Sutton-Slevinski Collaboration S. Slevinski
Internet-Draft Center for Sutton Movement Writing
Intended status: Informational June 24, 2017
Expires: December 26, 2017

Formal SignWriting
draft-slevinski-formal-signwriting-02

Abstract

Sutton SignWriting is the universal and complete solution for written sign language. It has been applied by a wide and deep international community of sign language users. Sutton SignWriting is an international standard for writing sign languages by hand or with computers. From education to research, from entertainment to religion, SignWriting has proven useful because people are using it to write signed languages.

Formal SignWriting (FSW) is a faithful character-encoding of Sutton SignWriting based on 2-dimensional mathematics. FSW defines a formal language for written sign languages where any sign of any sign language can be written as a string of ASCII or Unicode characters. The mathematical names are explained with tokens and regular expression patterns. Signs are written in a spatial SignBox, where each symbol is positioned with a 2-dimension Cartesian coordinate. For sorting, each sign can have an optional temporal sequence of symbols that is outside of the SignBox. To create sentences, signs are written sequentially, interspersed with punctuation symbols.

The styling string of Formal SignWriting uses a lite markup to define a variety of styling options. The entire sign can be customized for padding, coloring, and size. Individual symbols within a sign can be customized for coloring and size. For SVG, class names and IDs can be defined.

The query language of Formal SignWriting uses a lite markup, similar to FSW, to define a variety of searching possibilities. The spatial SignBox can be searched for symbols or ranges of symbols. For each symbol or range, the search can specify if the symbol only needs to be found somewhere in the SignBox, or if the symbol needs to be found near certain coordinates. The temporal sequence can be searched for starting symbols, written as a sequential list of symbols and ranges of symbols. When searching the temporal sequence, the search results will be limited to signs that start with a matching temporal sequence. Each query string is transformed into one or more regular expressions. The regular expressions are used to quickly search large amounts of data.

Formal SignWriting has been specifically designed to integrate with standard technology on the phone, tablet, and desktop. Four main components make this integration possible: 1) Fonts, 2) Scalar Vector Graphics, 3) HTML and CSS, and 4) JavaScript.

Formal SignWriting as ASCII characters is compatible with and optimized for UTF-8. There are 2 options for Formal SignWriting in Unicode. Option 1 is an alternate encoding that replaces the Sutton SignWriting block in Unicode. Option 1 focuses on small size, simple design, and ease of use. Option 2 is an encoding that is 97.5% official Unicode with defined characters in the Sutton SignWriting block. Option 2 focuses on augmenting the Unicode 8 standard with 17 new control characters for a design that is compatible with Formal SignWriting.

This memo defines a conceptual character encoding map for the Internet community. It is published for reference, examination, implementation, and evaluation. Distribution of this memo is unlimited.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on December 26, 2017.

Copyright Notice

Table of Contents

1. Sutton SignWriting

Sutton SignWriting is the universal and complete solution for written sign language. It has been applied by a wide and deep international community of sign languages including: American Sign Language, Arabian Sign Languages, Australian Sign Language, Bolivian Sign Language, Brazilian Sign Language, British Sign Language, Catalan Sign Language, Colombian Sign Language, Czech Sign Language, Danish Sign Language, Dutch Sign Language, Ethiopian Sign Language, Finnish Sign Language, Flemish Sign Language, French-Belgian Sign Language, French Sign Language, German Sign Language, Greek Sign Language, Irish Sign Language, Italian Sign Language, Japanese Sign Language, Malawi Sign Language, Malaysian Sign Language, Maltese Sign Language, Mexican Sign Language, Nepalese Sign Language, New Zealand Sign Language, Nicaraguan Sign Language, Norwegian Sign Language, Peruvian Sign Language, Philippines Sign Language, Polish Sign Language, Portugese Sign Language, Quebec Sign Language, South African Sign Language, Spanish Sign Language, Swedish Sign Language, Swiss Sign Language, Taiwanese Sign Language, and Tunisian Sign Language.

Sutton SignWriting is an international standard for writing sign languages by hand or with computers. From education to research, from entertainment to religion, SignWriting has proven useful because people are using it to write signed languages.

1.1. Script

Sign language is vastly different than spoken language. Instead of the sequential sounds of the voice, there is a 3 dimensional space with simultaneous action. Sutton SignWriting creates 2-dimensional writing that is visually icon and full of featural information. This is true on the symbol level and on the sign level. A symbol represents phonemic information and is full of featural information to better understand the phonemes of the symbols. A sign is a 2-dimensional arrangement of symbols and is full of featural information to better understand the morphemes of the signs.

