *boundMin* and the *boundMax* attributes are included, then the value of the *boundMin* attribute is less than or equal to the value of the *boundMax* attribute.

6.1.5 Examples

- Example 27 demonstrates the definition of a Logicle transformation (see section 6.5) with $boundMin = 0.1$ and $boundMax = 0.9$.

```
<transforms:transformation transforms:id="Bound_Logicle"
                        transforms:boundMin="0.1" transforms:boundMax="0.9">
  <transforms:logicle transforms:T="262144" transforms:M="4.5"
                    transforms:W="0.5" transforms:A="0" />
</transforms:transformation>
```

Example 27: Logicle transformation with a boundary restricting its result to the $[0.1,0.9]$ interval.

In order to illustrate the effect of this boundary, let us assume we have a polygon gate that references the Logicle transformation defined in example 27. This polygon gate is defined in example 28.

```

<gating:PolygonGate gating:id="BlueTriangle">
  <gating:dimension gating:compensation-ref="FCS"
    gating:transformation-ref="Bound_Logicle" >
    <data-type:fcs-dimension data-type:name="FITC-A"/>
  </gating:dimension>
  <gating:dimension gating:compensation-ref="FCS"
    gating:transformation-ref="Bound_Logicle">
    <data-type:fcs-dimension data-type:name="PE-A"/>
  </gating:dimension>
  <gating:vertex>
    <gating:coordinate data-type:value="0.4"/>
    <gating:coordinate data-type:value="0"/>
  </gating:vertex>
  <gating:vertex>
    <gating:coordinate data-type:value="0"/>
    <gating:coordinate data-type:value="0.4"/>
  </gating:vertex>
  <gating:vertex>
    <gating:coordinate data-type:value="1"/>
    <gating:coordinate data-type:value="1"/>
  </gating:vertex>
</gating:PolygonGate>

```

Example 28: Polygon gate referencing a transformation with a boundary.

This gate is shown in Figure 23. On the left in panel A, the blue triangle demonstrates the polygon gate as defined in example 28, and the red square demonstrates the boundary that is part of the scaling transformation defined in example 27. The effect of the transformation boundary is shown on a few example events that are positioned using a corresponding Logicle scale, but without considering the boundary. Red crosses indicate events outside of the polygon gate, and green crosses events in the gate. Noticeably, there are several green crosses placed outside of the blue triangle. These events are outside of the defined boundary. Once the boundary is applied, these events are *pulled* towards the red square, which brings them in the blue triangle and explains why they are considered to be in the polygon gate. Panel B on the right demonstrates an alternative view of the boundary effect, which can be imagined as extending the gate to $\pm\infty$ in places where it crosses the transformation boundary.

- (b) Example 29 demonstrates the definition of an inverse hyperbolic sine transformation (see section 6.4) with a left boundary restricting its result to the $[-0.1, \infty]$ interval.

```

<transforms:transformation transforms:id="Left_Bound_Arcsinh_262144_4.5_0"
  transforms:boundMin="-0.1">
  <transforms:fasinh transforms:T="262144" transforms:M="4.5" transforms:A="0" />
</transforms:transformation>

```

Example 29: Inverse hyperbolic sine transformation with a left boundary.

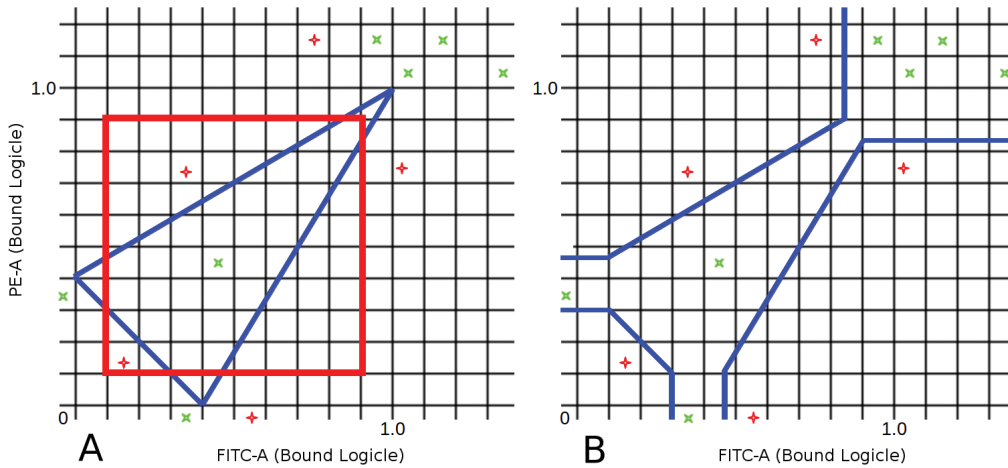


Figure 23: The effect of transformation boundary as described section 6.1.5 (a).

(c) Example 30 demonstrates the definition of a parametrized ratio transformation (see section 8.1) with a boundary restricting its result to the $[0.2, 5]$ interval.

```
<transforms:transformation transforms:id="myBoundRatio"
    transforms:boundMin="0.2" transforms:boundMax="5">
  <transforms:fratio transforms:A="1" transforms:B="0" transforms:C="0">
    <data-type:fcs-dimension data-type:name="FL1-A" />
    <data-type:fcs-dimension data-type:name="FL2-A" />
  </transforms:fratio>
</transforms:transformation>
```

Example 30: Ratio transformation with a boundary restricting its result to the $[0.2, 5]$ interval.

6.2 Parametrized linear transformation – *flin*

6.2.1 Definition

Parametrized linear transformation (*flin*) is defined by the following function:

$$flin(x, T, A) = \frac{x + A}{T + A}$$

where

- $x \in \mathbb{R}$ is the value that is being transformed (an FCS dimension value).
- $T \in \mathbb{R}, T > 0$ is the *top of scale* value, such as 262144 (*i.e.*, 2^{18}) as commonly the case.
- $A \in \mathbb{R}, 0 \leq A \leq T$ is a constant determining the bottom end of the transformation.

This transformation provides a linear display that maps scale values from the $[-A, T]$ interval to the $[0, 1]$ interval. However, it is defined for all $x \in \mathbb{R}$ including outside of the $[-A, T]$ interval.

6.2.2 Syntax specification

Parametrized linear transformation shall be specified by the *flin* element as follows:

- (a) The *flin* element shall be placed as a sub element of the *transformation* element; same as with any other scale transformation, the *transformation* element shall use the *id* attribute to assign a unique identifier to this transformation so that it can be referenced from a gate description, and it may use the *boundMin* and *boundMax* attributes in order to specify a boundary as described in section 6.1.
- (b) The *flin* element shall contain the *T* attribute specifying the “top scale” value as a positive number.
- (c) The *flin* element shall contain the *A* attribute specifying the *A* constant as non-negative number; the value of *A* shall be less than the value of *T*.
- (d) A *custom_info* element may be placed in the *flin* element. The *custom_info* element may contain any additional custom information related to this transformation; the format of this element (*i.e.*, its attributes, sub elements, etc.) is not constrained by this specification as long as the document remains a valid XML.

6.2.3 Validity conditions

A linear transformation definition is valid if and only if:

- (a) It is valid according to the XML schema.
- (b) The value of the *A* attribute is less or equal to the value of the *T* attribute, *i.e.*, $A \leq T$.

6.2.4 Examples

- (a) Example 31 demonstrates the definition of a parametrized linear transformation with $T = 1000$ and $A = 0$. A few example values before and after the transformation are listed in Table 5.

```
<transforms:transformation transforms:id="myLin">
  <transforms:flin transforms:T="1000" transforms:A="0" />
</transforms:transformation>
```

Example 31: Parametrized linear transformation definition with $T = 1000$ and $A = 0$.

- (b) Example 32 demonstrates the definition of a parametrized linear transformation with $T = 1000$ and $A = 100$. A few example values before and after the transformation are listed in Table 5.
- (c) Example 33 demonstrates the definition of a parametrized linear transformation with $T = 1024$ and $A = 256$. A few example values before and after the transformation are listed in Table 5.

x	$\mathit{flin}(x, 1000, 0)$	$\mathit{flin}(x, 1000, 100)$	$\mathit{flin}(x, 1024, 256)$	$\mathit{bound}(\mathit{flin}(x, 1000, 0), 0, 0.8)$
-100	-0.1	0.0	0.121875	0
-10	-0.01	≈ 0.081818	0.1921875	0
0	0.0	≈ 0.090909	0.2	0
10	0.01	0.1	0.2078125	0.01
100	0.1	≈ 0.181818	0.278125	0.1
120	0.12	0.2	0.29375	0.12
890	0.89	0.9	0.8953125	0.8
1000	1.0	1.0	0.98125	0.8

Table 5: Sample values of the flin transformation with various T and A .

```
<transforms:transformation transforms:id="myLin2">
  <transforms:flin transforms:T="1000" transforms:A="100" />
</transforms:transformation>
```

Example 32: Parametrized linear transformation definition with $T = 1000$ and $A = 100$.

```
<transforms:transformation transforms:id="myLin3">
  <transforms:flin transforms:T="1024" transforms:A="256" />
</transforms:transformation>
```

Example 33: Parametrized linear transformation definition with $T = 1024$ and $A = 256$.

```
<transforms:transformation transforms:id="myBoundLin3"
  transforms:boundMin="0" transforms:boundMax="0.8">
  <transforms:flin transforms:T="1000" transforms:A="0" />
</transforms:transformation>
```

Example 34: Parametrized linear transformation definition with a boundary.

