Image Cytometry Experiment Format

International Society for Advancement of Cytometry Candidate Recommendation

Document Status

This document is an ISAC Candidate Recommendation of the Image Cytometry Experiment Format specification. Based on additional feedback from implementors and users of this specification, features and design aspects specified in this document may be changed in the final version of the Recommendation.

This 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.

Disclaimer of Liability

The International Society for Advancement of Cytometry (ISAC) disclaims liability for any injury, harm, or other damage of any nature whatsoever, to persons or property, whether direct, indirect, consequential or compensatory, directly or indirectly resulting from publication, use of, or reliance on this Specification, and users of this Specification, as a condition of use, forever release ISAC from such liability and waive all claims against ISAC that may in any manner arise out of such liability. ISAC further disclaims all warranties, whether express, implied or statutory, and makes no assurances as to the accuracy or completeness of any information published in the Specification.

In issuing and making this Specification available, ISAC is not undertaking to render professional or other services for or on behalf of any person or entity, nor is ISAC undertaking to perform any duty owed by any person or entity to someone else. Anyone using this document should rely on his or her own independent judgment or, as appropriate, seek the advice of a competent professional in determining the exercise of reasonable care in any given circumstances.
Attention is called to the possibility that implementation of this Specification 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.

ICEFormat 1.1, ISAC Candidate Recommendation, version 111214, December 14, 2011

Abstract

In image cytometry, most instruments produce image and feature data in proprietary file formats that are virtually impossible to process by third party software and therefore, researchers are tightly bound to analytical software provided by the specific hardware manufacture. Consequently, the current state hinders interoperability and independent validation of experimental results.

This document represents the Image Cytometry Experiment Format (the ICEFormat), an open file format to capture image cytometry data, metadata and features in a standardized manner allowing these to be processed by independently developed software application. The adoption of the ICEFormat can facilitate the interchange and validation of data between different software packages with the potential of significant interoperability increase. It supports various use cases including individual and composite images, masks with a variety of bit depths, features of multiple data types and high throughput plate-based experiments. The ICEFormat strikes a balance between readability and file size by combining an XML file containing metadata about identified objects and pointers to externally stored binary files containing image data, masks and feature values. Optionally, these can be bundled together using the Archival Cytometry Standard (ACS) file format.

Keywords: image cytometry, cytology, data standard, file format, XML

TABLE OF CONTENTS

Image Cytometry Experiment Format

1. Overview

1.1 Introduction

In image cytometry, most instruments produce and store image and feature data in proprietary file formats that are difficult or even impossible to process by third party software. Consequently, researchers are limited to the use of analytical software provided by the manufacture of their particular instrument. The current state hinders interoperability and independent validation of experimental results.

In order to address this issue, the Image Cytometry Experiment Format (the ICEFormat) has been developed as an open file format capturing image cytometry data, metadata, features and their values in a standardized manner. The adoption of the ICEFormat will allow image cytometry data to be processed and analyzed by independently developed software applications and therefore, it will facilitate the interchange of image cytometry data between different software packages with the potential to increase interoperability and support scientific collaboration and validation.

1.2 Scope

This document provides detailed specification of the ICEFormat. It covers the following:

— The syntax specification of the ICEFormat XML file capturing metadata about identified objects and pointers to externally stored image data, masks and feature values.

— The syntax specification of binary files storing object masks referenced from the ICEFormat XML file.

— The syntax specification of files storing feature values for features and object defined by the ICEFormat XML file.

This document also demonstrates how the ICEFormat components (i.e., ICEFormat XML file, masks, image data and feature data) can be bundled together using the Archival Cytometry Standard (ACS [1]) specification. This demonstration is provided for informative purpose only; please consult the ACS specification as a normative reference and for additional options on how the ICEFormat components can be bundled.

This document does not specify file formats to store image data; relevant established open image file formats can be utilized for this purpose. Image file formats that are proprietary, vendor-specific, that do not have a publicly available specification, or that require the acquisition of a license in order to decode the image shall not be used as ICEFormat components. In addition, image formats that are highly flexible (such as TIFF [2]) should be used in the simplest configuration possible in order to facilitate interoperability between different software packages.

1.3 Purpose

The purpose of this specification is to provide standard means to store and electronically communicate image cytometry data including images, masks and feature values so that these become available to third party software applications for further analysis. Ultimately, the ICEFormat aims to facilitate cross-platform sharing of image cytometry data, metadata and analysis descriptions between different software packages.

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, XML Schema Part 0: Primer Second Edition [4].

— W3C Recommendation, XML Schema Part 1: Structures Second Edition [5].

— W3C Recommendation, XML Schema Part 2: Datatypes Second Edition [6].

The following documents are useful for the application of this standard. They represent other standards and standard proposals relevant to this specification.

— 754-2008 IEEE Standard for Floating-Point Arithmetic [7].

— ISAC Candidate Recommendation DRAFT, Archival Cytometry Standard [1].

1.5 The Content of this Specification

This specification consists of the following parts:

a) Normative: This document providing a detailed description of the ICEFormat.

b) Normative: The XML schema ICE.xsd defining syntax of the ICEFormat XML file.

c) Normative: The XML schema ICEStrValues.xsd defining syntax of XML files capturing feature values of features whose data type is a character string.

d) Informative: Examples of ICEFormat files.

e) Informative: HTML documentation of ICE.xsd and ICEStrValues.xsd XML schemas.

All the components of this standard are available from World Wide Web at http://www.isac-net.org/ as well as http://flowcyt.sf.net/.

1.6 Acronyms and Abbreviations

ACS	Archival Cytometry Standard
HTML	Hypertext Markup Language
ICEFormat	Image Cytometry Experiment Format
IEEE	Institute of Electrical and Electronics Engineers
ISAC	International Society for Advancement of Cytometry
RFC	Request for Comments
TIFF	Tagged Image File Format
URL	Uniform Resource Locator
W3C	World Wide Web Consortium

XML	Extensible Markup Language

1.7 Terminology

For the purpose of this specification, the following terms are understood as follows:

1.7.1 Image

An image is a (typically two-dimensional) digital representation of one or more objects captured by an optical device. An image of (an) object(s) has a similar appearance to the object(s) being captured. We distinguish individual images (i.e., one image per object) and composite images (i.e., multiple objects captured in a single image with masks used to specify objects within the image). In addition, images may (optionally) belong to a specific channel.

1.7.2 Object

An object is the result of a segmentation algorithm applied to an image. Depending on the segmentation algorithm used, the settings, and the environment, the object may represent different entities, such as a cell, a subcellular structure such as a cell nucleus, chromosome, etc., or a supercellular structure, such as a gland, vessel etc.

1.7.3 Segmentation

Segmentation is the process of partitioning a digital image in order to locate objects visualized in the image. Segmentation is typically performed by a segmentation algorithm. While the standard does not provide a mechanism to list all possible segmentation algorithms and their various parameters and settings, segmentations may be defined and each mask may (optionally) be assigned to segmentation.

1.7.4 Mask

A mask is a specification of which parts of an image (i.e., which pixels) correspond to specific object(s) and which are considered the image background. Note that a mask can define multiple objects in an image. In our representation (see section 5), the mask is a two-dimensional array, which size corresponds to the size of the image that the mask shall be applied to. In this array, a positive number indicates an object (of an identifier corresponding to that number) being present at a specific pixel. Zero indicates that there is no object (i.e., just background) at a specific pixel is the image. Consequently, the objects specified by a single mask are non-overlapping and the number of objects specifiable by a mask is dependent on the bit depth of the mask definition.

1.7.5 Channel

For the purpose of this specification, a channel is combination of specific excitation, emission and detection parameters and settings that resulted in an image. While the standard does not provide a mechanism to list all possible parameters and settings that could describe a channel, channels may be defined and each image may (optionally) be assigned to a channel in order to allow different images to be grouped as being acquired via a particular channel.

1.7.6 Feature

A feature is a characteristic of objects (e.g., the nuclear intensity, cytoplasmic intensity, etc.). Features have various data types, commonly floating point numbers but also integer numbers, character strings, Boolean, an image associated with an object, etc. A feature value is the actual value of that characteristic on a specific object.

