WILL

WILL Data Format Specification

version 1.0

Wacom Co., Ltd.

WILL Data Format Specification.

Abstract

This document describes WILL Data Format. WILL Data Format serves as data model for a digital ink representing ink entered with an electronic pen or stylus. The digital ink according to WILL Data Format can be serialized into a binary stream by several serialization schemes and stored in files in various types of file format. In order to provide examples, this document also describes a Data Encoding scheme and WILL File Format as informative specification.

Content

Abstract 2
1. Introduction 3
2. Normative Reference 3
3. Scope 4
4. Terms and definitions 4
5. Will Data Format 5
    5.1. Path 5
    5.1.1 Stroke Width 6
    5.1.2 Stroke Color 6
    5.1.3 Start Paramter 6
    5.1.4 End Paramter 6
    5.2. Segment (Catmull-Rom) 6
    5.2.1 Position 7
    5.3. Encoding and Decoding Path 8
    5.3.1 Encoding and Decoding of Position and Stroke Width 8
    5.3.2 Examples of Path properties 8
    5.3.3 List of Paths 9
Annex A. (Informative) Serialization 10
    A. 1 Serializing digital ink according to WILL Data Format 10
    A. 2 Example of Schema file (Protocol buffers) 10
Annex B. WILL File Format 11

1. Introduction

Wacom Ink Layer Language ("WILL") is a universal inking engine and ink layer framework that connects hardware, software, and applications and enables a high-quality digital pen and ink experience. WILL Data Format describes data model of a digital ink entered with digital pen or stylus. The digital ink according to WILL Data Format is extensible and can be associated with other types of data including metadata indicating time, geographical location, or certificate information.

The digital ink according to WILL Data Format can be serialized into a binary stream by several serialization schemes. Such binary streams can be stored in a file according to several types of file formats such as WILL File Format , SVG file or PDF file. The digital ink according to WILL Data Format can be edited after creation and can also be enriched with features like beautification and recognition. Compared to other proprietary inking solutions, digital ink according to WILL Data Format does not preseve all the input data but preserve binary data necessary to reproduce the digital ink. This results in small data packages, allowing for fast transmission, and easier integration with cloud services and metadata.

In order to provide reference implementation handling the digital ink according to WILL Data Format, WILL SDKs, which function to output, input, and render digital ink according to the WILL Data Format, are made available to partners and external developers.

2. Normative Reference

[InkML] Ink Markup Language (InkML)

http://www.w3.org/TR/InkML/

[SVG] Scalable Vector Graphics (SVG) 1.1 (Second Edition)

http://www.w3.org/TR/SVG/

[PDF] ISO 32000-1:2008 Document management -- Portable document format -- Part 1: PDF 1.7

https://www.iso.org/standard/51502.html

[OPC] ISO/IEC 29500-2:2008 Information technology -- Document description and processing languages -- Office Open XML File Formats -- Part 2: Open Packaging Conventions

http://www.iso.org/standard/51459.html

[RDF] RDF/XML Syntax Specification (Revised), D. Beckett, Editor, W3C Recommendation, 10 February 2004,

http://www.w3.org/TR/2004/REC-rdf-syntax-grammar-20040210/.

[ISO/IEC 10918] ISO/IEC. 1994. JPEG Standard for digital compression and encoding of continuous-tone still images.
https://www.iso.org/standard/18902.html

3. Scope

This document defines data model of digital ink according to WILL Data Format.

- In Section 5, this document provides the data model and encoding and decoding of digital ink according to WILL Data Format as mandatory part.

The following two Annexes provide specific examples of serializations and packaging. Specific serialization and packaging schemes are beyond the scope of this document.

- In Annex A, this document provides an example serialization scheme of digital ink according to WILL Data Format as informative specification.
- In Annex B, this documents provides an example file format (WILL File Format) packaging digital ink according to WILL Data Format.