(d) Example 34 demonstrates the definition of the same linear transformation as in example 31; however, a boundary is used to restrict the results of this transformation to the $[0, 0.8]$ interval. A few example values before and after the transformation are listed in Table 5.

6.3 Parametrized logarithmic transformation – flog

6.3.1 Definition

Parametrized logarithmic transformation (flog) is defined by the following function:

$$\mathit{flog}(x, T, M) = \frac{1}{M} \log_{10}\left(\frac{x}{T}\right) + 1$$

where

- \log_{10} is the common logarithm (*i.e.*, logarithm with base 10).
- $x \in \mathbb{R}, x > 0$ is the value that is being transformed (an FCS dimension value). Typically, $x \leq T$ although the transformation function is also defined for $x > T$.

- $T \in \mathbb{R}, T > 0$ is the *top of scale* value, such as 262144 (*i.e.*, 2^{18}) as commonly the case.
- $M \in \mathbb{R}, M > 0$ is the desired number of logarithmic *decades*.

This transformation provides a logarithmic display that maps scale values from the $(0, T]$ interval to the $(-\infty, 1]$ interval such that the data value T is mapped to 1 and M decades of data are mapped onto the unit interval. Also, $\lim_{x \rightarrow 0}(flog(x, T, M)) = -\infty$.

6.3.2 Syntax specification

Parametrized logarithmic transformation shall be specified by the *flog* element as follows:

- The *flog* element shall be placed as a sub element of the *transformation* element; same as with any other scale transformation, the *transformation* element shall use the *id* attribute to assign a unique identifier to this transformation so that it can be referenced from a gate description, and it may use the *boundMin* and *boundMax* attributes in order to specify a boundary as described in section 6.1.
- The *flog* element shall contain the T attribute specifying the “top scale” value as a positive number.
- The *flog* element shall contain the M attribute specifying the desired “number of decades” as a positive number.
- A *custom_info* element may be placed in the *flog* element. The *custom_info* element may contain any additional custom information related to this transformation; the format of this element (*i.e.*, its attributes, sub elements, etc.) is not constrained by this specification as long as the document remains a valid XML.

6.3.3 Validity conditions

A logarithmic transformation definition is valid if and only if it is valid according to the XML schema. All validity constrains, including $T > 0$ and $M > 0$, are part of the XML schema validation.

6.3.4 Examples

- Example 35 demonstrates the definition of a parametrized logarithmic transformation with $T = 10000$ and $M = 5$. A few example values before and after the transformation are listed in Table 6.

```
<transforms:transformation transforms:id="myLog">
  <transforms:flog transforms:T="10000" transforms:M="5" />
</transforms:transformation>
```

Example 35: Parametrized logarithmic transformation definition with $T = 10000$ and $M = 5$.

- Example 36 demonstrates the definition of a parametrized logarithmic transformation with $T = 1023$ and $M = 4.5$. A few example values before and after the transformation are listed in Table 6.

x	$flog(x, 10000, 5)$	$flog(x, 1023, 4.5)$	$flog(x, 262144, 4.5)$	$bound(flog(x, 10000, 5), 0, \infty)$
-1	ND*	ND*	ND*	ND*
0	ND*, $\lim_{x \rightarrow 0} = -\infty$	ND*, $\lim_{x \rightarrow 0} = -\infty$	ND*, $\lim_{x \rightarrow 0} = -\infty$	ND*, $\lim_{x \rightarrow 0} = 0$
0.5	≈ 0.139794	≈ 0.264243	≈ -0.271016	≈ 0.139794
1	0.2	≈ 0.331139	≈ -0.204120	0.2
10	0.4	≈ 0.553361	≈ 0.018102	0.4
100	0.6	≈ 0.775583	≈ 0.240324	0.6
1000	0.8	≈ 0.997805	≈ 0.462547	0.8
1023	≈ 0.801975	1.0	≈ 0.464741	≈ 0.801975
10000	1.0	≈ 1.220028	≈ 0.684768	1.0
100000	1.2	≈ 1.442250	≈ 0.906991	1.2
262144	≈ 1.283708	≈ 1.535259	1.0	≈ 1.283708

*ND stands for “Not Defined”.

Table 6: Sample values of the $flog$ transformation with various T and M .

```
<transforms:transformation transforms:id="myLog2">
  <transforms:flog transforms:T="1023" transforms:M="4.5" />
</transforms:transformation>
```

Example 36: Parametrized logarithmic transformation definition with $T = 1023$ and $M = 4.5$.

- (c) Example 37 demonstrates the definition of a parametrized logarithmic transformation with $T = 262144$ (*i.e.*, 2^{18}) and $M = 4.5$. A few example values before and after the transformation are listed in Table 6.

```
<transforms:transformation transforms:id="myLog3">
  <transforms:flog transforms:T="262144" transforms:M="4.5" />
</transforms:transformation>
```

Example 37: Parametrized logarithmic transformation definition with $T = 262144$ and $M = 4.5$.

- (d) Example 38 demonstrates the definition of the same parametrized logarithmic transformation as in example 35; however, a single-side boundary is used to restrict the results of this transformation to the $[0, \infty]$ interval. A few example values before and after the transformation are listed in Table 6.

```
<transforms:transformation transforms:id="myBoundLog"
  transforms:boundMin="0">
  <transforms:flog transforms:T="10000" transforms:M="5" />
</transforms:transformation>
```

Example 38: Parametrized logarithmic transformation definition with a boundary.

6.4 Parametrized inverse hyperbolic sine transformation – *fasinh*

6.4.1 Definition

Parametrized inverse hyperbolic sine transformation (*fasinh*) is defined by the following function:

$$fasinh(x, T, M, A) = \frac{\operatorname{asinh}(x \sinh(M \ln(10))/T) + A \ln(10)}{(M + A) \ln(10)}$$

where

- asinh is the inverse hyperbolic sine function.
- \sinh is the hyperbolic sine function.
- \ln is the natural logarithm (*i.e.*, logarithm with base $e \approx 2.718281828$).
- $x \in \mathbb{R}, x > 0$ is the value that is being transformed (an FCS dimension value). Typically, $x \leq T$ although the transformation function is also defined for $x > T$.
- $T \in \mathbb{R}, T > 0$ is the *top of scale* value, such as 262144 (*i.e.*, 2^{18}) as commonly the case.
- $M \in \mathbb{R}, M > 0$ is the desired number of *decades*.
- $A \in \mathbb{R}, 0 \leq A \leq M$ is the number of additional *negative decades* that will be “brought to scale”.

This transformation is equivalent to *Logicle*($T, 0, M, A$) (see section 6.5). It provides an inverse hyperbolic sine transformation that maps a data value onto the interval $[0,1]$ such that:

- The top of scale value (*i.e.*, T) is mapped to 1.
- Large data values are mapped to locations similar to an $(M + A)$ -*decade* logarithmic scale.
- A decades of negative data are brought on scale.

6.4.2 Syntax specification

Parametrized inverse hyperbolic sine transformation shall be specified by the *fasinh* element as follows:

- (a) The *fasinh* element shall be placed as a sub element of the *transformation* element; same as with any other scale transformation, the *transformation* element shall use the *id* attribute to assign a unique identifier to this transformation so that it can be referenced from a gate description, and it may use the *boundMin* and *boundMax* attributes in order to specify a boundary as described in section 6.1.
- (b) The *fasinh* element shall contain the T attribute specifying the “top scale” value as a positive number.
- (c) The *fasinh* element shall contain the M attribute specifying the desired “number of decades” as a positive number.

- (d) The *fasinh* element shall contain the *A* attribute specifying the desired “number of additional negative decades” as a non-negative number.
- (e) A *custom_info* element may be placed in the *fasinh* element. The *custom_info* element may contain any additional custom information related to this transformation; the format of this element (*i.e.*, its attributes, sub elements, etc.) is not constrained by this specification as long as the document remains a valid XML.

6.4.3 Validity conditions

A parametrized inverse hyperbolic sine transformation definition is valid if and only if:

- (a) It is valid according to the XML schema (validity constrains, such as $T > 0$, $M > 0$ and $A \geq 0$, are part of the XML schema validation).
- (b) The value of the *A* attribute shall be less or equal to the value of the *M* attribute (*i.e.*, $A \leq M$).

6.4.4 Examples

- (a) Example 39 demonstrates the definition of a parametrized inverse hyperbolic sine transformation with $T = 1000$, $M = 4$ and $A = 1$. A few example values before and after the transformation are listed in Table 7.

```
<transforms:transformation transforms:id="myASinH">
  <transforms:fasinh transforms:T="1000" transforms:M="4" transforms:A="1" />
</transforms:transformation>
```

Example 39: Inverse hyperbolic sine transformation with $T = 1000$, $M = 4$ and $A = 1$.

x	$fasinh(x,1000,4,1)$	$fasinh(x,1000,5,0)$	$fasinh(x,1000,3,2)$
-10	≈ -0.200009	≈ -0.6	≈ 0.199144
-5	≈ -0.139829	≈ -0.539794	≈ 0.256923
-1	≈ -0.000856	≈ -0.400009	≈ 0.358203
0	0.2	0	0.4
0.3	≈ 0.303776	≈ 0.295521	≈ 0.412980
1	≈ 0.400856	≈ 0.400009	≈ 0.441797
3	≈ 0.495521	≈ 0.495425	≈ 0.503776
10	≈ 0.600009	≈ 0.6	≈ 0.600856
100	≈ 0.8	≈ 0.8	≈ 0.800009
1000	1	1	1

Table 7: Sample values of the *fasinh* transformation with various T , M and A .