1.8 Keywords Indicating Requirement Levels

The key words *shall*, *should*, and *may* in this document are to be interpreted as described in RFC 2119 [8] and are also compatible with the IEEE Standards Style Manual. The word *shall* is used to indicate mandatory requirements to be followed in order to conform to the standard and from which no deviation is permitted (*shall* equals *is required to*). The word *should* is used to indicate that among several possibilities one is recommended as particularly suitable, without mentioning or excluding others; or that a certain course of action is preferred but not necessarily required; or that (in the negative form) a certain course of action is deprecated but not prohibited (*should* equals *is recommended that*). The word *may* is used to indicate a course of action permissible within the limits of the standard (*may* equals *is permitted to*).

1.9 Namespaces and their Prefixes within this Document

The following XML namespaces [9] are being used in this specification:

a) **File ICE.xsd**

Prefix	**Namespace**
none	`http://www.isac-net.org/std/ICEFormat/1.0/ice`
`xs:`	`http://www.w3.org/2001/XMLSchema`

b) **File ICEStrValues.xsd**

Prefix	**Namespace**
none	`http://www.isac-net.org/std/ICEFormat/1.0/iceStrValues`
`ice:`	`http://www.isac-net.org/std/ICEFormat/1.0/ice`
`xs:`	`http://www.w3.org/2001/XMLSchema`

c) **ICEFormat XML examples**

Prefix	**Namespace**
none	`http://www.isac-net.org/std/ICEFormat/1.0/ice`
`xsi:`	`http://www.w3.org/2001/XMLSchema-instance`

d) **Examples of XML files capturing feature values whose data type is a character string**

Prefix	**Namespace**
none	`http://www.isac-net.org/std/ICEFormat/1.0/iceStrValues`
`xsi:`	`http://www.w3.org/2001/XMLSchema-instance`

2. Design Principles and Rationale

2.1 Using XML for Metadata and Experiment Data Directory

XML technology was chosen for the implementation of the ICEFormat Data Directory file (section 4) since it is a well established and supported technology, free of any of legal encumbrances, and it provides a robust and durable format for information storage and transmission. It can be used to describe and identify information accurately and unambiguously, in a way that computers and humans can easily interpret and manipulate it. XML allows for separation of the content from the visual appearance. Moreover, it allows documents to be created and handled consistently and without structural errors since it provides a standardized way of describing, controlling, and allowing/disallowing particular types of document structure. The usage of XML for the ICEFormat Data Directory file facilitates the interchange and validation of ICEFormat between different software packages with the potential of increase of hardware and software interoperability.

2.2 Keeping Images, Masks and Feature Values in Separate Files

Image cytometry commonly produces large amounts of data. Therefore, an appropriate data organization is needed for performance reasons as well as to keep a clear design so that data remains easily readable. We have chosen to keep the ICEFormat Data Directory in a dedicated XML file. As shown in Figure 1, the actual images, masks, and feature values are stored in separate files using appropriate file formats. These files are referenced from the ICEFormat Data Directory file. This design allows us to keep the ICEFormat Data Directory reasonably small so that it can be held in memory for fast processing. Moreover, it allows us to keep existing file formats for the image data and use efficient binary representations for masks and (most) feature values. This design is also consistent with the Archival Cytometry Standard (ACS) paradigm and the ACS specification can be used to bundle all the ICEFormat components in a single archive file.

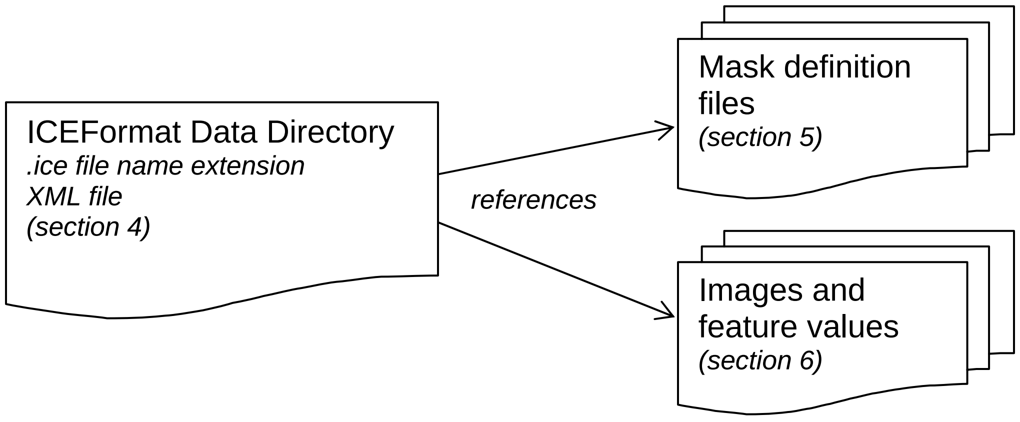

Figure 1 – ICEFormat structure with a data directory, masks, images and feature values

2.3 Storing Boolean Feature Values as Whole Bytes

Storing Boolean feature values as single bytes rather than single bits may be considered as a waste of space. However, there are significant advantages of this approach. First, it allows us to capture the value “unknown” rather than a simple “true” and “false” (section 4.5.3). More importantly, this representation does not break byte boundaries and therefore, it will allow us to keep Boolean feature values along with other types of feature values in a single binary file (section 6.1). Consequently, this approach simplifies the specification with no need to introduce special cases for Boolean values. Finally, we believe that the increased space requirements have minimal impact on the total size of the dataset considering that the image data and other types of feature values represent the components that significantly contribute to space requirements of a dataset. In addition, a file with Boolean values captured as single bytes will compress extremely well if the ICEFormat structure is bundled in an ACS container.

3. Conformance

3.1 File Conformance

To be conformant with this standard, an ICEFormat file shall meet the following requirements:

a) The file name of the ICEFormat Data Directory XML file shall contain the `.ice` file name extension as stated in section 4.1.

b) The ICEFormat Data Directory shall be a valid XML file conformant to W3C Recommendation, Extensible Markup Language (XML) 1.0 (Fourth Edition) [3].

c) The ICEFormat Data Directory shall be valid according to the provided ICE.xsd XML schema.

d) Files with images, masks and feature values that are relevant to the particular image cytometry experiment shall be referenced from the ICEFormat Data Directory using Uniform Resource Locators (URLs) relative to the ICEFormat Data Directory XML file. The slash character shall be used to separate directories in the file path. Absolute URLs and URLs referring to the local file system above the directory with the ICEFormat Data Directory file (or outside of an ACS container if ACS is being used) shall not be used. For example, these URLs shall be avoided: file://C:\My Documents\Images\img1.gif, file://../images/i001.png.

e) The ICEFormat Data Directory XML file shall conform to the specification stated in section 4.

f) Mask definitions shall be stored in binary files as stated in section 5.

g) Feature values including image data shall be stored in additional files as stated in section 6.

3.2 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” ICEFormat and it shall meet the following criteria:

— *Writing*: When ICEFormat files are produced (written) then these shall be valid ICEFormat files according to section 3.1.

— *Reading*: The software application (or the hardware instrument) shall be able to read any ICEFormat file structure that is valid according to section 3.1, process the ICEFormat Data Directory file, and locate and load image data files, masks and feature values. The software application (or hardware instrument) shall also be able to process at least one of the commonly used image file formats, understand and apply mask(s) in order to identify objects in images and assign stored feature values to defined objects.

4. ICEFormat Data Directory

4.1 General

An ICEFormat Data Directory file is a valid XML [3] file named with the `.ice` file name extension.