Punctuation is represented by a single symbol and separates a series of signs into structured sentences. Line breaks should not occur before punctuation.

When written vertically, SignWriting can use 3 different lanes: left, middle, and right. The middle lane is the default lane and punctuation is always used in the middle lane. No matter the lane, the center of a sign is aligned with the center of the lane. The left and right lanes are used to represent body weight shifts and are represented by a horizontal offset from the middle lane. Body weight shifts are important to the grammar of sign languages, used for two different grammatical aspects: 1) role shifting during sign language storytelling, and 2) spatial comparisons of two items under discussion. One "role" or "item" is placed on the right side of the body (right lane), and the other on the left side of the body (left lane), and the weight shifts back and forth between the two, with the narrator in the middle (middle lane).

1.2. Symbols

The Sutton SignWriting Symbols are the building blocks of Sutton SignWriting. The symbols are arranged in 2 dimensions to create the sign images. The symbols are organized with a 16-bit coded character set and a layered hierarchy. The symbols are defined in the International SignWriting Alphabet 2010 (ISWA 2010). The ISWA 2010 is a product of the Sutton-Slevinski collaboration.

2. Formal SignWriting

Formal SignWriting (FSW) is a faithful character-encoding of Sutton SignWriting based on 2-dimensional mathematics. FSW defines a formal language for written sign languages where any sign of any sign language can be written as a string of ASCII or Unicode characters. Each sign is written as a separate word.

2.1. Building Blocks

The mathematical words of Formal SignWriting are plain text strings of characters.

2.1.1. Regular Expressions

Regular Expressions define string matching criteria. Regular Expressions offer fast processing and wide support on the various platforms.

Formal SignWriting is defined with regular expressions. Formal languages and regular expressions are used to solve fundamental problems.

Regular Expression Basics

Characters	Description	Example
*	Match a literal 0 or more times	ABC* matches AB, ABC, ABCC, ...
+	Match a literal 1 or more times	ABC+ matches ABC, ABCC, ABCCC, ...
?	Match a literal 0 or 1 times	ABC? matches AB or ABC
{#}	Match a literal "#" times	AB{2} matches ABB
[ ]	Match any single literal from a list	[ABC] matches A, B, or C
[-]	Match any single literal in a range	[A-C] matches A, B, or C
( )	Creates a group for matching	A(BC)+ matches ABC, ABCBC, ABCBCBC, ...
(|)	Matches one of several alternatives	(AB|BC|CD) will match AB, BC, or CD

Table 1

2.1.2. Token Patterns

The Formal SignWriting encoding model makes explicit those features which can be effectively and efficiently processed. The mathematical names are structured with 11 different tokens. They can be grouped in 4 layers: the 5 structural makers (A, B, L, M, R), the 3 base symbol ranges (w, s, P), the 2 modifier indexes (i, o), and the numbers (n).

The Tokens of Formal SignWriting

Token	Description
A	Sequence Marker
B	SignBox Marker
L	Left Lane Marker
M	Middle Lane Marker
R	Right Lane Marker
w	Writing BaseSymbols
s	Detailed Location BaseSymbols
P	Punctuation BaseSymbols
i	Fill Modifiers
o	Rotation Modifiers
n	Number from 250 to 749

Table 2

These tokens are used in patterns to form written sign language.

2.1.3. Characters

Each of the tokens can be encoded with a variety of characters. Formal SignWriting in ASCII has been stable since Janguary 2012. Unicode option 1 was published October 2016. Unicode option 2 was published December 2016.

Character Options

Set	Section	Description
ASCII	Section 4.1	Formal SignWriting as ASCII characters is the standard that is currently used by the Center for Sutton Movement Writing and the sign language Wikipedia projects on Wikimedia Incubator. The ASCII form is fully supported with the fonts, styling string, query string, and the various transformations.
Unicode Option 1	Section 4.2	Formal SignWriting as an alternate encoding that replaces the Sutton SignWriting block in Unicode. Option 1 focuses on small size, simple design, and ease of use. This option can complicate processing with UTF-8 and UTF-16 considerations.
Unicode Option 2	Section 4.3	Formal SignWriting as an encoding that is 97.5% official Unicode with defined characters in the Sutton SignWriting block. Option 2 focuses on augmenting the Unicode 8 standard with 17 new control characters for a design that is compatible with Formal SignWriting. This option is three times larger than Option 1 and requires the support of the ligature feature "ccmp" for Glyph Composition/Decomposition. Support in older software is limited. In the browser, "ccmp" ligatures support is possible, but it will require extra css to define the font famiily and may require extra css to enable the "ccmp" feature.