- (b) Example 40 demonstrates the definition of a parametrized inverse hyperbolic sine transformation with $T = 1000$, $M = 5$ and $A = 0$. A few example values before and after the transformation are listed in Table 7.

```
<transforms:transformation transforms:id="myASinH2">
  <transforms:fasinh transforms:T="1000" transforms:M="5" transforms:A="0" />
</transforms:transformation>
```

Example 40: Inverse hyperbolic sine transformation with $T = 1000$, $M = 5$ and $A = 0$.

- (c) Example 41 demonstrates the definition of a parametrized inverse hyperbolic sine transformation with $T = 1000$, $M = 3$ and $A = 2$. A few example values before and after the transformation are listed in Table 7.

```
<transforms:transformation transforms:id="myASinH3">
  <transforms:fasinh transforms:T="1000" transforms:M="3" transforms:A="2" />
</transforms:transformation>
```

Example 41: Inverse hyperbolic sine transformation with $T = 1000$, $M = 3$ and $A = 2$.

- (d) Example 29 in section 6.1.5 (b) demonstrates the definition of an inverse hyperbolic sine transformation with a boundary.

6.5 Logicle transformation

6.5.1 Definition

Logicle transformation as published by Moore and Parks [22] is defined by the following function:

$$\text{logicle}(x, T, W, M, A) = \text{root}(B(y, T, W, M, A) - x)$$

and B is a modified biexponential function

$$B(y, T, W, M, A) = ae^{by} - ce^{-dy} - f$$

where

- $x \in \mathbb{R}$ is the value that is being transformed (an FCS dimension value). Typically, $x \leq T$ although the transformation function is also defined for $x > T$.
- $y \in \mathbb{R}$ is the result of the transformation.
- $T \in \mathbb{R}, T > 0$ is the *top of scale* value.
- $M \in \mathbb{R}, M > 0$ is the number of *decades* that the true logarithmic scale approached at the high end of the Logicle scale would cover in the plot range.
- $W \in \mathbb{R}, 0 \leq W \leq M/2$ is the number of such decades in the approximately linear region. The choice of $W = M/2$ specifies a scale that is essentially linear over the whole range except for a small region of large data values. For situations in which values of W approaching $M/2$ might be chosen, ordinary linear display scales will usually be more appropriate. The choice of $W = 0$ gives the hyperbolic sine function.

- $A \in \mathbb{R}$, $-W \leq A \leq M - 2W$ is the number of additional decades of negative data values to be included.
- *root* is a standard root finding algorithm (*e.g.*, Newton's method) that finds y such as $B(y, T, W, M, A) = x$.

and a , b , c , d and f are defined by means of T , W , M , A , w , x_0 , x_1 , x_2 , c_a and f_a as

$$\begin{aligned} w &= \frac{W}{M + A} \\ x_2 &= \frac{A}{M + A} \\ x_1 &= x_2 + w \\ x_0 &= x_2 + 2w \\ b &= (M + A) \ln(10) \end{aligned}$$

d is a constant so that

$$2(\ln(d) - \ln(b)) + w(d + b) = 0$$

given b and w
and

$$\begin{aligned} c_a &= e^{x_0(b+d)} \\ f_a &= e^{bx_1} - \frac{c_a}{e^{dx_1}} \\ a &= \frac{T}{e^b - f_a - \frac{c_a}{e^d}} \\ c &= c_a a \\ f &= f_a a \end{aligned}$$

The Logicle scale is the inverse of a modified biexponential function. It provides a Logicle display that maps scale values onto the $[0, 1]$ interval such that the data value T is mapped to 1, large data values are mapped to locations similar to an $(M + A)$ -decade logarithmic scale, and A decades of negative data are brought on scale. For implementation purposes, it is recommended to follow guidance in [22] and see the reference implementation rather than encoding the transformation directly based on the presented formulas since a naïve implementation will likely suffer from significant round off errors.

The standard Logicle scale (*i.e.*, with $A = 0$) is defined, so that the most negative value on scale has the same absolute value as the most positive data value in what can be considered the quasilinear region of the scale. The zero data value is at the center of the quasilinear region, and versions differing only in the value of W approach the same logarithmic behavior at the high end. It has the desirable feature of making the scale position of large data values essentially the same among Logicle displays with different values of W and closely matching a true logarithmic display with the same value for M , which facilitates visual comparison of distributions.

In a few situations, altering the negative data range to be greater or less than the nominal quasilinear region of a standard Logicle transformation may be desirable. This may be accomplished by altering the A parameter that specifies additional decades of negative data values.

Positive values of A can be used to extend the range of negative data values on scale, however, the added scale range will not be quasilinear and can lead to spurious data peaks like those seen in logarithmic displays. If there is simply a need to display more negative data range, increasing W will accomplish this while maintaining quasilinearity for all on-scale negative values. For data that cannot include negative values, it may be advantageous to set $A = -W$ producing a display with no negative range but with zero on scale and with near-linear behavior near zero. Nonzero values of A will change the display scale of large data values whose consistency is one of the desirable features of the standard Logicle method. However, the parameter A should not be a routinely adjusted user parameter but should instead be reserved for these special cases. In general, the width parameter W is the only one that should be routinely varied, and the best approach in general is to set it with respect to the most negative relevant values in the data as described in the original Logicle manuscript [23]. Figure 24 demonstrates the effect of the increase of parameter W from 0 to 2.

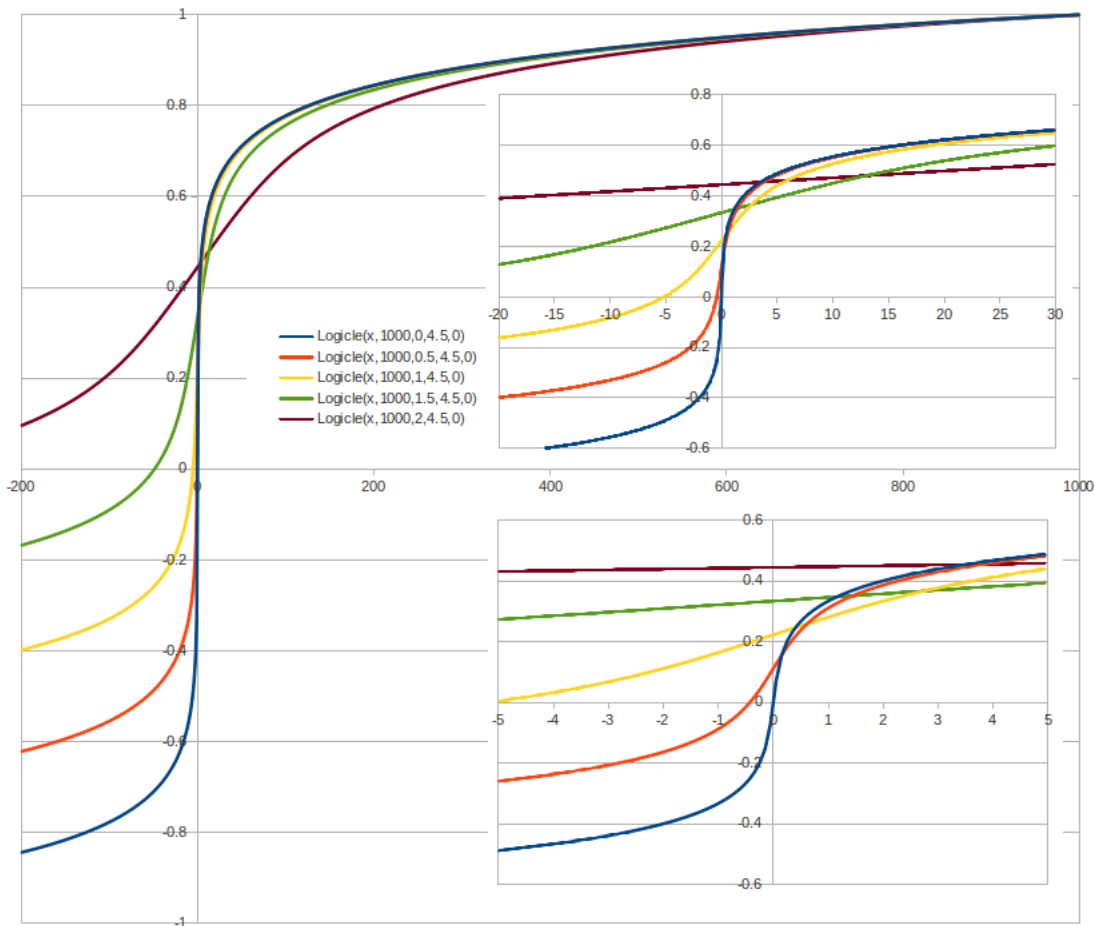


Figure 24: The effect of increasing the W parameter in the Logicle transformation.