4.2 ICEFormat Data Directory File Structure

An ICEFormat Data Directory XML file shall successfully validate against the provided ICE.xsd XML schema file. The *ICEFormat* element (*ice* namespace, section 1.9) is expected as the root element of the XML file. The *ICEFormat* element shall include the *version* attribute that is fixed to the value of “1.0” in this version of the specification. The following sub-elements may be used:

- *ChannelDefinitions*: Definitions of channels (optional, section 4.3)
- *SegmentationDefinitions*: Definitions of segmentations (optional, section 4.4)
- *FeatureDefinitions*: Definitions of features (optional, section 4.5)

After these optional elements, either *DataSet* (multiple *DataSet* elements may be present) or *Plate* (in case of a plate-based experiment(s), multiple *Plate* elements may be used) shall be used. All of these elements are defined in the *ice* namespace (section 1.9). In addition, a single *Sitemap* element may be included to provide information about positions of image data from multiple sites were collected. A syntax description of an ICEFormat Data Directory XML file follows:

```
<?xml version="1.0" encoding="utf-8"?>
<ICEFormat
    xmlns = "http://www.isac-net.org/std/ICEFormat/1.0/ice"
    xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation = "http://www.isac-net.org/std/ICEFormat/1.0/
                          http://flowcyt.sf.net/ice/ICE.xsd"
    version = "1.0">

  <!-- … Description of the contents of the ICEFormat Data Directory file … -->

  <ChannelDefinitions>
       … Definitions of channels (optional, section 4.3) …
  </ChannelDefinitions>

  <SegmentationDefinitions>
       … Definitions of segmentations (optional, section 4.4) …
  </SegmentationDefinitions>

  <FeatureDefinitions>
       … Definitions of features (optional, section 4.5) …
  </FeatureDefinitions>

  <DataSet>
       … Description of a dataset (section 4.6, there could also be multiple datasets) …
       … Alternativelly, a Plate could be described (section 4.7) …
  </DataSet>

  <Sitemap>
       … In addition, a Sitemap may be included (optional, section 4.8)…
  </Sitemap>

</ICEFormat>
```

4.3 Channel Definitions

Channel definitions may (optionally) be provided as part of the ICEFormat Data Directory. If provided, these shall be enclosed by the *ChannelDefinitions* element containing a sequence of one or more *Channel* elements. Each *Channel* element shall contain the *Id* attribute specifying the identifier of the channel that may be used further to reference that particular channel. Channel identifiers shall be unique among all defined channels within the ICEFormat Data Directory. In addition, a *Description* attribute may be used to provide a free text description of the channel. Finally, additional custom-defined attributes and sub-elements may be used to specify further details about the channel. An example of a channel definition follows:

```
<ChannelDefinitions>
  <Channel Id="c12345" Description="488nm laser, 533/20 BP filter"
           Additional_custom_attribute="some value">
    <Additonal><custom>metadata</custom></Additonal>
  </Channel>
  <!-- Additional channels may be defined -->
</ChannelDefinitions>
```

4.4 Segmentation Definitions

Segmentation definitions may (optionally) be provided as part of the ICEFormat Data Directory. If provided, these shall be enclosed by the *SegmentationDefinitions* element containing a sequence of one or more *Segmentation* elements. Each *Segmentation* element shall contain the *Id* attribute specifying the identifier of the segmentation that may be used further to reference that particular segmentation. Segmentation identifiers shall be unique among all defined segmentations within the ICEFormat Data Directory. In addition, a *Description* attribute may be used to provide a free text description of the segmentation. Finally, additional custom-defined attributes and sub-elements may be used to specify further details about the segmentation. An example of a segmentation definition follows:

```
<SegmentationDefinitions>
  <SegmentationId="Seg12345" Description="Segmentation Algorithm XYZ"
           Additional_custom_attribute="some value">
    <Additonal><custom>metadata</custom></Additonal>
  </Segmentation>
  <!-- Additional segmentations may be defined -->
</SegmentationDefinitions>
```

4.5 Feature Definitions

Features shall be defined by a *FeatureDefinitions* element containing a sequence of *FeatureDefinition* sub-elements, one for each feature being defined and recorded by the ICEFormat. There are two places where features may be defined within the ICEFormat Data Directory. For features that are shared among all datasets within the ICEFormat Data Directory, the *FeatureDefinitions* element shall be placed as a sub-element of the *ICEFormat* root element just before a sequence of *DataSet* elements or (a) Plate(s). Features that are specific for a single dataset within the ICEFormat Data Directory may be defined within the particular dataset as mentioned in section 4.6.2. A combination of globally defined features and features specific for a particular dataset may also be specified. Finally, if there are features that are shared among some but not all datasets, then these features may be defined on the dataset level (section 4.6.2) with corresponding features using the same identifiers in order to allow for these to be properly matched across datasets.

In cases where the order of features is important, features are considered to maintain the order in which they have been defined. Features that have been defined without associating any values with them (see section 4.6.5) will typically be skipped (omitted) by ICEFormat processing software applications. The content of the *FeatureDefinition* element depends on the data type of the feature being defined. Generally, there is an *Info…* element with further sub-elements refining the definition. Two sub-elements are shared for all types of features:

— *Description*: text based description of the feature; may be left blank and does not have to be unique among features.

— *ID*: a non-empty identifier of the feature; this shall be unique among all features defined in the ICEFormat Data Directory (i.e. as stated here as well as in section 4.6.2).

Additional sub-elements may be required based on the data type of the feature. The following types of features are supported:

4.5.1 Integer-based Features

A feature of the integer data type is defined by the *InfoInt* element. This element shall include the following additional sub-element:

— *BitDepth*: The bit depth used to store feature values of this particular feature, one of 8, 16, or 32 shall be used. Depending on this value, the actual features will be captured using little-endian encoded signed integer values occupying either 1 byte (8 bits), 2 bytes (16 bits), or 4 bytes (32 bits) in a binary feature values file. Please refer to sections 4.6.5 a) and 6.1 for information how to store feature values.

Example:

```
<FeatureDefinition>
  <InfoInt>
    <Description>Number of bright pixels</Description>
    <ID>F001</ID>
    <BitDepth>16</BitDepth>
  </InfoInt>
</FeatureDefinition>
```

4.5.2 Floating-point-based Features

A feature of the floating integer data type is defined by the *InfoFloat* element. This element shall include the following additional sub-element:

— *BitDepth*: The bit depth used to store feature values of this particular feature, one of 32 or 64 shall be used. Depending on this value, the actual feature values will be captured using little-endian encoded floating point values occupying either 4 bytes (32 bits) or 8 bytes (64 bits) in a binary feature values file and saved in accordance with the IEEE 754 specification [7]. Please refer to sections 4.6.5 a) and 6.1 for information how to store feature values.

Examples:

```
<FeatureDefinition>
  <InfoFloat>
     <Description>Area</Description>
     <ID>F002a</ID>
     <BitDepth>32</BitDepth>
   </InfoFloat>
</FeatureDefinition>
```

```
<FeatureDefinition>
  <InfoFloat>
     <Description>Cytoplasmic Intensity</Description>
     <ID>F002b</ID>
     <BitDepth>64</BitDepth>
   </InfoFloat>
</FeatureDefinition>
```

4.5.3 Boolean-based Features

A feature of the Boolean data type is defined by the *InfoBoolean* element. This element shall include the following additional sub-element:

- *BitDepth*: The bit depth used to store feature values of this particular feature is fixed to 8 bits (1 byte) per a Boolean values. The actual features will be captured using little-endian encoded signed integer values occupying 1 byte (8 bits), with binary 0 indicating “false” and binary 1 indicating “true”, and any other value indicating “unknown”. Please refer to sections 4.6.5 a) and 6.1 for information how to store feature values.

Example:

```
<FeatureDefinition>
  <InfoBoolean>
    <Description>Apoptotic?</Description>
    <ID>F003</ID>
    <BitDepth>8</BitDepth>
  </InfoBoolean>
</FeatureDefinition>
```

4.5.4 Composite Images

The visual representation of objects defined by a specific mask applied to a particular composite image is considered as a type of a feature. This feature is defined by the *InfoCompositeImage* element. The actual feature value is created by applying the specified mask on a specific composite image to retrieve a particular visual representation of an object. The *InfoCompositeImage* element includes the following additional sub-elements:

- *ChannelID*: Optional, the reference to a channel associated with this image. The channel shall be referenced by its ID. See section 4.3 for information how to define channels.
- *ImageID*: Required, the reference to a composite image that this feature is related to. The composite image shall be referenced by its ID. See section 4.6.3 for information how to describe composite images.
- *MaskID*: Required, the reference to a mask definition that this feature is related to. The mask shall be referenced by its ID. See section 4.6.4 for information how to describe masks.