4. Terms and definitions

For the purposes of this document, the following terms and definitions apply.

digital ink

Data representing ink entered with an electronic pen or stylus. Digital ink can include several Paths and other data e.g., metadata.

Path

Entity used to reproduce a path, trace, or trajectory of the electronic pen. Each instance of Path includes one or more Segments.

Segment

Entity used to reproduce a part (segment) of the Path. Each instance of Segment includes Positions.

Position

Property of the Segment. Positon includes two dimensional data (X, Y) which are used as a control point for a Spline curve algorithm (Cutmull-Rom curve).

WILL Data Format

Data model for describing the digital ink, which is part of WILL.

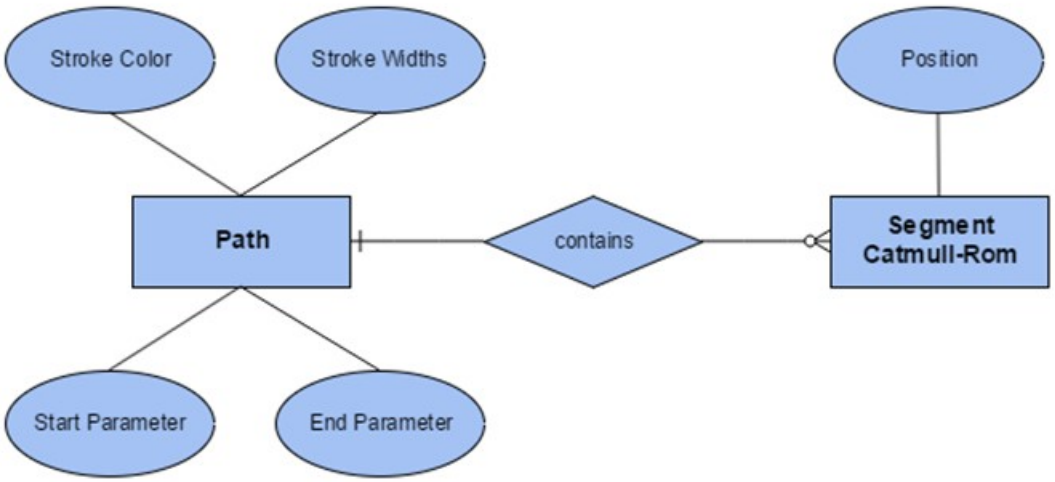
WILL File Format

A type of container file format designed to store the digital ink according to WILL Data Format.

5. Will Data Format

WILL Data Format serves as data model for a digital ink representing ink typically entered with an electronic pen. Figure 1 is an entity relationship diagram for the data model for digital ink according to WILL Data Format.

Figure 1 — Data model ER diagram



5.1. Path

Any digital ink according to InkML, SVG, or WILL Data Format primarily represents path (or trace, trajectory) of the electronic pen. In WILL Data Format, data representing the path is called **Path**. **Path** corresponds to 'path' element in SVG and <trace> element in InkML. In other type of data format, **Path** may be also called 'stroke' or 'stroke data'.

As shown in Figure 1, **Path** entity is the core building block in digital ink. Every stroke or trace is represented by an instance of **Path**.

Path in its simplest form **contains** one or more **Segment**s (Catmull-Rom), which includes a series of positional points**, Position**s. But, to achieve a high quality digital ink experience, **Path** includes **Stroke Width** and **Stroke Color** as its properties. In addition to those properties, digital ink in WILL Data Format includes **Start Parameter** and **End Parameter** as **Path** entity's properties.

5.1.1 Stroke Width

The **Stroke width** contains a list of positive values used to calculate the width of the stroke at different points along the path. If the list contains a single value, then the path has a uniform width. If the list contains less values than the number of the control points used to define the path, the last value is repeated multiple times to match the number of control points.

5.1.2 Stroke Color

The **stroke Color** contains a color value applied to the instance of Path.