6.5.2 Syntax specification

Logicle transformation shall be specified by the *logicle* element as follows:

- (a) The *logicle* element shall be placed as a sub element of the *transformation* element; same as with any other scale transformation, the *transformation* element shall use the *id* attribute to assign a unique identifier to this transformation so that it can be referenced from a gate description, and it may use the *boundMin* and *boundMax* attributes in order to specify a boundary as described in section 6.1.
- (b) The *logicle* element shall contain the *T* attribute specifying the “top scale” value as a positive number.
- (c) The *logicle* element shall contain the *M* attribute specifying the desired “number of decades” as a positive number.
- (d) The *logicle* element shall contain the *W* attribute specifying the number of such decades in the approximately linear region. The value of the *W* attribute shall be greater than or equal to zero and less than or equal to the value of the *M* attribute divided by 2 (*i.e.*, $0 \leq W \leq M/2$).
- (e) The *logicle* element shall contain the *A* attribute specifying the desired “number of additional negative decades”. The value of the *A* attribute shall be greater than or equal to minus of the value of the *W* attribute and less than or equal to the value of the *M* attribute minus twice the value of the *W* attribute (*i.e.*, $-W \leq A \leq M - 2W$).
- (f) A *custom_info* element may be placed in the *logicle* element. The *custom_info* element may contain any additional custom information related to this transformation; the format of this element (*i.e.*, its attributes, sub elements, etc.) is not constrained by this specification as long as the document remains a valid XML.

6.5.3 Validity conditions

A Logicle transformation definition is valid if and only if:

- (a) It is valid according to the XML schema.
- (b) The value of the *W* attribute shall be greater than or equal to zero and less than or equal to the value of the *M* attribute divided by 2 (*i.e.*, $0 \leq W \leq M/2$).
- (c) The value of the *A* attribute shall be greater than or equal to minus of the value of the *W* attribute and less than or equal to the value of the *M* attribute minus twice the value of the *W* attribute (*i.e.*, $-W \leq A \leq M - 2W$).

Constraints on the *T*, *W*, *M* and *A* attribute values are the same as with the Hyperlog transformation except for the value of *W*, which shall be greater of equal to 0 for Logicle, while it shall be greater than 0 for Hyperlog (see section 6.6.3).

6.5.4 Examples

- (a) Example 42 demonstrates the definition of a Logicle transformation with $T = 1000$, $W = 1$, $M = 4$, and $A = 0$. A few example values before and after the transformation are listed in Table 8.

```
<transforms:transformation transforms:id="myLogicle1">
  <transforms:logicle transforms:T="1000" transforms:W="1"
    transforms:M="4" transforms:A="0" />
</transforms:transformation>
```

Example 42: Logicle transformation definition with $T = 1000$, $W = 1$, $M = 4$ and $A = 0$.

x	$Logicle(x, 1000, 1, 4, 0)$	$Logicle(x, 1000, 1, 4, 1)$	$Logicle(x, 1000, 0, 4, 1)$
-10	≈ 0.067574	≈ 0.254059	≈ -0.200009
-5	≈ 0.147986	≈ 0.318389	≈ -0.139829
-1	≈ 0.228752	≈ 0.383001	≈ -0.000856
0	0.25	0.4	0.2
0.3	≈ 0.256384	≈ 0.405107	≈ 0.303776
1	≈ 0.271248	≈ 0.416999	≈ 0.400856
3	≈ 0.312897	≈ 0.450318	≈ 0.495521
10	≈ 0.432426	≈ 0.545941	≈ 0.600009
100	≈ 0.739548	≈ 0.791638	≈ 0.8
1000	1	1	1

Table 8: Sample values of the *Logicle* transformation with various T , W , M , and A .

- (b) Example 43 demonstrates the definition of a Logicle transformation with $T = 1000$, $W = 1$, $M = 4$, and $A = 1$. A few example values before and after the transformation are listed in Table 8.

```
<transforms:transformation transforms:id="myLogicle2">
  <transforms:logicle transforms:T="1000" transforms:W="1"
    transforms:M="4" transforms:A="1" />
</transforms:transformation>
```

Example 43: Logicle transformation definition with $T = 1000$, $W = 1$, $M = 4$, and $A = 1$.

- (c) Example 44 demonstrates the definition of a Logicle transformation with $T = 1000$, $W = 0$, $M = 4$, and $A = 1$. Note that any Logicle transformation with W set to 0 equals to the parametrized inverse hyperbolic sine transformation with the same values of T , M and A . In this example, $Logicle(x, 1000, 0, 4, 1) = f_{asinh}(x, 1000, 4, 1)$. A few example values before and after the Logicle transformation are listed in Table 8.

```
<transforms:transformation transforms:id="myLogicle3">
  <transforms:logicle transforms:T="1000" transforms:W="0"
    transforms:M="4" transforms:A="1" />
</transforms:transformation>
```

Example 44: Logicle transformation definition with $T = 1000$, $W = 0$, $M = 4$, and $A = 1$.

- (d) Example 27 in section 6.1.5 (a) demonstrates the definition of a Logicle transformation with a boundary.

6.6 Hyperlog transformation

6.6.1 Definition

Hyperlog transformation, originally published by Bagwell [24], has been parametrized to match the other log-like transformations supported by Gating-ML. Hyperlog is defined by the following function:

$$\text{hyperlog}(x, T, W, M, A) = \text{root}(EH(y, T, W, M, A) - x)$$

and

$$EH(y, T, W, M, A) = ae^{by} + cy - f$$

where

- $x \in \mathbb{R}$ is the value that is being transformed (an FCS dimension value). Typically, $x \leq T$ although the transformation function is also defined for $x > T$.
- $y \in \mathbb{R}$ is the result of the transformation.
- $T \in \mathbb{R}, T > 0$ is the *top of scale* value.
- $M \in \mathbb{R}, M > 0$ is the number of *decades* that the true logarithmic scale approached at the high end of the Hyperlog scale would cover in the plot range.
- $W \in \mathbb{R}, 0 < W \leq M/2$ is the number of such decades in the approximately linear region.
- $A \in \mathbb{R}, -W \leq A \leq M - 2W$ is the number of additional decades of negative data values to be included.

and a, b, c and f are defined by means of $T, W, M, A, w, x_0, x_1, x_2, e_0, c_a$ and f_a as

$$\begin{aligned} w &= \frac{W}{M + A} \\ x_2 &= \frac{A}{M + A} \\ x_1 &= x_2 + w \\ x_0 &= x_2 + 2w \\ b &= (M + A) \ln(10) \\ e_0 &= e^{bx_0} \\ c_a &= \frac{e_0}{w} \\ f_a &= e^{bx_1} + c_a x_1 \\ a &= \frac{T}{e^b + c_a - f_a} \\ c &= c_a a \\ f &= f_a a \end{aligned}$$

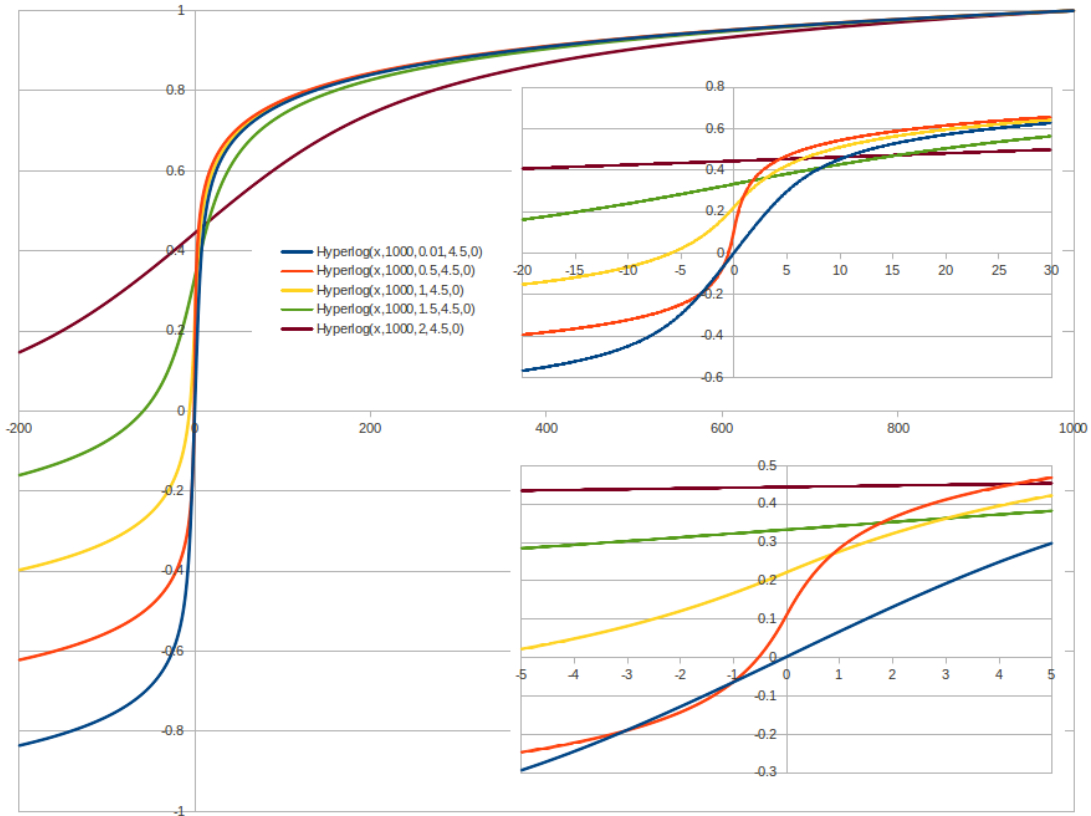


Figure 25: The effect of increasing the W parameter in the Hyperlog transformation.