Example:

```
<FeatureDefinition>
  <InfoCompositeImage>
    <Description>Green Filter</Description>
    <ID>F004a</ID>
    <ChannelID>c12345</ChannelID>
    <ImageID>I001</ImageID>
    <MaskID>M001</MaskID>
  </InfoCompositeImage>
</FeatureDefinition>

<FeatureDefinition>
  <InfoCompositeImage>
    <Description>Red Filter</Description>
    <ID>F004b</ID>
    <ImageID>I002</ImageID>
    <MaskID>M001</MaskID>
  </InfoCompositeImage>
</FeatureDefinition>
```

4.5.5 Individual Images

The visual representation of objects in individual images (i.e., one image per object) is considered as a type of a feature. This feature is defined by the *InfoIndividualImage* element with the shared *Description* and *ID* sub-elements. In addition, a reference to a channel associated with the image may be provided using the *ChannelID* element. The channel shall be referenced by its ID. See section 4.3 for information how to define channels. The actual feature value is an image of the object.

Example:

```
<FeatureDefinition>
  <InfoIndividualImage>
    <Description>Filter 1</Description>
    <ID>F005</ID>
    <ChannelID>c12345</ChannelID> <!-- Optional -->
  </InfoIndividualImage>
</FeatureDefinition>
```

4.5.6 String-based Features

A feature of the string (of characters) data type is defined by the *InfoString* element with no additional sub-elements except for the shared *Description* and *ID*. As it is the case with all feature values, they shall be stored in separately from the ICEFormat Data Directory XML file. In this case, the actual feature values shall be stored in a separate XML file corresponding to the ICEStrValues.xsd XML schema. Please refer to sections 4.6.5 a) and 6.3 for information how to store feature values.

Example:

```
<FeatureDefinition>
  <InfoString>
    <Description>Object Name</Description>
    <ID>F006a</ID>
  </InfoString>
</FeatureDefinition>

<FeatureDefinition>
  <InfoString>
    <Description>Object Description</Description>
    <ID>F006b</ID>
  </InfoString>
</FeatureDefinition>
```

4.5.7 Object Classification

Object classification, i.e., object assignment to one of the predefined classes, is a type of feature defined by the *InfoClassification* element. This element shall include the following additional sub-elements:

— *BitDepth*: The bit depth used to store feature values of this particular feature, one of 8, 16, or 32 shall be used. Depending on this value, the actual features will be captured using little-endian encoded unsigned integer values occupying either 1 byte (8 bits), 2 bytes (16 bits), or 4 bytes (32 bits) in a binary feature values file. The value 0 is reserved to indicate that an object has not been assigned to any class. Values 1, 2, 3, … correspond to classes in the same order as they are defined by the *Class* elements that are following after the *BitDepth* element (see below). Please refer to sections 4.6.5 a) and 6.1 for information how to store feature values.

— *Class*: A sequence of *Class* elements shall follow after the *BitDepth* element. There shall be one *Class* element for each of the class defined; specifically, the number of defined classes shall be greater or equal to the largest integer value present in the appropriate feature values binary file (section 6.1). Note: Typically, these two numbers will be equal; however, the number of defined classes may eventually be greater than the number of classes present in a particular dataset (e.g., there is a class X but no object has been classified as class X). On the other hand, the binary feature value file (section 6.1) shall not contain any class that has not been defined in the ICEFormat Data Directory XML file.

Example:

```
<FeatureDefinition>
  <InfoClassification>
    <Description>Cell type</Description>
    <ID>F007</ID>
    <BitDepth>8</BitDepth>
    <Class>T cell</Class>
    <Class>B cell</Class>
  </InfoClassification>
</FeatureDefinition>
```

4.5.8 Associations Among Objects

A feature of the association type is defined by the *InfoAssociation* element. Values of this feature are essentially integers but their semantic is to provide association links between objects, which may come from different datasets, segmentations, etc. The association link is established by matching values of features of the association type (objects with the same value of this feature shall be associated). All datasets defined within the current ICEFormat Data Directory shall be parsed for associated features. Only features with the same identifier shall be matched, i.e., the association feature may either be defined globally for multiple datasets or the same feature identifier shall be used in case features are defined on the dataset-specific level (see section 4.6.2) or in multiple ICEFormat files.

Similar to the integer-based feature type, this element shall include the following additional sub-element:

— *BitDepth*: The bit depth used to store feature values of this particular feature, one of 8, 16, or 32 shall be used. Depending on this value, the actual features will be captured using little-endian encoded signed integer values occupying either 1 byte (8 bits), 2 bytes (16 bits), or 4 bytes (32 bits) in a binary feature values file. Please refer to sections 4.6.5 a) and 6.1 for information how to store feature values.

This feature allows for the description of associations with the one-to-one as well as the one-to-many multiplicities.

Example:

```
<FeatureDefinition>
  <InfoAssociation>
    <Description>Granules to cells assignment</Description>
    <ID>GC001</ID>
    <BitDepth>16</BitDepth>
  </InfoAssociation>
</FeatureDefinition>
```

Let’s assume there are two datasets within the ICE Format Data Directory and the GC001 feature is applicable to both. One dataset contains “cells” and the values of the GC001 feature are 101, 102, 103 (there are 3 objects in the dataset containing cells). The other dataset contains “granules” and the values of this feature are 101, 102, 101, 102, 102, 104 (there are 6 objects in the granules dataset). From this description we could infer that cell 1 is associated with granules 1 and 3, cell 2 with granules 2, 4 and 5, there are no granules associated with the last cell number 3, and there are no cells associated with the last granule 6.

If there were additional datasets within the current ICE Format Data Directory then these shall all be parsed to identify any additional associations linking to these datasets.

4.6 Description of a Dataset

A data set is described by the *Dataset* element containing a sequence of the following sub-elements: *MetaData* (section 4.6.1), *FeatureDefinitions* (optional, section 4.6.2), *CompositeImages* (optional, section 4.6.3), *Masks* (optional, section 4.6.4), and *FeatureValues* (section 4.6.5). In image cytometry, image data from multiple sites may be collected. The *Dataset* element may also contain a *SiteRef* attribute that references a specific site associated with this particular dataset. If *SiteRef* is used then a site map specifying positions of various sites may be used (see section 4.8 for details). Please note that multiple datasets may be described even if the *SiteRef* attribute is not used. Such a situation should be interpreted that there is no specific site plan or that the site position is irrelevant. Also, the *SiteRef* attribute may be used without the site map being defined.

The syntax of the *Dataset* element is shown below:

```
<DataSet SiteRef="s1">
  <MetaData> Metadata definitions </MetaData>
  <FeatureDefinitions> Feature definitions (optional) </FeatureDefinitions>
  <CompositeImages> Optional definitions of composite images </CompositeImages>
  <Masks> Optional masks definitions </Masks>
  <FeatureValues> Definition and references to feature values </FeatureValues>
</DataSet>
```

4.6.1 Dataset Metadata

Metadata of the dataset shall be described in the *MetaData* element.

a) **Number of objects**

The *MetaData* element shall contain a required sub-element named *NumberOfObjects* specifying the number of objects described by this ICEFormat structure. The number of objects shall correspond to the number of objects described in feature values (section 4.6.5).

b) **Timestamp**

An optional *Timestamp* element may follow after the *NumberOfObjects*. The timestamp may be specified as absolute (using the *Absolute* sub-element) and/or relatively using the *Relative* sub-element. If the absolute notation is used, the value of the *Absolute* element shall correspond to the dateTime data type as specified in XML Schema Part 2: Datatypes Second Edition [6], for example 2011-04-04T14:05:22+08:00. If the relative notation is used, the *Relative* element shall include a *Value* attribute (a floating point number) and, optionally, it may include a *Unit* attribute. If a unit is used then it should be one of the suitable units of time according to the International System of Units (SI). The following SI units may be used for this purpose:

• s	second (the base SI unit of time)
• ms	millisecond (10^{-3} second)
• us	microsecond (10^{-6} second)
• ns	nanosecond (10^{-9} second)
• ps	picosecond (10^{-12} second)

c) **Z-position**

An optional *Z-position* element may be included. Similarly to the timestamp, the Z-position may be specified as absolute (using the *Absolute* sub-element) and/or relatively using the *Relative* sub-element. In both cases, the *Absolute* or the *Relative* element shall include a *Value* attribute (a floating point number) and, optionally, it may include a *Unit* attribute. If a unit is used then it should be one of the suitable units of length according to the International System of Units (SI). The following SI units may be used for this purpose:

• m	metre (the base SI unit of length)
• dm	decimetre (10^{-1} metre)
• cm	centimetre (10^{-2} metre)

- mm millimetre (10^{-3} metre)
- um micrometre (10^{-6} metre)
- nm nanometre (10^{-9} metre)
- pm picometre (10^{-12} metre)

d) **Custom metadata**

Additional metadata may be provided using one or more *Custom* elements. The type of provided additional metadata is purposefully not specified by this specification since it may be highly variable depending on what instruments are being used and which use cases are being addressed. The *Custom* element is not restricted by this specification as long as the document remains a valid XML.