5.1.3 Start Parameter

This parameter specifies the t-value ([0, 1]) of the first instance of **Segment** (Catmull-Rom) of multiple instances of **Segment**s at which display or rendering of the instance of **Path** starts. Default value is 0. Please be noted that this Start Parameter exists only in first instance of instances of **Segment**s in one instance of **Path**.

5.1.4 End Parameter

This parameter specifies the t-value ([0, 1]) of the last instance of **Segment** of multiple instances of **Segment**s at which display or rendering of the instance of **Path** starts. Default value is 1. Please be noted that Start Parameter exists only in last instance of instances of Segments in one instance of **Path**.

5.2. **Segment** (Catmull-Rom)

As shown in Figure 1, one instance of **Path** includes one or more instances of **Segment**s. Geometry (or shape of curve) of each instance of **Segment** is reproduced Catmull-Rom splines with four **Positions**.

5.2.1 Position

Position includes two-dimensional data (X , Y), which is used as control points for Catmull-Rom spline curve algorithm. Geometries (shapes) of multiple **Segments** based on Catmull-Rom spline are represented with a series of the two-dimensional data, **Position**s (P_0, P_1, P_2, ... $P_{n)}$. Figure 2 illustrates the geometry of the first instance of Segment . The geometry of the first instance of Segment is defined by four Positions(P_0, P_1, P_2, P_3). Figure 3 illustrates the geometry of the i-th instance of Segment.

The geometry of the i-th instance of **Segment** is determined by four **Position**s (P_{i-1}, P_i, P_{i+1}, P_{i+2}). The tangent at each Positon is calculated using the previous and next Position on the spline.

Figure 2 Geometry of the first instance of **Segment**

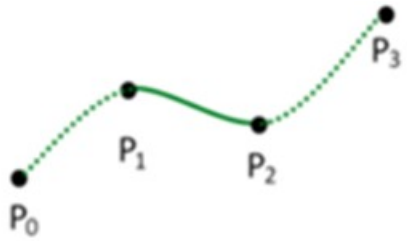
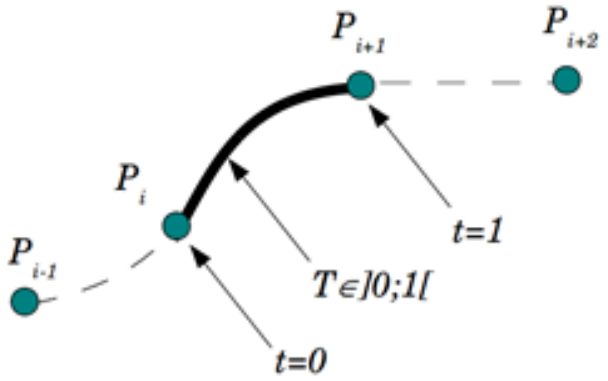


Figure 3 Geometry of the i-th instance of **Segment**



5.3. Encoding and Decoding Path

5.3.1 Encoding and Decoding of Position and Stroke Width

Delta encoding/decoding is applied to values of all **Position**s within an instance of **Path**, as well as the corresponding value(s) of **Stroke Width**. To enable further optimization, these values are represented in integer form (as fixed-point numbers) before the delta encoding procedure.

5.3.2 Examples of Path properties

The followings are examples of values of properties of an instance of **Path.** This example is based on the schema illustrated in Annex A:

- For **Start Parameter**, a value in the interval [0, 1], (default 0.0) is to be specified for 'startParamter' as float value in the illustrated schema.
- For **End Parameter**, a value in the interval [0, 1], (default 1.0) is to be specified for 'endParameter' as float value.
- The value of 'decimalPrecision' indicates the number of digits after the radix point values stored in point (for **Positions**) and strokeWidth (for ***Stroke Width***) properties.
- A delta-encoded list of the coordinate values (*x, y)* of **Positions** for all **Segments** in a **Path** is to be specified as 'points' by using repeated sint 32 as their data type.