As with the Logicle scale, generally only the width parameter W (*i.e.*, the number of decades in the approximately linear region) should be routinely varied and may also be set as described in the original Logicle manuscript [23]. Figure 25 demonstrates the effect of the increase of parameter W from 0.01 to 2.

For implementation purposes, it is highly recommended to follow the reference implementation rather than encoding the transformation directly based on the presented formulas since a naïve implementation will likely suffer from significant round off errors.

6.6.2 Syntax specification

Hyperlog transformation shall be specified by the *hyperlog* element as follows:

- (a) The *hyperlog* element shall be placed as a sub element of the *transformation* element; same as with any other scale transformation, the *transformation* element shall use the *id* attribute to assign a unique identifier to this transformation so that it can be referenced from a gate description, and it may use the *boundMin* and *boundMax* attributes in order to specify a boundary as described in section 6.1.
- (b) The *hyperlog* element shall contain the T attribute specifying the “top scale” value as a positive number.
- (c) The *hyperlog* element shall contain the M attribute specifying the desired “number of decades” as a positive number.

- (d) The *hyperlog* element shall contain the W attribute specifying the number of such decades in the approximately linear region. The value of the W attribute shall be greater than zero and less than or equal to the value of the M attribute divided by 2 (*i.e.*, $0 < W \leq M/2$).
- (e) The *hyperlog* element shall contain the A attribute specifying the desired “number of additional negative decades”. The value of the A attribute shall be greater than or equal to minus of the value of the W attribute and less than or equal to the value of the M attribute minus twice the value of the W attribute (*i.e.*, $-W \leq A \leq M - 2W$).
- (f) A *custom_info* element may be placed in the *hyperlog* element. The *custom_info* element may contain any additional custom information related to this transformation; the format of this element (*i.e.*, its attributes, sub elements, etc.) is not constrained by this specification as long as the document remains a valid XML.

6.6.3 Validity conditions

A Hyperlog transformation definition is valid if and only if:

- (a) It is valid according to the XML schema.
- (b) The value of the W attribute shall be greater than or equal to zero and less than or equal to the value of the M attribute divided by 2 (*i.e.*, $0 < W \leq M/2$).
- (c) The value of the A attribute shall be greater than or equal to minus of the value of the W attribute and less than or equal to the value of the M attribute minus twice the value of the W attribute (*i.e.*, $-W \leq A \leq M - 2W$).

Constraints on the T , W , M , and A attribute values are the same as with the Logicle transformation except for the value of W , which shall be greater than 0 for Hyperlog, while it may be greater or equal to 0 for Logicle (see section 6.5.3).

6.6.4 Examples

- (a) Example 45 demonstrates the definition of a Hyperlog transformation with $T = 1000$, $W = 1$, $M = 4$, and $A = 0$. A few example values before and after the transformation are listed in Table 9.

```
<transforms:transformation transforms:id="myHyperlog1">
  <transforms:hyperlog transforms:T="1000" transforms:W="1"
    transforms:M="4" transforms:A="0" />
</transforms:transformation>
```

Example 45: Hyperlog transformation definition with $T = 1000$, $W = 1$, $M = 4$ and $A = 0$.

- (b) Example 46 demonstrates the definition of a Hyperlog transformation with $T = 1000$, $W = 1$, $M = 4$, and $A = 1$. A few example values before and after the transformation are listed in Table 9.

x	$\text{Hyperlog}(x, 1000, 1, 4, 0)$	$\text{Hyperlog}(x, 1000, 1, 4, 1)$	$\text{Hyperlog}(x, 1000, 0.01, 4, 1)$
-10	≈ 0.083554	≈ 0.266843	≈ 0.017447
-5	≈ 0.155868	≈ 0.324695	≈ 0.106439
-1	≈ 0.229477	≈ 0.383581	≈ 0.182593
0	0.25	0.4	0.202
0.3	≈ 0.256239	≈ 0.404991	≈ 0.207833
1	≈ 0.270523	≈ 0.416419	≈ 0.221407
3	≈ 0.309091	≈ 0.447273	≈ 0.259838
10	≈ 0.416446	≈ 0.533157	≈ 0.386553
100	≈ 0.731875	≈ 0.7855	≈ 0.774211
1000	1	1	1

Table 9: Sample values of the *Hyperlog* transformation with various T , W , M , and A .

```
<transforms:transformation transforms:id="myHyperlog2">
  <transforms:hyperlog transforms:T="1000" transforms:W="1"
    transforms:M="4" transforms:A="1" />
</transforms:transformation>
```

Example 46: Hyperlog transformation definition with $T = 1000$, $W = 1$, $M = 4$, and $A = 1$.

```
<transforms:transformation transforms:id="myHyperlog3">
  <transforms:hyperlog transforms:T="1000" transforms:W="0"
    transforms:M="4" transforms:A="1" />
</transforms:transformation>
```

Example 47: Hyperlog transformation definition with $T = 1000$, $W = 0$, $M = 4$, and $A = 1$.

(c) Example 47 demonstrates the definition of a Hyperlog transformation with $T = 1000$, $W = 0$, $M = 4$, and $A = 1$. A few example values before and after the transformation are listed in Table 9.

(d) Example 48 demonstrates the definition of the same Hyperlog transformation as in example 45; however, a single-side boundary is used to restrict the results of this transformation to the $[-\infty, 0.9]$ interval.

```
<transforms:transformation transforms:id="myBoundHyperlog"
  transforms:boundMax="0.9">
  <transforms:hyperlog transforms:T="1000" transforms:W="1"
    transforms:M="4" transforms:A="0" />
</transforms:transformation>
```

Example 48: Hyperlog transformation definition with a boundary.

7 Compensation description

7.1 Definition

In flow cytometry, the emission spectral overlap of fluorescent labels makes it usually necessary to correct detected signals before using the values as a basis for other analyses. Fluorescence compensation [25] is the process by which the total detected signal is corrected for the spectral overlap to yield an estimate of the actual amount of each dye. This process is described by the standard compensation computing algorithm in section 7.6.1.

Recently, novel instruments have been introduced measuring the fluorescence spectrum with greater detail by many more detectors than dyes in the sample. The originally square spillover matrix became a non-square matrix describing the relative amount of fluorescence detected by each of the detectors for each of the dyes in the sample. The process of compensation turned hereby into the problem of spectral unmixing as described in Section 7.6.2.

7.2 Syntax specification

Fluorescence compensation shall be described by a spectrum matrix. The term “spectrum” has been introduced as applicable for both the traditional compensation as well as spectral unmixing. Here, we describe the syntax of this description. The semantics is further explained in section 7.3. A $n \times m$ spectrum matrix, $n \in \mathbb{N}, m \in \mathbb{N}, n \leq m$ (historically, $n = m$, see section 7.3) shall be described by the *spectrumMatrix* element placed at the top level in the Gating-ML file (i.e., as a sub element of the *Gating-ML* element) as follows:

- (a) The *spectrumMatrix* element shall use the *id* attribute to globally identify the spectrum (spillover) matrix by assigning it a unique identifier.
- (b) The *spectrumMatrix* element may use the Boolean *matrix-inverted-already* attribute to explicitly specify whether the matrix has been inverted already. The default value of this attribute is *false*, in which case the matrix is considered to be the spectrum (spillover) matrix. If the *matrix-inverted-already* is set to *true* then the matrix is considered to be the (pseudo)inverse of the spectrum (spillover) matrix.
- (c) A *custom_info* element may be placed at the beginning of the *spectrumMatrix* element. The *custom_info* element may contain any additional custom information related to this spectrum matrix; the format of this element (i.e., its attributes, sub elements, etc.) is not constrained by this specification as long as the document remains a valid XML.
- (d) A *fluorochromes* element shall be placed at the beginning of the *spectrumMatrix* element or right after the *custom_info* element if the *custom_info* element is used.
- (e) n *fcs-dimension* elements shall be placed as sub elements of the *fluorochromes* element.
- (f) Each *fcs-dimension* element shall include a *name* attribute. These names shall be unique within the *fluorochromes* element as well as the *detectors* elements, i.e., no fluorochrome name shall be equal to any detector name, or to any other fluorochrome name, and no detector name shall be equal to any fluorochrome name, or to any other detector name within the matrix definition. Names assigned to fluorochromes (rows) of the matrix shall

be used to reference resulting gating dimensions in gates that are also referencing this spectrum (spillover) matrix.

- (g) A *detectors* element shall be placed after the *fluorochromes* element.
- (h) m *fcs-dimension* elements shall be placed as sub elements of the *detectors* element.
- (i) Each *fcs-dimension* element shall include a *name* attribute to reference an FCS dimension as described in section 4.1. The names shall be unique within the *detectors* element as well as the *fluorochromes* element, *i.e.*, no detector name shall be equal to any fluorochrome name.
- (j) The spectrum shall be written in the row major order as follows:
 - (i) n *spectrum* elements shall follow as sub elements of the *spectrumMatrix* element after the *fcs-dimensions* element; each *spectrum* element corresponds to a row of the spectrum (spillover) matrix.
 - (ii) m *coefficient* elements shall be included as sub elements of each of the *spectrum* elements.
 - (iii) Each *coefficient* element shall specify a numeric spectrum/spillover coefficient value using the *value* attribute.