Two examples of the *MetaData* element follow:

```
<MetaData>
  <NumberOfObjects>7654</NumberOfObjects>
  <Timestamp> <!-- Timestamp is optional -->
    <-- Absolute, Relative or both may be provided -->
    <Absolute>2011-04-04T14:05:22+08:00</Absolute>
    <Relative Value="23.5" Unit="ms" /> <!-- Unit is optional -->
  </Timestamp>
  <Z-position> <!-- Z-position is optional -->
    <Relative Value="20" Unit="nm" />  <!-- Unit is optional -->
  </Z-position>
  <Custom> <!-- Optional, 0 or more Custom elements -->
    <DataGeneratedBy>
      <Make>Amnis Corporation</Make>
      <Model>ImageStream 100</Model>
    </DataGeneratedBy>
  </Custom>
  <Custom>Software: IDEAS</Custom>
  <Custom>One<can>have</can>any well formed XML <here/>.</Custom>
</MetaData>

<MetaData>
  <NumberOfObjects>123</NumberOfObjects>
  <Timestamp> <!-- Timestamp is optional -->
    <Relative Value="23.5" Unit="ms" /> <!-- Unit is optional -->
  </Timestamp>
  <Z-position> <!-- Z-position is optional -->
    <Absolute Value="100" Unit="nm" />  <!-- Unit is optional -->
  </Z-position>
</MetaData>
```

4.6.2 Dataset-specific Feature Definitions

Features shall be defined by a *FeatureDefinitions* element containing a sequence of *FeatureDefinition* sub-elements, one for each feature being defined and recorded by the ICEFormat. There are two places where features may be defined within the ICEFormat Data Directory. For features that are shared among multiple datasets within the ICEFormat Data Directory, the *FeatureDefinitions* element shall be placed as a sub-element of the *ICEFormat* root element just before a sequence of *DataSet* elements or (a) Plate(s). Features that are specific for a single dataset may be defined directly within the particular dataset definition. The syntax of feature definition is the same for both, global and dataset-specific features. Please see section 4.5 for details about feature definitions.

4.6.3 Composite Images

The *CompositeImages* element may be present to list composite images present in the dataset. Each image is listed by an *Image* element stating the identifier, URL and dimensions of the image with the following sub-elements:

- *ID*: a non-empty identifier of the image; this shall be unique among all images.
- *URL*: a URL element pointing to the image; the element shall contain the following attributes:
  - *url* (required): The actual Uniform Resource Locators (URL) of the file that contains the image. The URL shall use the *file* schema and shall be relative to the location of the ICEFormat Data Directory XML file. The *url* is the only attribute if the file format captures a single image only. Additional attributes as detailed below are used to reference specific images within multi-frame image file formats, such as multi-page TIFFs or various movie file formats.
  - *page* (optional): A page attribute shall be used as a one-based index (positive integer) to reference a single image within the file format used if that contains multiple images. For example, the page attribute shall be used to specify a page number in a multi-page TIFF, or a frame number in a movie file format. The first image (page, frame, etc.) within the file shall be referenced by setting the page attribute to 1; the second image shall be referenced by setting the page attribute to 2, etc.
- *Width*: Width of the image in pixels.
- *Height*: Height of the image in pixels.

An example of the *CompositeImages* element follows:

```
<CompositeImages> <!-- Association with channels described in 4.5.4 -->
  <Image>
    <ID>I001</ID>
    <URL url="file://Images/Green.tif" />
    <Width>512</Width>
    <Height>512</Height>
  </Image>
  <Image>
    <ID>I002</ID>
    <URL url="file://Images/Red.tif" page=”2” />
    <Width>512</Width>
    <Height>512</Height>
  </Image>
</CompositeImages>
```

Any suitable established open image file format may be used to store the actual image data files. Image file formats that are proprietary, vendor-specific, that do not have a publicly available specification, or that require the acquisition of a license in order to decode the image shall not be used. In addition, image formats that are highly flexible (such as TIFF [2]) should be used in the simplest configuration possible in order to facilitate interoperability between different software packages.

4.6.4 Masks

The *Masks* element may be present to list masks used to isolate objects in composite images. Each mask is listed by a *Mask* element. Similarly to a composite image, a mask is defined by an identifier, URL and size of the mask. In addition to an image, a mask shall contain a bit depth specification and it may contain an enumeration of objects (object numbers) present in the mask definition file. The *Mask* element contains the following sub-elements:

- *ID*: a non-empty identifier of the mask; this shall be unique among all masks.
- *URL*: a Uniform Resource Locators (URL) of the mask definition file; the URL shall use the *file* schema and shall be relative to the location of the ICEFormat Data Directory XML file.
- *Width*: Width of the mask in pixels; the width shall correspond to the width of the composite image(s) that the mask is applicable to.

- *Height*: Height of the mask in pixels; the height shall correspond to the height of the composite image(s) that the mask is applicable to.
- *BitDepth*: The bit depth used to store the mask definition in a binary file; one of 8, 16, or 32 shall be used. Depending on this value, there will be 1 byte (8 bits), 2 bytes (16 bits), or 4 bytes (32 bits) reserved for each pixel of the mask. See section 5 for details on how to store masks in binary files.
- *SegmentationID*: Reference to a segmentation may be present if this particular mask shall be associated with one of the defined segmentations. Segmentations shall be referenced by their ID identifiers. See section 4.4 for information about how to define segmentations.
- *MaskObjectNumber*: A sequence of *MaskObjectNumber* elements may be present. If it is present, there shall be *n MaskObjectNumber* elements where *n* corresponds to the number of objects as specified in the data set metadata (section 4.6.1). In this case, each *MaskObjectNumber* element contains a positive integer number that corresponds to the number used to reference that object in the binary mask definition file (section 5). This construct allows for a binary mask file to be reused for a subset of an original set of objects (e.g., cells). However, if the sequence of *MaskObjectNumber* elements is not present, the object numbers in the mask are expected to be numbered sequentially as 1, 2, 3, ..., *n*.

An example of the *Masks* element follows:

```
<Masks>

  <Mask>
    <ID>M001</ID>
    <URL>file://Masks/Mask001.bin</URL>
    <Width>512</Width>
    <Height>512</Height>
    <BitDepth>8</BitDepth>
    <SegmentationID>S12345</SegmentationID> <!-- Optioanl -->
    <MaskObjectNumber>23</MaskObjectNumber>
    <MaskObjectNumber>45</MaskObjectNumber>
    <MaskObjectNumber>67</MaskObjectNumber>
  </Mask>

  <Mask>
    <ID>M002</ID>
    <URL>file://Masks/Mask002.bin</URL>
    <Width>512</Width>
    <Height>512</Height>
    <BitDepth>8</BitDepth>
  </Mask>

</Masks>
```

See section 5 for details on how to store masks in binary files.

4.6.5 Feature Values

Information about the actual feature values is captured by the *FeatureValues* element. This element contains a sequence of the *FeatureValue* sub-elements. Here, we distinguish three types of feature values based on the data type of the feature:

- *Primitive feature values* shall be used for features of all “primitive” data types; specifically for integer-based features, floating-point based features, Boolean-based features, String-based features, classifications of objects and associations among objects.
- *Composite images* shall be used for composite image features.
- *Individual images* shall be used for individual image features.

There may be features defined as described in section 4.6.2 without associating any values with these features. These features will typically be skipped (omitted) by software processing the ICEFormat.

Depending on the type, these feature values shall be described by the *Primitive*, *CompositeImage*, and *IndividualImage* elements as follows:

a) **Primitive feature values**

Value of integer-based features, floating-point based features, Boolean-based features, String-based features, classifications of objects and associations among objects shall be described by the *Primitive* element with the following sub-elements:

— *FeatureID*: A sequence of one or more *FeatureID* elements shall identify feature(s) with values stored in a single binary file. Values of more than one feature may be captured in a single file as long as String-based features are kept separate from other features of primitive data types. See section 6 for details on how to store feature values in external files.