- A delta-encoded list of width values for **Stroke Width** is to be specified as 'strokeWidths' by using repeated sint 32.
- A delta-encoded list of color values for **Stroke Color** is to be specified as 'strokeColors' by using repeated sint 32. This list may contain a delta-encoded list of color values in the *RGBA* format.

5.3.3 List of Paths

The *Path* messages that define all of the encoded **Path**s in a drawing canvas are represented as a length-delimited list. Table 1 identifies the list including a plurality of instances of Paths.

Table 1. List of Paths

Description	Bytes sequence
128 bit varint	*Path 1* message length
Bytes	*Path 1* message bytes
128 bit varint	*Path 2* message length
Bytes	*Path 2* message bytes
...	...
128 bit varint	*Path N* message length
Bytes	*Path N* message bytes

Annex A. (Informative) Serialization

A. 1 Serializing digital ink according to WILL Data Format

In order to serialize the Path in an efficient way, the list of Path is serialized as binary data. As an example, developers may use 'Protocol buffers' provided by Google. Protocol buffers are Google's language-neutral, platform-neutral, extensible mechanism for serializing structured data. The method involves an interface description language[1] that describes the structure of some data and a program that generates source code from that description for generating or parsing a stream of bytes that represents the structured data.

In the following, the specification of the Path data utilizing the interface description language is discussed.

A. 2 Example of Schema file (Protocol buffers)

Values or instances of Start Parameters, **End Parameters**, **Points,** Stroke Width, and other properties of **Path** may be encoded in or decoded from a binary stream form by using any serialization/deserialization tool such as *Google protocol buffers*. Figure 4 is an example schema file.

Figure A1- example schema file

```
package WacomInkFormat;

option optimize_for = LITE_RUNTIME;

message Path {
   optional float  startParameter   = 1 [default = 0];
   optional float  endParameter     = 2 [default = 1];
   optional uint32 decimalPrecision = 3 [default = 2];
   repeated sint32 points          = 4 [packed = true];
   repeated sint32 strokeWidths     = 5 [packed = true];
   repeated sint32 strokeColor      = 6 [packed = true];
}
```

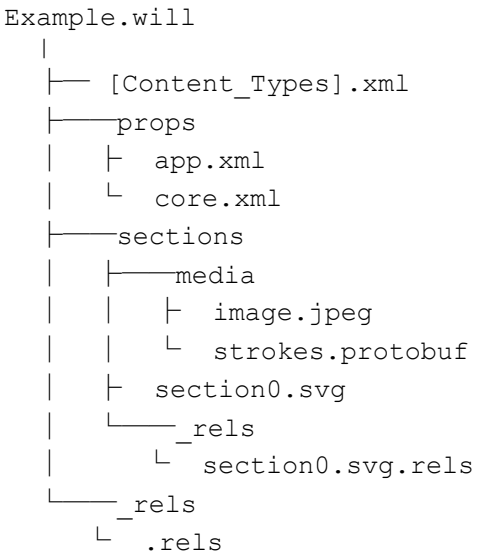
[1] Google Protocol Buffer interface description language. https://developers.google.com/protocol-buffers/docs/proto : Last accessed 05/31/2017.

Annex B. WILL File Format (.will)

The digital ink (binary stream) according to WILL Data Format can be embedded in several type of files such as PDF, SVG, or a file according to WILL File Format. WILL File Format is a type of container file which can include digital ink according to WILL Data Format and other files such as SVG or JPEG. The digital ink according to WILL Data Format can be packaged using WILL File Format following the Open Packaging Convention ("OPC") ISO standard. The digital ink stored in a file according to WIL File Format complies the Scalable Vector Graphics (SVG) open standard. WILL File Format utilizes OPC formatted ZIP archive. Figure B1 and B2 and Table B1 show an example of a typical files packaged in a file according to WILL File Format[2] .

The strokes 387757975.protobuf corresponds to binary stream serialized according to Annex A.