7.3 Semantic specification

Assuming the *matrix-inverted-already* is set to *false* (or the attribute is not present) then let M be a $n \times m$ spectrum (spillover) matrix ($n \in \mathbb{N}, m \in \mathbb{N}, n \leq m$). Let X_f be the i -th argument specified within the *fluorochromes* element. Let Y_d be the j -th argument specified within the *detectors* element. Let S be the i -th *spectrum* element within the definition of spectrum matrix M . Let C be the j -th *coefficient* element within the *spectrum* element S . Let V be the value of the value attribute of C . Example 49 demonstrates the XML representation of the spectrum matrix M as described.

For typical spillover case, the entry shown in example 49 means that V is the spillover coefficient from X_f to Y_d . The spillover coefficient from X_f to Y_d is the ratio of the amount of signal in the Y_d channel to the amount of signal in the “primary” X_f channel for particles carrying only the X_f dye. For spectral unmixing cases, the entry shown in example 49 means that V is the spectrum coefficient from X_f to Y_d . The spectrum coefficient from X_f to Y_d is the relative amount of signal on the Y_d detector for particles carrying only the X_f dye.

If the *matrix-inverted-already* is set to *true*, then the matrix is considered to be the (pseudo)inverse of the spectrum (spillover) matrix. Typical fluorescence compensation (or spectral unmixing) algorithms include the computation of the (pseudo)inverse of the spectrum (spillover) matrix. This calculation does not need to be performed if the matrix is (pseudo)inverted already.

7.4 Validity conditions

A spectrum matrix definition is valid if and only if:

- (a) It is valid according to the XML schema.

```

<transforms:spectrumMatrix transforms:id="M"
                          transforms:matrix-inverted-already="false">
  <transforms:fluorochromes>
    ...
    <!-- i-th fcs-dimension element within the fluorochromes element -->
    <data-type:fcs-dimension data-type:name="Xf" />
    ...
  </transforms:fluorochromes>
  <transforms:detectors>
    ...
    <!-- j-th fcs-dimension element within the detectors element -->
    <data-type:fcs-dimension data-type:name="Yd" />
    ...
  </transforms:detectors>
  ...
  <!-- i-th spectrum element within the spectrumMatrix element -->
  <transforms:spectrum>
    ...
    <!-- j-th coefficient element within the spectrum element -->
    <transforms:coefficient transforms:value="V" />
    ...
  </transforms:spectrum>
  ...
</transforms:spectrumMatrix>

```

Example 49: Definition of a spectrum (spillover) matrix.

- (b) All the listed names are unique; detector names shall reference list mode data parameters, fluorochrome names shall be used to name compensated (unmixed) dimensions.
- (c) The definition is a proper $n \times m$ matrix, $n \in \mathbb{N}$, $m \in \mathbb{N}$, $n \leq m$, *i.e.*, there are n *fcs_dimension* elements as sub elements of the *fluorochromes* element, m *fcs_dimension* elements as sub elements of the *detectors* element and n *spectrum* elements, each *spectrum* element including m *coefficient* elements.
- (d) The rows of matrix M (corresponding to the individual fluorochrome spectra) shall be linearly independent, *i.e.*, no fluorochrome spectra may occur twice, which is to say, the matrix M is non-singular (invertible) in the square case and that the pseudo inverse has full rank otherwise.

There are no additional validity conditions that would ensure a “normalized” form of the matrix. In conventional compensation, there is an independent detector for each channel. Therefore, the best practice is to adjust the voltages on the detectors to bring a reference intensity particle to standard operating conditions, which fixes the scale between some amount of dye and the measurements on the dedicated detector. The point of normalizing to unity on the diagonal on conventional compensation is that it leaves this scale unchanged on compensated data. There is no direct equivalent for spectrum based instruments. The normalization to sum of one means that all the unmixed data will be scaled proportional to the total signal on all

channels. If the voltages were constant, this would be proportional to total photoelectrons. This is arguably a good choice, but since the detectors in use do have finer voltage control and the dyes have different brightnesses, the scales will be quite different than with a conventional instrument. On the other hand, if a spectrum is obtained based on probes with known intensity, one could choose to normalize the rows so that the unmixed data were on that scale automatically, which wouldn't disturb the unmixing. If the instrument was in standardized conditions when the spectra were obtained, then normalizing to 1 on the maximum channel for the detector would give results most similar to a conventional instrument. In summary, this specification does not enforce further validity conditions on the matrix as there are potentially several sensible conventions that implementors may choose to follow.

7.5 Examples

(a) Example 50 demonstrates a simple 3×3 spillover matrix. The same matrix is shown in Table 10.

```
<transforms:spectrumMatrix transforms:id="M">
  <transforms:fluorochromes>
    <data-type:fcs-dimension data-type:name="FITC" />
    <data-type:fcs-dimension data-type:name="PerCP" />
    <data-type:fcs-dimension data-type:name="APC" />
  </transforms:fluorochromes>
  <transforms:detectors>
    <data-type:fcs-dimension data-type:name="FL1-H" />
    <data-type:fcs-dimension data-type:name="FL2-H" />
    <data-type:fcs-dimension data-type:name="FL3-H" />
  </transforms:detectors>
  <transforms:spectrum>
    <transforms:coefficient transforms:value="1" />
    <transforms:coefficient transforms:value="0.022" />
    <transforms:coefficient transforms:value="0.083" />
  </transforms:spectrum>
  <transforms:spectrum>
    <transforms:coefficient transforms:value="0.125" />
    <transforms:coefficient transforms:value="1" />
    <transforms:coefficient transforms:value="0.051" />
  </transforms:spectrum>
  <transforms:spectrum>
    <transforms:coefficient transforms:value="0.037" />
    <transforms:coefficient transforms:value="0.078" />
    <transforms:coefficient transforms:value="1" />
  </transforms:spectrum>
</transforms:spectrumMatrix>
```

Example 50: A simple 3×3 spillover matrix definition.

(b) Example 51 demonstrates a simple 2×3 spectrum matrix. The same matrix is shown in Table 11. Real examples are much larger and therefore it would not be practical to include

	FL1-H	FL2-H	FL3-H
FITC	1	0.022	0.083
PerCP	0.125	1	0.051
APC	0.037	0.078	1

Table 10: Example of a simple 3×3 spillover matrix.

these in the text of the specification.

```
<transforms:spectrumMatrix transforms:id="M2">
  <transforms:fluorochromes>
    <data-type:fcs-dimension data-type:name="FITC"/>
    <data-type:fcs-dimension data-type:name="PE"/>
  </transforms:fluorochromes>
  <transforms:detectors>
    <data-type:fcs-dimension data-type:name="FL1-A"/>
    <data-type:fcs-dimension data-type:name="FL2-A"/>
    <data-type:fcs-dimension data-type:name="FL3-A"/>
  </transforms:detectors>
  <transforms:spectrum>
    <transforms:coefficient transforms:value="0.78" />
    <transforms:coefficient transforms:value="0.13" />
    <transforms:coefficient transforms:value="0.22" />
  </transforms:spectrum>
  <transforms:spectrum>
    <transforms:coefficient transforms:value="0.05" />
    <transforms:coefficient transforms:value="0.57" />
    <transforms:coefficient transforms:value="0.89" />
  </transforms:spectrum>
</transforms:spectrumMatrix>
```

Example 51: A simple 2×3 spectrum matrix definition.

	FL1-A	FL2-A	FL3-A
FITC	0.78	0.13	0.22
PE	0.05	0.57	0.89

Table 11: Example of a simple 2×3 spectrum matrix.

7.6 Compensation computing algorithm

7.6.1 Standard compensation based on square spectrum (spillover) matrices

Standard fluorescence compensation may be performed as described in this section. However, the purpose of the presented algorithm is to clarify potentially ambiguous points in the specification rather than to standardize the compensation computing algorithm itself.

- (a) Allocate a square matrix S of the size $n \times n$, where n is the number of listed detectors.

- (b) Assign a total ordering on the n detectors so that the order corresponds to detectors order as listed in the *fcs-dimension* elements. For $i \in \{1, 2, \dots, n\}$, the i -th detector gets assigned the i -th row and the i -th column of the matrix S .
- (c) Assign values to fields of matrix S by reading and parsing the *spectrum* and *coefficient* elements in the row-major order, each *spectrum* element denoting a new row of the matrix, each *coefficient* element denoting a value for a field. Formally, assign $S[i, j] = V$ for each value V of the *value* attribute of the j -th *coefficient* element that is placed within the i -th *spectrum* element.
- (d) If the value of the *matrix-inverted-already* attribute is set to *false*, or the attribute is not present in the matrix definition, then compute the matrix S^{-1} as the inverse of the matrix S . If the value of the *matrix-inverted-already* attribute is set to *true*, then the matrix was inverted already. In that case, you already have the inverse and therefore, simply change the “label” of S to S^{-1} , which will be used for further calculations.
- (e) For each event e in the list more data file:
 - (i) Read the vector v of expression values of e . Include only values of detectors listed in matrix S and permute (rearrange) the expression values in v to match the order they are listed for S . Apply the channel-to-scale transformation on v (see section 3.3.4) as necessary for event values saved in an FCS file.
 - (ii) Calculate the vector of compensated values v_c of expression values in vector v as $v_c = v \times S^{-1}$, *i.e.*, consider v as row vector and multiply v by the inverse of the S ; the result v_c will also be a row vector. The vector v_c will contain the compensated FCS dimensions in the same order as are the uncompensated dimensions in v . Please note that there may be negative values resulting from a valid compensation.