— *URL*: a Uniform Resource Locators (URL) of the file with feature values corresponding to the referenced features; the URL shall use the *file* schema and shall be relative to the location of the ICEFormat Data Directory XML file.

Examples:

```
<FeatureValue>
  <Primitive>
    <!-- Both floating points and therefore values may be combined -->
    <FeatureID>F002a</FeatureID>
    <FeatureID>F002b</FeatureID>
    <URL>file://FeatureValues/BinaryData1.bin</URL>
  </Primitive>
</FeatureValue>
```

```
<FeatureValue>
  <Primitive>
    <FeatureID>F006a</FeatureID>
    <FeatureID>F006b</FeatureID>
    <URL>file://FeatureValues/ObjectNames.xml</URL>
  </Primitive>
</FeatureValue>
```

b) **Composite images**

Values of features that are defined as visual representation of objects created by a specific mask applied to a particular composite image shall be described by the *CompositeImage* element. This element shall contain a sequence of one or more *FeatureID* sub-elements listing related features. In this case, there shall be no additional elements since the actual feature values (images of the objects) are obtained by inspecting the feature definition in the appropriate *InfoCompositeImage* element, extracting the image and mask identifiers, locating the image and the mask and applying the mask on the image. The presence of the *CompositeImage* elements among the feature values is for consistency purposes only, i.e., all feature values are explicitly listed even though there is no significant information in the case of composite images.

Example:

```
<FeatureValue>
  <CompositeImage>
    <FeatureID>F004a</FeatureID>
    <FeatureID>F004b</FeatureID>
  </CompositeImage>
</FeatureValue>
```

c) **Individual images**

Values of features that are defined as individual images of objects shall be described by the *IndividualImage* element with the following sub-elements:

- *FeatureID*: A reference to a (single) related feature. If there are multiple individual image features in the data set (e.g., images captured by different channels), their values shall be listed in separate *FeatureValue* elements.
- *URL*: a sequence of URL elements pointing to images of the objects. There shall be *n URL* elements where *n* corresponds to the number of objects as specified in the data set metadata (section 4.6.1). The order of the URL elements shall keep the same order of objects as any feature values files. Each URL element shall contain the following attributes:
  - *url* (required): The actual Uniform Resource Locators (URL) of the file that contains the image. The URL shall use the *file* schema and shall be relative to the location of the ICEFormat Data Directory XML file. This will be the only attribute if the file format captures a single image only. Additional attributes as detailed below are used to reference specific images within multi-frame image file formats, such as multi-page TIFFs or various movie file formats.
  - *page* (optional): A page attribute shall be used as a one-based index (positive integer) to reference a single image within the file format used if that contains multiple images. For example, the page attribute shall be used to specify a page number in a multi-page TIFF, or a frame number in a movie file format. The first image (page, frame, etc.) within the file shall be referenced by setting the page attribute to 1; the second image shall be referenced by setting the page attribute to 2, etc.

Example:

```
<FeatureValue>
  <IndividualImage>
    <FeatureID>F005</FeatureID>
    <URL url="file://Images/Individual/Image00001.png" />
    <URL url="file://Images/Individual/Image00002.tiff" page="1" />
    <URL url="file://Images/Individual/Image00002.tiff" page="2" />
      … <!-- All other image file references --> …
    <URL>file://Images/Individual/Image07654.png</URL>
  </IndividualImage>
</FeatureValue>
```

4.7 Plate-based Data

As mentioned in section 4.2, the ICEFormat supports plate-based data. For a plate-based experiment(s), the *Plate* element shall be used as a sub-element of the *ICEFormat* element in the XML file (there may be multiple plates defined in the ICEFormat Data Directory). An example of a header of an ICEFormat Data Directory XML file with a single plate-based experiment follows:

```
<?xml version="1.0" encoding="utf-8"?>
<ICEFormat
    xmlns = "http://www.isac-net.org/std/ICEFormat/1.0/ice"
    xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation = "http://www.isac-net.org/std/ICEFormat/1.0/
                          http://flowcyt.sf.net/ice/ICE.xsd"
    version = "1.0">

  <ChannelDefinitions>
       … Definitions of channels (optional, section 4.3) …
  </ChannelDefinitions>

  <SegmentationDefinitions>
       … Definitions of segmentations (optional, section 4.4) …
  </SegmentationDefinitions>

  <FeatureDefinitions>
```

```
        … Definitions of features (optional, section 4.5) …
  </FeatureDefinitions>

  <Plate Id="P1"> <!-- The Id of the plate is optional; there may be multiple plates -->
        … Description of plate-based data …
  </Plate>

  <Sitemap>
        … In addition, a Sitemap may be included (optional, section 4.8)…
  </Sitemap>

</ICEFormat>
```

The *Plate* element shall contain a definition of the plate layout (section 4.7.1) in the *Layout* element and a sequence of *well* elements (section 4.7.2), one for each (non-empty) well of the plate.

4.7.1 Plate Layout Definition

There are two options to define a plate layout, one can either select one of the standard plate layouts or define a custom plate.

a) **Standard plate layout**

A standard plate layout shall be specified by the *Standard* element with one of the supported layout types. One of the following values shall be used: *6 well plate*, *12 well plate*, *24 well plate*, *48 well plate*, *96 well plate*, *384 well plate*, or *1536 well plate*.

Example:

```
<Layout>
   <Standard>96 well plate</Standard>
</Layout>
```

b) **Custom plate layout**

A custom plate layout shall be specified by the *Custom* element with two sub-elements: the *Rows* element and the *Columns* element. The *Rows* and *Columns* elements shall include a positive integer value stating the number of rows and columns of the plate, respectively.

Example:

```
<Layout>
  <Custom>
    <Rows>2</Rows>
    <Columns>2</Columns>
  </Custom>
</Layout>
```

4.7.2 Wells

Each non-empty well of the plate shall be described by a *Well* element with the following sub-elements.

- *RowID*: An identifier of the row of a plate where the particular well is located, such as A, B, C, etc. A common row identifier should be used. The maximum of distinct row identifiers shall not exceed the number of rows on the particular plate, i.e., either as specified by the *Rows* element in custom plates or as present on the particular standard plate type. There may be less distinct row identifiers if some of the rows have been left empty or if data from these rows is not present in the particular ICEFormat structure.
- *ColumnsID*: An identifier of the column of a plate where the particular well is located, such as 01, 02, 03, etc. A common column identifier should be used. The maximum of distinct column

identifiers shall not exceed the number of columns on the particular plate, i.e., either as specified by the *Columns* element in custom plates or as present on the particular standard plate type. There may be less distinct column identifiers if some of the columns have been left empty or if data from these columns is not present in the particular ICEFormat structure.

— *DataSet*: Description of the data from the well; the format of the *DataSet* element shall correspond to the format of the *DataSet* element as described in section 4.6. There may be multiple *DataSet* elements in a well (e.g., these data sets may correspond to different sites).

The combination of the *RowID* value and the *ColumnID* value shall represent a unique identifier of a well, i.e., there shall not be multiple *Well* elements containing the same row and column identifiers.

4.7.3 Example of a Plate-based ICEFormat Data Directory

Below is an example of an ICEFormat Data Directory XML file for plate-based data.

```
<Plate Id="P1"> <!-- The Id of the plate is optional -->

  <Layout><Standard>96 well plate</Standard></Layout>

  <Well> <!-- Well A01 -->
    <RowID>A</RowID>
    <ColumnID>01</ColumnID>

    <DataSet>

      <MetaData> <!-- Only 2 objects to keep the example compact -->
        <NumberOfObjects>2</NumberOfObjects>
      </MetaData>

      <FeatureValues>
        … Description of the feature values for dataset in well A01 …
      </FeatureValues>

    </DataSet>

  </Well>

  <Well> <!-- Well A02 -->
    <RowID>A</RowID>
    <ColumnID>02</ColumnID>
    … Description of the dataset in well A02 …
  </Well>

  … Description of other non-empty wells from the 96 well plate …
</Plate>
```

4.8 Site Map

If the *SiteRef* attribute has been used for the description of data sets in the ICEFormat Data Directory XML file (see section 4.6) then a *Sitemap* element may be used as a sub-element of the *ICEFormat* element to describe locations of various sites. There may only be one *Sitemap* element and it shall define positions of all the sites referenced from all the *SiteRef* attributes within the XML. Please note that the same site may be referenced multiple times, e.g., if each well on a plate uses the same sites. In this case, the position of these sites may only be defined once but they may be referenced from several wells. Please note also that there may be no *Sitemap* defined although datasets are referencing sites.