Figure B1- example .will file structure



---

[2] The example is based on the technical implementation discussed in Section

Annex A. (Informative) Serialization

A. 1 Serializing digital ink according to WILL Data Format.

Table B1- example .will file structure and description

Section	Description
[Content_Types].xml	Contains information about the content types in the WILL file.
/props/app.xml	Contains information about the application that originally created the file.
/props/core.xml	Contains general information and metadata about the file.
/sections/media/image.jpeg	Contains raster data that is part of the multi-media content. JPEGs are encoded with JFIF standard 1.01 **[ISO/IEC 10918]**.
/sections/media/strokes.protobuf	Contains all strokes (paths) encoded using serialization scheme according to Annex A.
/sections/section.svg	Contains the multi-media content encoded using SVG.

Figure B2- examples of files packaged

Content_Types.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<Types xmlns="http://schemas.openxmlformats.org/package/2006/content-types">
  <Default Extension="rels" ContentType="application/vnd.openxmlformats-package.relationships+xml"/>
  <Default Extension="svg"  ContentType="image/svg+xml"/>
  <Default Extension="jpg"  ContentType="image/jpeg"/>
  <Default Extension="jpeg" ContentType="image/jpeg"/>
  <Default Extension="png"  ContentType="image/png"/>
  <Default Extension="protobuf" ContentType="application/vnd.willfileformat.path+protobuf"/>
  <Override PartName="/props/core.xml" ContentType="application/vnd.openxmlformats-package.core-
properties+xml"/>
  <Override PartName="/props/app.xml" ContentType="application/vnd.willfileformat.extended-
properties+xml"/>
  <Override PartName="/style/paints.protobuf" ContentType="application/vnd.willfileformat.paint+protobuf"/>
</Types>
```

props/app.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<Properties
   xmlns="http://schemas.willfileformat.org/2015/relationships/extended-properties">
   <Application>Bamboo Spark</Application>
   <AppVersion>1.0</AppVersion>
</Properties>
```

props/core.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<coreProperties
   xmlns="http://schemas.openxmlformats.org/package/2006/relationships/metadata/core-properties"
   xmlns:dc="http://purl.org/dc/elements/1.1/"
   xmlns:dcterms="http://purl.org/dc/terms/"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <dcterms:created xsi:type="dcterms:W3CDTF">2016-05-12T16:46:34Z</dcterms:created>
   <dc:title>WCM0074</dc:title>
</coreProperties>
```

/sections/media/image.jpeg

This section inludes image data.

/sections/media/strokes.protobuf

This section includes binary encoded ink data.

/sections/section.svg

```
<?xml version="1.0" encoding="UTF-8"?>
<svg xmlns="http://www.w3.org/2000/svg" xmlns:r="http://schemas.willfileformat.org/2015/relationships"
width="328.0" height="439.0">
<defs>
<pattern id="background" patternUnits="userSpaceOnUse" width="328.0" height="439.0"><image r:id="image1"
x="0.0" y="0.0" preserveAspectRatio="xMidYMid slice"/>                                    </pattern>
</defs>
<rect id="willfileformat.background.rect" x="0" y="0" width="328.0" height="0.0" fill="url(#background)"/><g
r:id="image0"/>
</svg>
```

_rels/rels

```
<?xml version="1.0" encoding="UTF-8"?>
<Relationships
   xmlns="http://schemas.openxmlformats.org/package/2006/relationships">
   <Relationship Id="core-properties"
Type="http://schemas.openxmlformats.org/package/2006/relationships/metadata/core-properties"
Target="/props/core.xml"/>
   <Relationship Id="extended-properties" Type="http://schemas.willfileformat.org/2015/relationships/extended-
properties" Target="/props/app.xml"/>
   <Relationship Id="section0" Type="http://schemas.willfileformat.org/2015/relationships/section"
Target="/sections/section0.svg"/>
</Relationships>
```