7.6.2 Compensation based on non-square spectrum matrices

Even for compensation with non-square spectrum matrices, we assume that the detector measurements are a linear combination of the amounts of the various dyes present, and some errors or noise in the system. Therefore, we seek a linear combination of the measured values that is an estimator of the amount of each dye. We know what detector values to expect (relatively) from each dye from single stained controls. For n dyes (fluorochromes) and m detectors, we have an $n \times m$ spectrum matrix and we seek an $n \times m$ unmixing matrix. In order to compensate, we will multiply each m vector of measurement values by the unmixing matrix and get an n vector of dye estimates.

When $n = m$, there is only one such unmixing matrix for a given spectrum matrix, and the unmixing process equals to an ordinary compensation. When $m > n$, the system is overdetermined and there typically isn’t an exact solution. Therefore, we ask for the one that is the “best”, *e.g.*, in the sense of minimizing the squares of residual errors. The optimal solution is found by some kind of optimization process.

The standard compensation computing algorithm described in Section 7.6.1 can be used for spectral unmixing (compensation based on non-square spectrum matrices) except for computing the matrix S^{-1} as the inversion of matrix S . Instead, a Moore–Penrose pseudoinverse [26] S^+ of the matrix S would be used. A computationally simple and accurate way to compute

the pseudoinverse S^+ is by using the singular value decomposition. If $S = U\Sigma V^T$ is the singular value decomposition of S , then $S^+ = V\Sigma^+U^T$. For a diagonal matrix such as Σ , we obtain the pseudoinverse Σ^+ by taking the reciprocal of each non-zero element on the diagonal, leaving the zeros in place, and transposing the resulting matrix. This method is known as the ordinary least squares (OLS) unmixing and would result in the optimal solution provided the noise followed the normal (Gaussian) distribution and the variance was constant and centered around zero. OLS or the nonnegative least squares (NNLS) [27] may be used as basic unmixing algorithms, however; since the optimal conditions may not be met in typical flow cytometry data, more sophisticated approaches have been developed in order to obtain more accurate estimates. Please note that the use of some of these sophisticated methods may produce different results and it may be subject matter covered by patent rights. Also, for the purpose of handling the unmixed data, note that in contrast to traditional compensation, which subtracts intensities and therefore returns smaller values, a properly implemented unmixing will return values describing the “reconstituted” signal with values potentially higher than any entry in the input (mixed) vector.

If proprietary spectral unmixing is used in any hardware or software, we recommend that the unmixed data be saved (exported) along with Gating-ML so that gates can directly reference and use the unmixed data as gating dimensions. The spectrum matrix may not have to be included in the Gating-ML file in these cases.

8 Additional transformations description

Currently, the parametrized ratio transformation is the only supported additional transformation. This transformation creates a new FCS dimension based on two existing FCS dimensions.

8.1 Parametrized ratio transformation – *fratio*

8.1.1 Definition

Parametrized ratio transformation (*fratio*) is defined by the following function:

$$fratio(x, y, A, B, C) = A \frac{x - B}{y - C}$$

where

- $x \in \mathbb{R}$, $y \in \mathbb{R}$, $y \neq C$ are two FCS dimensions.
- $A \in \mathbb{R}$, $B \in \mathbb{R}$, $C \in \mathbb{R}$ are *fratio* parameters.

8.1.2 Syntax specification

Parametrized ratio transformation shall be specified by the *fratio* element as follows:

- The *fratio* element shall be placed as a sub element of the *transformation* element; the *transformation* element shall use the *id* attribute to assign a unique identifier to this transformation so that it can be referenced from a *new-dimension* element of a gate description. In addition, the *transformation* element may use the *boundMin* and *boundMax* attributes in order to specify a boundary as described in section 6.1.

- (b) The *fratio* element shall contain the *A*, *B*, and *C* attributes specifying the *fratio* parameters.
- (c) A *custom_info* element may be placed in the *fratio* element. The *custom_info* element may contain any additional custom information related to this transformation; the format of this element (*i.e.*, its attributes, sub elements, etc.) is not constrained by this specification as long as the document remains a valid XML.
- (d) Two *fcs-dimension* elements shall follow the *custom_info* element (if the *custom_info* element is used), or they shall be placed as the only sub elements of the *fratio* element (if the *custom_info* element is not used).
- (e) Each *fcs-dimension* element shall contain a *name* attribute referencing an FCS dimension as specified in section 4.1.

8.1.3 Validity conditions

A parametrized ratio transformation definition is valid if and only if it is valid according to the XML schema. All validity constraints are part of the XML schema validation.

8.1.4 Examples

- (a) Example 52 demonstrates the definition of a ratio transformation applied on FCS dimensions FL1-A and FL2-A with $A = 1$, $B = 0$ and, $C = 0$. A few example values are listed in Table 12.

```
<transforms:transformation transforms:id="myRatio">
  <transforms:fratio transforms:A="1" transforms:B="0" transforms:C="0" />
  <data-type:fcs-dimension data-type:name="FL1-A" />
  <data-type:fcs-dimension data-type:name="FL2-A" />
</transforms:transformation>
```

Example 52: Parametrized ratio transformation with $A = 1$, $B = 0$, and $C = 0$.

x	y	$fratio(x,y,1,0,0)$	$fratio(x,y,10,5,5)$	$fratio(x,y,0.5,-10,25)$	$bound(fratio(x,y,1,0,0),0,5)$
-10	-5	2	15	0	2
-10	0	ND*	30	0	ND*
0	5	0	ND*	-0.25	0
0	0	ND*	10	-0.2	ND*
10	25	0.4	2.5	ND*	0.4
10	30	0.333333 ⁻	2	2	0.333333 ⁻
10	50	0.2	1.111111 ⁻	0.4	0.2
10	-25	-0.4	-1.666666 ⁻	-0.2	0
100	5	20	ND*	-2.75	5
100	50	2	21.111111 ⁻	2.2	2
768	50	15.56	169.555555 ⁻	15.36	5

*ND stands for "Not Defined".

Table 12: Sample values of the *fratio* transformation with various *A*, *B*, and *C*.

- (b) Example 53 demonstrates the definition of a ratio transformation applied on FCS dimensions FL1-H and FL1-W with $A = 10$, $B = 5$ and, $C = 5$. A few example values are listed in Table 12.

```
<transforms:transformation transforms:id="myRatio2">
  <transforms:fratio transforms:A="10" transforms:B="5" transforms:C="5" />
  <data-type:fcs-dimension data-type:name="FL1-H" />
  <data-type:fcs-dimension data-type:name="FL1-W" />
</transforms:transformation>
```

Example 53: Parametrized ratio transformation with $A = 10$, $B = 5$ and $C = 5$.

- (c) Example 54 demonstrates the definition of a ratio transformation applied on FCS dimensions FL3-A and FL4-A with $A = 0.5$, $B = -10$ and, $C = 25$. A few example values are listed in Table 12.

```
<transforms:transformation transforms:id="myRatio2" customAttributes="allowed">
  <transforms:fratio transforms:A="0.5" transforms:B="-10" transforms:C="25" />
  <data-type:custom_info>
    Custom info may be part of any transformation definition.
  </data-type:custom_info>
  <data-type:fcs-dimension data-type:name="FL3-A" />
  <data-type:fcs-dimension data-type:name="FL4-A" />
</transforms:transformation>
```

Example 54: Parametrized ratio transformation with $A = 0.5$, $B = -10$, and $C = 25$.

- (d) Example 30 in section 6.1.5 (c) demonstrates the definition of the same ratio transformation as in example 52; however, a boundary is used to restrict the results of this transformation to the $[0, 5]$ interval. A few example values are listed in Table 12.

Annex A References

- [1] Lee JA, Spidlen J, Boyce K, Cai J, Crosbie N, Dalphin M, Furlong J, Gasparetto M, Goldberg M, Goralczyk EM, Hyun B, Jansen K, Kollmann T, Kong M, Leif R, McWeeney S, Moloshok TD, Moore W, Nolan G, Nolan J, Nikolich-Zugich J, Parrish D, Purcell B, Qian Y, Selvaraj B, Smith C, Tchuvatkina O, Wertheimer A, Wilkinson P, Wilson C, Wood J, Zigon R, for Advancement of Cytometry Data Standards Task Force IS, Scheuermann RH, and Brinkman RR. MIFlowCyt: the Minimum Information about a Flow Cytometry Experiment. *Cytometry Part A*, 2008;73(10):926–930.
- [2] Spidlen J, Moore W, Parks D, Goldberg M, Bray C, Bierre P, Gorombey P, Hyun B, Hubbard M, Lange S, Lefebvre R, Leif R, Novo D, Ostruszka L, Treister A, Wood J, Murphy RF, Roederer M, Sudar D, Zigon R, and Brinkman RR. Data file standard for flow cytometry, version FCS 3.1. *Cytometry A*, 2010;77(1):97–100.
- [3] World Wide Web Consortium (W3C). Extensible Markup Language (XML) 1.0 (Fifth Edition), 2008. <http://www.w3.org/TR/2008/REC-xml-20081126/>.
- [4] World Wide Web Consortium (W3C). W3C Namespaces in XML 1.0 (Third Edition), 2009. <http://www.w3.org/TR/2009/REC-xml-names-20091208/>.
- [5] World Wide Web Consortium (W3C). W3C XML Schema Definition Language (XSD) 1.1 Part 1: Structures, 2012. <http://www.w3.org/TR/2012/REC-xmlschema11-1-20120405/>.
- [6] World Wide Web Consortium (W3C). W3C XML Schema Definition Language (XSD) 1.1 Part 2: Datatypes, 2012. <http://www.w3.org/TR/2012/REC-xmlschema11-2-20120405/>.
- [7] Seamer LC, Bagwell CB, Barden L, Redelman D, Salzman GC, Wood JC, and Murphy RF. Proposed new data file standard for flow cytometry, version FCS 3.0. *Cytometry*, 1997;28(2):118–122.
- [8] World Wide Web Consortium (W3C), Math Working Group. W3C Recommendation – Mathematical Markup Language (MathML) Version 2.0 (Second Edition), 2003. <http://www.w3.org/TR/2003/REC-MathML2-20031021/>.
- [9] Sun Microsystems, Inc. Java Architecture for XML Binding (JAXB): A Primer. <http://java.sun.com/developer/technicalArticles/xml/jaxb/>.
- [10] Bioconductor - Open Source Software for Bioinformatics. <http://www.bioconductor.org/>.
- [11] World Wide Web Consortium (W3C). W3C Recommendation: Document Object Model (DOM) Level 1 Specification, Version 1.0. <http://www.w3.org/TR/1998/REC-DOM-Level-1-19981001/>.
- [12] Megginson D and Brownell D. SAX: Simple API for XML. <http://www.megginson.com/downloads/SAX/>.