This specification supports two different ways of how site locations may be expressed, using a grid-based site map (section 4.8.1) or using a position-based site map (section 4.8.2).

4.8.1 Grid-based Site Map

In a grid-based system, a grid is specified by a number of rows and columns. Each site is then assigned a specific row and column depending on the location of this site.

If a grid-based site map is used then the *Grid* element shall be placed as a sub-element of the *Sitemap* element. The *Grid* element shall contain two additional elements, *Rows* and *Columns*, whose values specify the number of rows and the number of columns in the grid, respectively. After these elements, there shall be a sequence of one or more *Site* elements, each of these containing three attributes: *ID*, *Row*, and *Column*.

The *ID* attribute serves as the identification of a site and shall therefore be unique among all the sites in the site map. The value of this attribute is used to reference a particular site from a data set using the *SiteRef* attribute as mentioned in section 4.6. The position of all sites referenced from ICE format data sets should be specified using a site map.

The *Row* attribute is used to specify the row number of the site. This number shall be greater or equal to 1 and not greater than the number of rows on the grid (i.e., as specified by the *Rows* attribute mentioned above). Rows are numbered from top to bottom (i.e. row number 1 is the top most row).

The *Column* attribute is used to specify the column number of the site. This number shall be greater or equal to 1 and not greater than the number of columns on the grid (i.e., as specified by the *Columns* attribute mentioned above). Columns are numbered from left to right (i.e. columns number 1 is the columns on the left side of the grid).

An example of a grid-based site map follows.

```
<Sitemap>
  <Grid>
    <Rows>2</Rows>
    <Columns>2</Columns>
    <Site ID="s1" Row="1" Column="1" />
    <Site ID="s2" Row="2" Column="1" />
  </Grid>
</Sitemap>
```

4.8.2 Position-based Site Map

In a position-based system, the size of the system is specified by height and width expressed either in pixels (i.e., virtual pixels, size relative to the size of captured images), or one of the suitable units of length according to the International System of Units (SI) may be used. For the purpose of this specification, the following SI units may be used:

• m	metre (the base SI unit of length)
• dm	decimetre (10^{-1} metre)
• cm	centimetre (10^{-2} metre)
• mm	millimetre (10^{-3} metre)
• um	micrometre (10^{-6} metre)
• nm	nanometre (10^{-9} metre)
• pm	picometre (10^{-12} metre)

If a position-based site map is used then the *Position* element shall be placed as a sub-element of the *Sitemap* element. The *Position* element should contain three additional elements, *Units*, *Width* and *Height*. The *Units* element shall contain one of the following unit abbreviations: *pixel*, *m*, *dm*, *cm*, *mm*, *um*, *nm*, *pm*, *fm*, *am*, or *zm* with their meaning described above. The *Width* and *Height* elements shall contain the size of the position-based system in the stated unit (a positive floating point number may be used unless the *pixel* unit is chosen, in which case a positive integer value shall be used).

After these elements, there shall be a sequence of one or more *Site* elements, each of these described as stated below.

Each Site element shall contain an *ID* attribute, which serves as the identification of a site and shall therefore be unique among all the sites in the site map. The value of this attribute is used to reference a particular site from a data set using the *SiteRef* attribute as mentioned in section 4.6. The position of all sites referenced from ICE format data sets should be specified using a site map.

Each Site element may have up to four additional attributes: *Left*, *Right*, *Bottom*, and *Top*. These attributes specify the position of the site (the closest edge) relative to the left, right, bottom and top of the system, respectively. The unit is assumed the same as stated above in the *Units* element. A floating point value may be used unless the *pixel* unit has been chosen, in which case a positive integer value shall be used. At least one of *Left* or *Right* and one of *Bottom* or *Top* should be specified for each side.

An example of a position -based site map follows.

```
<Sitemap>
  <Position>
    <Units>pixel</Units>
    <Width>10000</Width>
    <Height>10000</Height>
    <Site ID="s1" Left="234" Bottom="567" />
    <Site ID="s2" Left="1234" Bottom="1567" Right="2234" Top="2567" />
  </Position>
</Sitemap>
```

5. Binary Mask

5.1 General

A mask specifies the position of one or more objects in a composite image. A mask is stored in a binary file as a sequence of 1, 2, or 4 byte little-endian unsigned integer values, each of these values corresponding to a single pixel of the composite image. The number of bytes allocated for each value shall correspond to the bit depth specified for the particular mask (see section 4.6.4) as follows: 1 byte = 8 bit depth, 2 bytes = 16 bit depth, and 4 bytes = 32 bit depth.

5.2 Objects and Background in the Mask

Each positive integer value specifies the object number present in the particular spot (i.e, pixel) in the image. The first object is encoded by the number specified by the first mask object number (the *MaskObjectNumber* element, see section 4.6.4), the second one by the second mask object number, etc. If mask object numbers are not explicitly specified as part of the mask definition in the ICEFormat Data Directory XML file then they are assumed to go sequentially from 1 to *n*, where *n* is the number of objects specified for the particular dataset as described in section 4.6.1.

A zero value (0) in the mask binary file notifies that no object was identified in the particular spot (pixel) in the image (i.e., zero is value reserved for background). Please note that this representation does not allow for encoding of multiple overlapping objects by a single mask. Multiple mask definition files may be used if overlapping objects need to be defined.

5.3 Correspondence of Masks and Images

The integer values in the sequence of mask definition correspond to pixels in the image starting in the top left corner, going right to the top right corner, and proceeding with additional rows from left to right until the last row in the image is reached. Specifically, let *w* be the width of the image (as well as the width of the mask) and let *h* be the height of the image (as well as the height of the mask). Then the first integer value in the sequence corresponds to the pixel of the top left corner in the image file. The second integer value corresponds to the pixel in first row and second column, etc. The w^{th} integer value in the sequence corresponds to the pixel in the top right corner, the $(w*(h-1)+1)^{th}$ value corresponds to the pixel in bottom left corner and the $(w*h)^{th}$ value to the bottom right corner. Consequently, let b be the bit depth of the mask specified in the mask definition (section 4.6.4), then the size of the binary file with the mask definition in bytes shall correspond to $(w*h*b) / 8$.

6. Feature Values

The actual feature values shall not be included in the ICEFormat Data Directory XML file. Instead, they shall be stored in additional files. The file format of the file depends on the data type of the feature as described further.

6.1 Binary Feature Values of Primitive Data Types

Values of integer-based, floating-point-based, and Boolean-based features, object classifications as well as object associations shall be stored in binary files. The values shall be little-endian encoded. Each value occupies the number of bits as specified by the bit depth of the feature definition in the ICEFormat Data Directory XML file (section 4.6.2). Stored bytes shall be interpreted as follows:

- In the case of integer-based features, bytes shall be interpreted as signed integer values.
- In the case of floating-point-based features, bytes shall be interpreted as floating point values in accordance with the IEEE 754 specification [7].
- In case of Boolean-based features, bytes shall be interpreted as signed integer values and these shall be mapped to Boolean values as follows: 0 as *false*, 1 as *true*, any other value as *unknown*.
- In case of object classifications, bytes shall be interpreted as unsigned integer values and these shall be mapped to classes defined in the ICEFormat Data Directory XML file (section 4.5.7). Specifically, the value zero (0) represents an unknown class; values 1, 2, 3, …, k represent classes in the order as defined in the ICEFormat Data Directory XML file. The highest value encoding classes for a particular feature shall be less or equal to the number of classes defined in the ICEFormat Data Directory XML file for the appropriate feature.
- In case of object associations, values shall be interpreted as integers (see above) and objects with the same value of the same feature (matched by feature ID) shall be associated. See also example in section 4.5.8 for further details.

A particular feature values file may be used to capture feature values of a single feature or to capture values of multiple binary encoded features.

If a particular feature value file is used to capture feature values of a single feature, the values simply follow one after another, one value for each object. For this purpose, the objects can be arbitrary ordered as long as the same order is maintained across values for all features. Consequently, the size of the binary file shall equal to

$(n * b / 8)$ bytes

where n is the number of objects described in the ICEFormat Data Directory (section 4.6.1) and b is the bit depth specified for the feature according to section 4.6.2.

