WILL File Format

WILL Overview

WILL data primarily represents pen strokes as digital ink. A pen stroke in its simplest form consists of a series of positional points but to achieve a high quality digital ink experience additional information is needed including stroke colour, width and appearance. The WILL SDK employs sophisticated techniques to record and render stroke data, applying smoothing algorithms to enhance the experience.

The WILL data file is packaged using the Open Packaging Convention (OPC) ISO standard. OPC describes conventions for the use of XML, ZIP and other openly available technologies to organize the content and resources within a package.

The data stored in a WILL file is based on the Scalable Vector Graphics (SVG) open standard with extensions applied to cater for WILL requirements. SVG is the language based on XML for describing two-dimensional vector and mixed vector/raster graphics.

The SVG data is encoded in binary form using Google protocol buffers. The technique provides cross platform compatibility with rapid encoding and decoding. Protocol buffers are Google's open standard for serializing structured data using an extensible mechanism which is both language and platform neutral.

WILL File Format

The WILL file format uses an OPC formatted ZIP archive as a container. A typical WILL file expands as follows:

```
Example.will
  |
   [Content_Types].xml
  props
      app.xml
      core.xml
  sections
    media
        image2049281480.jpeg
        strokes387757975.protobuf
      section0.svg
    _rels
         section0.svg.rels
  _rels
       .rels
```

Section	Description
/props/app.xml	contains information about the application that originally created the file.
/props/core.xml	contains general information and meta-data about the file.
/sections/media/image.jpeg	contains raster data that is part from the multi-media content.
/sections/media/strokes.protobuf	contains all strokes (paths) encoded using WILL encoding scheme (based on Google Protocol Buffers).
/sections/section.svg	contains the multi-media content encoded using SVG.

Examples of the section contents follow:

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

image data

/sections/media/strokes.protobuf

Binary protobuf 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

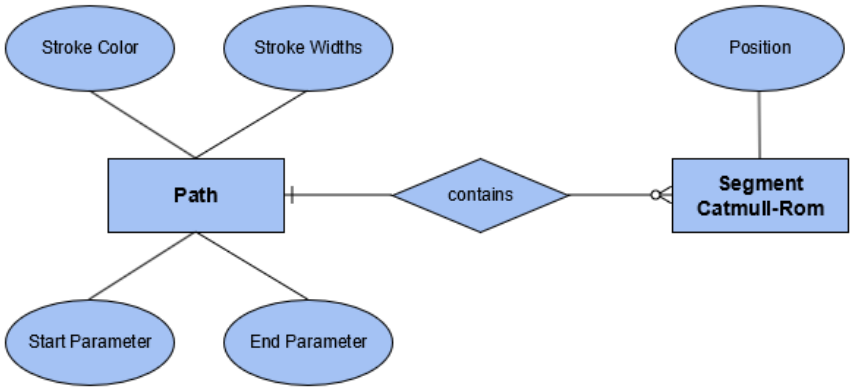
```
<?xml version="1.0" encoding="UTF-8"?>
<Relationships
    xmlns="http://schemas.openxmlformats.org/package/2006/relationships">
    <Relationship Id="core-properties" Type="http://schemas.openxmlformats.org/package/2006/relationships
/metadata/core-properties" Target="/props/core.xml"/>
    <Relationship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

Will Data Format

Path Model

The Path entity is the fundamental building block in digital ink. Every stroke is represented by a Path instance.

The conceptual model used in the WILL data format is described using an entity-relationship model:

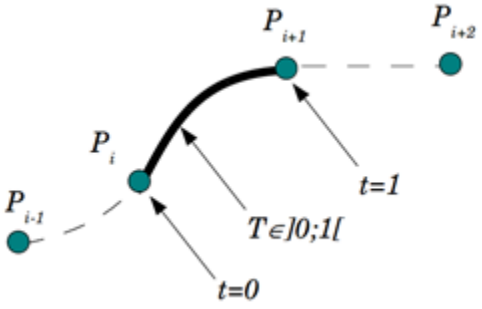


Path style

- *Stroke Colour* - the colour used to render the stroke
- *Stroke Widths* - 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 constant stroke width. If the list contains less values that the number of the control points used to define the path, the last value is taken multiple time to match the number of control points.

Path geometry

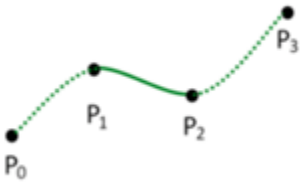
Path geometry is based on Catmull-Rom splines formulated such that the tangent at each point is calculated using the previous and next point on the spline:



- *Start Parameter* - Specifies the t-value of the first segment from which the path should start.
- *End Parameter* - Specifies the t-value of the last segment in which the path should end.

The geometry of the path is defined by a collection of path segments represented by Catmull-Rom curves:

A Catmull-Rom curve is represented as a collection of two-dimensional positions P_0, P_1, P_2, ... P_n , where each four points P_{i-1}, P_i, P_{i+1}, P_{i+2} define one segment using Catmull-Rom interpolation method to define the curve between P_i and P_{i+1}.



Path Data

Data encoding

The technique of delta encoding is applied to all control point coordinates within a stroke, as well as the corresponding width values for strokes with variable width. To enable further optimization, these values are represented in integer form (as fixed-point numbers) before the delta encoding procedure.

Coordinates, width values, and other properties for every stroke are encoded in a binary form using *Google protocol buffers*. This allows rapid encoding and decoding, and the resulting binary code is highly-portable and compact because the techniques used for representing repeated stroke values greatly improve the efficiency of the varint technique used.

Values are encoded according to the following schema:

```
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];
}
```

Data properties

Every *Path* message in the schema represents a single stroke and its properties as described below:

- A value in the interval [0, 1] that specifies the exact beginning of the path:

```
optional float startParameter = 1 [default = 0];
```

- A value in the interval [0, 1] that specifies the exact ending of the path:

```
optional float endParameter = 2 [default = 1];
```

- The number of digits after the radix point for values stored in *data* and *strokeWidth* properties:

```
optional uint32 decimalPrecision = 3 [default = 2];
```

- An encoded list of the *x* and *y* values of all control points for a particular stroke:

```
repeated sint32 data = 4 [packed = true];
```

- A delta encoded list of width values for all control points for a stroke with a variable stroke width:

```
repeated sint32 strokeWidth = 5 [packed = true];
```

For a stroke of constant width, this list contains a single value:

- A delta encoded list of color values in *RGBA* format:

```
repeated sint32 strokeColor = 6 [packed = true];
```

For a stroke of constant color, this list contains a single value. Most commonly, any variation in stroke color is in the opacity (the alpha channel of the *RGBA* format). In other words, the same *RGB* values apply throughout, but the *A* value changes.

Data list

The *Path* messages that define all of the encoded strokes is represented as a length-delimited list as follows:

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