- [13] World Wide Web Consortium (W3C). W3C Recommendation - XML Path Language (XPath), Version 1.0. <http://www.w3.org/TR/1999/REC-xpath-19991116/>.
- [14] World Wide Web Consortium (W3C). W3C Recommendation - XSL Transformations (XSLT), Version 1.0. <http://www.w3.org/TR/1999/REC-xslt-19991116/>.
- [15] World Wide Web Consortium (W3C). W3C Working Draft - XML Pointer Language (XPointer). <http://www.w3.org/TR/2002/WD-xptr-20020816/>.
- [16] Spidlen J, Leif RC, Moore W, Roederer M, Brinkman RR, and the International Society for the Advancement of Cytometry Data Standards Task Force. Gating-ML: XML-based gating descriptions in flow cytometry. *Cytometry A*, 2008:73(12):1151–1157.
- [17] Unidata. NetCDF – Network Common Data Form. <http://www.unidata.ucar.edu/software/netcdf/>.
- [18] Shafranovich Y. Common Format and MIME Type for Comma-Separated Values (CSV) File. Internet Society: Request for Comments (4180), 2005:<http://tools.ietf.org/html/rfc4180>.
- [19] O’Rourke J. *Computational Geometry in C*; Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 1998.
- [20] Sunday D. *Geometry Algorithms - Algorithm 3 - Point in a Polygon*.
- [21] Cheney S. *Drawing Polygons*, CS 559-14, Spring 2004.
- [22] Moore WA and Parks DR. Update for the logicle data scale including operational code implementations. *Cytometry A.*, 2012:81A(4):273–277.
- [23] Parks DR, Roederer M, and Moore WA. A new "Logicle" display method avoids deceptive effects of logarithmic scaling for low signals and compensated data. *Cytometry A.*, 2006:69(6):541–551.
- [24] Bagwell CB. Hyperlog-a flexible log-like transform for negative, zero, and positive valued data. *Cytometry A.*, 2005:64(1):34–42.
- [25] Roederer M. Spectral compensation for flow cytometry: visualization artifacts, limitations, and caveats. *Cytometry*, 2001:45(3):194–205.
- [26] Lawson CL and Hanson RJ. *Generalized Inverses: Theory and Applications*, Second Edition. Springer-Verlag GmbH, 2003.
- [27] Ben-Israel A and Greville TN. *Solving Least Squares Problems*. Prentice-Hall, 1974.

Annex B Summary of changes since Gating-ML 1.5

This informative annex provides a brief summary of the major changes to the Gating-ML specification since Gating-ML version 1.5, *i.e.*, since the ISAC Candidate Recommendation from October 30, 2008. A short rationale for these changes is also included.

- Spillover matrices have been replaced with spectrum matrices; these do not have to be square any no longer and, since there is no longer a direct correspondence between rows and columns, the spectrum matrices include separate sets of labels for detectors and fluorochromes. In addition, the option of providing the “already inverted” matrix is also supported.
- Decision tree gates are no longer be supported. These gates were not very useful and their inclusion complicated the implementation of the specification.
- Polytope gates are no longer be supported. These gates were not very useful and their inclusion complicated the implementation of the specification.
- Boolean gates no longer support the in-line definition of their operand gates. The elimination of this option simplifies the specification while still allowing to describe all cases. The operand gates may still be defined outside of the Boolean gate definition and referenced from a Boolean gate. In addition, each operand O may be used as either O (default) or $\text{NOT}(O)$ (by setting the value of the *use-as-complement* attribute to *true*). This allows for a simple creation of expressions such as “ $A \text{ AND NOT}(B)$ ”.
- Quadrant gates have been included. The inclusion of these gates has been requested by users of the specification and the community.
- Event values from FCS files are referenced as *scale values* (see section 3.3.4). Gating-ML 1.5 used to reference channel values and consequently, their conversion to the “scale values” had to be explicitly described in Gating-ML (typically the very first transformation), which unnecessarily complicated the workflow description considering that such a transformation is unambiguously defined within the FCS data file standard already.
- The general framework allowing for any kind of compound transformations is no longer supported. Instead, each gating dimension is defined by
 - (i) One FCS dimension (*i.e.*, values of the $\$PnN$ keywords as stored in the FCS data file or an appropriate alternative for different list mode data file formats) or a newly created dimension (ratio is the only supported transformation that creates a new dimension).
 - (ii) Zero or one explicitly defined compensation description. Technically, each gating dimension states explicitly whether the event values shall be used uncompensated or compensated. For compensated values, it is also specified whether the compensation shall be carried out based on a spectrum (spillover) matrix included in the Gating-ML file or according to the description included in the list mode data file (*e.g.*, the $\$SPILLOVER$ keyword in an FCS file). Gates applied on data compensated according to a Gating-ML spectrum matrix reference their dimensions using the

names of the rows of the matrix, which may be different from the detector names in case of spectral unmixing.

(iii) Zero or one scale transformation.

This new design aspect still supports all typical work flows while significantly simplifying the implementation of the specification.

- Custom and vendor specific information may be included in the Gating-ML file using the *custom_info* elements and additional custom attributes (see section 4.8). The *custom_info* elements may be placed at the top level (*i.e.*, as sub elements of the main *Gating-ML* element) or as sub elements of the *spectrumMatrix* elements, *transformation* elements, or any of the gate definition elements.
- Transformations, such as the polynomial of degree one, the quadratic transformation, the square root transformation, the exponential transformation, the hyperbolic sine transformation, the “EH” transformation (*i.e.*, inverse to Hyperlog), split scale, and the inverse split scale transformation are no longer supported with no equivalent replacement included. Most of these transformations were originally present since Gating-ML 1.5 included inverse transformations for all *display* transformations, which are not needed any more.
- The Logicle transformation has been added. This addition has been made in response to the fact that Stanford no longer requires royalties on the Logicle transformation, see <http://otlportal.stanford.edu/techfinder/technology/ID=23438>.
- Optional transformation boundaries have been added to all scale transformations as specified in section 6.1.
- The terminology has been improved, mostly to avoid confusion and conflicts between general (common) meaning of a term *vs.* its meaning in flow cytometry or this specification. For example, the term “parameter” is no longer used to reference “FCS dimensions” (see section 4.1). In addition, most of the transformations have been renamed in order to distinguish between a common and well defined mathematical function and a specifically parametrized Gating-ML transformation (*e.g.*, *asinh vs. fasinh*).
- Parameterization of the included transformation has been modified to harmonize how parameters are used across different transformations. In addition, the transformations have been altered to map the range of expected event values to the [0,1] interval. These changes simplify the comparison of different transformations and suggest a reasonable fall back strategy in case a particular transformation is not implemented. In addition, these changes simplify scale visualization due to the explicitly known minimum and maximum values (*i.e.*, 0 and 1, respectively, see section 4.7).
- The parametrized ratio transformation has been taken out from the list of “scale” transformations and placed into a special group of transformations creating new dimensions; these may be combined with scale transformations and compensation (spectral unmixing).
- The spillover (now spectrum) matrix coefficient values are no longer restricted to the [0, 1] interval. This restriction was wrongly introduced by Gating-ML 1.5.

- Reference implementation for the included transformations has been provided in order to facilitate correct third party implementations.
- MathML documentation for the transformations is no longer provided. MathML description of the transformation functions has not been useful to any of the users of the previous versions of the Gating-ML specification.

Annex C ISAC Data Standards Task Force Members

Ryan Brinkman, BC Cancer Agency (chair)

Jay Almarode, FlowJo, LLC

Ernie Anderson, Beckman Coulter Life Sciences

Kim Blenman, Yale University

Chris Bray, Verity Software House

Martin Büscher, Miltenyi Biotechnology

James Cavanaugh

Michael Goldberg, BD Biosciences

Bill Hyun, University of California, San Francisco

David Kripal, Cytex Development Inc.

Kevin Krouse, Labkey, Inc.

Robert Leif, Newport Instruments

Wayne Moore, Stanford University

David Novo, De Novo Software

David Parks, Stanford University

Josef Spidlen, BC Cancer Agency

Adam Treister, Fluourish, Inc.

James Wood, Wake Forest University

Michael Zordan, Sony Biotechnology