If a particular feature value file is used to capture feature values of k features then the values of feature number 1 are included first, one value for each object. Values of the second feature follow after all values of the first feature, etc. For this purpose, the order of features is defined by the order in which their identifiers are listed under the corresponding *FeatureValue*, *Primitive* elements (see section 4.6.5 a). Formally, let o_1, o_2, …, o_n be the order of all n object (an arbitrary order as long as it is maintained across all feature values), let f_1, f_2, …, f_k be the order of features as listed according to section 4.6.5 a) with values joined in a single binary file as described in the ICEFormat Data Directory, section 4.6.5, then values in the binary file shall correspond to $f_1(o_1)$, $f_1(o_2)$, …, $f_1(o_n)$, $f_2(o_1)$, $f_2(o_2)$, …, $f_2(o_n)$, … , $f_k(o_1)$, $f_k(o_2)$, …, $f_k(o_n)$ where $f_i(o_j)$ is the value of feature f_i applied on object o_j. Consequently, the size of the binary file shall equal to

$$(n * (b_1 + b_2 + \ldots + b_k) / 8) \text{ bytes}$$

where n is the number of objects described in the ICEFormat Data Directory (section 4.6.1) and b_i (i from 1..k) is the bit depth of feature f_i, specified according to section 4.6.2.

6.2 Images

For the purpose of this specification, an image of an object is considered a feature value of that object. Any suitable established open image file format may be used to store the actual image data files. Image file formats that are proprietary, vendor-specific, that do not have a publicly available specification, or that require the acquisition of a license in order to decode the image shall not be used. In addition, image formats that are highly flexible (such as TIFF [2]) should be used in the simplest configuration possible in order to facilitate interoperability between different software packages.

Both, individual images of a single object and composite images of multiple objects may be used. In case of individual images, the feature definition (sections 4.5.5) and the feature value definition (section 4.6.5 c) in the ICEFormat Data Directory shall be used to store and retrieve the links between objects and their visual representation. In case of composite images, information from the feature definition (sections 4.5.4), the composite image definition (section 4.6.3) mask definition (section 4.6.4), the feature value definition (section 4.6.5 b) and the binary mask (section 5) shall be combined and the mask applied to the image to retrieve visual representation of particular objects.

6.3 String-based Features Values

Values of features that are strings of characters shall be stored in XML files valid according to the ICEStrValues.xsd XML schema. The *StringFeatureValues* element shall be used as the root element of the XML file. An example of a header of a string-based feature value XML file follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<StringFeatureValues
  xmlns = "http://www.isac-net.org/std/ICEFormat/1.0/iceStrValues"
  xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation = "http://www.isac-net.org/std/ICEFormat/1.0/iceStrValues
                        http://flowcyt.sf.net/ice/ICEStrValues.xsd">
```

… Feature values for all objects …

```
</StringFeatureValues>
```

The *StringFeatureValues* element shall contain a sequence of *Feature* sub-elements. The *Feature* sub-elements shall correspond to features associated with this particular XML file by a feature value entry in the ICEFormat Data Directory XML file (section 4.6.5 a). Each *Feature* element shall contain a *FeatureID* sub-element and a sequence of *Value* sub-elements with the following semantics:

— *FeatureID*: An identifier of the feature; the identifier shall be present among feature identifiers associated with this XML file by a feature value entry in the ICEFormat Data Directory XML file (section 4.6.5 a). Vice versa, all features associated with this particular XML file by a feature value entry in the ICEFormat Data Directory XML file shall be present in this string feature values XML file.

— *Value*: A sequence of *Value* elements shall provide values of this feature for all objects. There shall be a *Value* element for each object. The number of *Value* elements shall correspond to the number of objects stated in the metadata section of the ICEFormat Data Directory XML file (section 4.6.1). The order of values shall be maintained across values for all features. International character sets are supported natively in XML (encoding attribute in the XML header). Characters that would collide with XML special characters, e.g., “<”, “>”, “&”, may be encoded using predefined XML entities, e.g., “&lt;”, “&gt;”, “&amp;”, “&quot;”, “&apos;”, or by entering the numeric character reference in one of the XML compatible forms.

An example of string-based feature values follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<StringFeatureValues … Header definition …>

  <Feature>
    <FeatureID>F006a</FeatureID>
    <Value>Value for object 1, Feature F006a</Value>
    <Value>Value for object 2, Feature F006a</Value>
        … Feature values of the F006a feature for all other objects …
  </Feature>

  <Feature>
        … Feature values of another feature for all objects …
  </Feature>

  … All other features and their values for all objects …

</StringFeatureValues>
```

Annex A

ICEFormat in ACS – Informative

Please note that this appendix is provided for informative purposes only. Use the Archival Cytometry Standard (ACS [1]) specification as a normative reference for the creation of ACS files.

For the purpose of ACS, the ICEFormat Data Directory file may be considered as the primary data file since it contains the necessary information to locate all the ICEFormat components. Therefore, in the simplest case with no revision history, neither digital signature required, the ICEFormat components can be bundled in an ACS container by:

a) Crating the *TOC1.xml* file in the root of the ICEFormat folder structure.
b) Zipping the ICEFormat folder structure and naming the resulting ZIP file with an *.acs* file extension.

The contents of the *TOC1.xml* file may be as simple as:

```
<?xml version="1.0" encoding="UTF-8"?>
<toc:TOC
  xmlns:toc          = "http://www.isac-net.org/std/ACS/1.0/toc/"
  xmlns:xsi          = "http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation = "http://www.isac-net.org/std/ACS/1.0/toc/
                        http://flowcyt.sf.net/acs/toc/TOC.v1.0.xsd">

    <toc:file toc:URI="file:///filename.ice" />

</toc:TOC>
```

where *filename.ice* stands for the name of the ICEFormat Data Directory file name. Certainly, additional attributes (such as the *mimeType* and *description*) as well as relations among files and other information may be included as permitted by the Archival Cytometry Standard (ACS [1]) specification.

Annex B

Bibliography – Informative

[1] Spidlen J, Moore W, Bray Chris, International Society for Advancement of Cytometry Data Standards Task Force, Brinkman R. Archival Cytometry Standard. http://flowcyt.sf.net/acs/latest.pdf 2010.

[2] Adobe Systems Inc. Tagged Image File Format (TIFF) Developer Resources. http://partners.adobe.com/public/developer/tiff 2010.

[3] World Wide Web Consortium (W3C). Extensible Markup Language (XML). http://www.w3.org/TR/REC-xml/.

[4] World Wide Web Consortium (W3C). XML Schema Part 0: Primer Second Edition. http://www.w3.org/TR/xmlschema-0/.

[5] World Wide Web Consortium (W3C). XML Schema Part 1: Structures Second Edition. http://www.w3.org/TR/xmlschema-1/.

[6] World Wide Web Consortium (W3C). XML Schema Part 2: Datatypes. http://www.w3.org/TR/xmlschema-2/.

[7] IEEE. IEEE 754: Standard for Binary Floating-Point Arithmetic. http://grouper.ieee.org/groups/754/.

[8] Bradner S., The Internet Engineering Task Force. Request for Comments: 2119 - Key words for use in RFCs to Indicate Requirement Levels. http://www.ietf.org/rfc/rfc2119.txt.

[9] W3C. W3C Recommendation - Namespaces in XML 1.0 (Second Edition). http://www.w3.org/TR/xml-names/.

Annex C

Revision History – Informative

Changes since ISAC Candidate Recommendation version 1.0 (June 1, 2011)

— **Added support for multi-frame image file formats:**
In the ICEFormat Data Directory, the URL element of an image has been extended to allow for an additional page attribute. This attribute shall be used to reference a single image within an image file format that captures multiple images (e.g., a multi-page TIFF or movie file formats). This extension has been applied to both composite (section 4.6.3) and individual images (section 4.6.5 c). The ICE.xsd XML schema and the text of the specification have been updated accordingly.

— **Updated version and date:**
The version has been changed to 1.1 and date to 2011-12-14. ICEFormat Data Directory files corresponding to this new version shall indicate so in the version attribute of the main ICEFormat element.