Table 3

2.1.4. Symbols

Symbols can be described with 3 tokens: base symbol, fill modifier, and rotation modifier.

Symbol Tokens

Token Pattern	Description
w	Writing BaseSymbols.
s	Detailed Location BaseSymbols.
P	Punctuation BaseSymbols.
i	Fill Modifiers.
o	Rotation Modifiers.
wio	A writing symbol as 3 tokens of writing base, fill modifier and rotation modifier. Writing symbols can be used in the spatial SignBox or the temporal sequence.
[ws]io	A writing symbol or a detailed location symbol as 3 tokens of base, fill modifier, and rotation modifier. Writing symbols and detail location symbols can be used in the temporal sequence.
Pio	A punctuation symbol as 3 tokens of punctuation base, fill modifier, and rotation modifier. Punctuation symbols divide signs into sentences.

Table 4

There are a variety of symbol types that are used for different purposes.

Symbol Types and Descriptions

Type	Description
all symbols	All symbols used in Formal SignWriting.
writing	Symbols that can be used in the spatial SignBox or the temporal sequence.
hand	Various handshapes
movement	Contact symbols, small finger movements, straight arrows, curved arrows and circles.
dynamic	Dynamic symbols are used to give the "feeling" or "tempo" to movement.
head	Symbols for the head and face.
hcenter	Used to determine the horizontal center of a sign. Same as the head type.
vcenter	Use to determine the vertical center of a sign. Includes the head and trunk types.
trunk	Symbols for torso movement, shoulders, and hips.
limb	Symbols for limbs and fingers.
location	Detailed location symbols can only be used in the temporal sequence.
punctuation	Punctual symbols are used to divide signs into sentences.

Table 5

Symbol types occur in specific ranges depending on the characters involved.

Symbol Types and Ranges

Type	ASCII	Option 1	Option 2
all symbols	S100 - S38b	U+40001 -U+4F428	U+1D800 - U+1DA8B
writing	S100 - S37e	U+40001 -U+4F904	U+1D800 - U+1DA7E
hand	S100 - 204	U+40001 -U+461A0	U+1D800 - U+1D904
movement	S205 - S2f6	U+461E1 -U+4BC98	U+1D905 - U+1D9F6
dynamic	S2f7 - S2fe	U+4BCA1 -U+4BF48	U+1D9F7 - U+1D9FE
head	S2ff - S36c	U+4BFA1 -U+4E8B1	U+1D9FF - U+1DA6C
hcenter	S2ff - S36c	U+4BFA1 -U+4E8B1	U+1D9FF - U+1DA6C
vcenter	S2ff - S375	U+4BFA1 -U+4EC38	U+1D9FF - U+1DA75
trunk	S36d - S375	U+4E8E1 -U+4EC38	U+1DA6D - U+1DA75
limb	S376 - S37e	U+4EC41 -U+4EFA0	U+1DA76 - U+1DA7E
location	S37f - S386	U+4EFA1 -U+4F2A0	U+1DA7F - U+1DA86
punctuation	S387 - S38b	U+4F2A1 -U+4F428	U+1DA87 - U+1DA8B

Table 6

2.1.4.1. ASCII

Symbol keys are 6 characters long. The first character of a symbol key is always "S". The next 3 characters identify the symbol base. The last two characters identify the fill and rotation modifiers respectively.

Symbol Key Definition

Regular Expression	Description
S	Start of symbol key
[123][0-9a-f]{2}	Symbol key base
[0-5]	Fill modifier
[0-9a-f]	Rotation modifier
S[123][0-9a-f]{2}[0-5][0-9a-f]	Symbol key definition

Table 7

2.1.4.2. Unicode Option 1

The 37,811 symbols of the International SignWriting Alphabet 2010 are uniquely idendified with Unicode characters in the range U+40001 to U+4F428.

2.1.4.3. Unicode Option 2

The 37,811 symbols of the International SignWriting Alphabet 2010 are defined with 3 characters each. A symbol is defined as a combination of base symbol, fill modifier, and a rotation modifier. The base symbols occur in the range U+1D800 to U+1DA8B. The fill modifiers occur in the range U+1DA9A to U+1DA9F. The rotation modifiers occur in the range U+1DAA0 to U+1DAAF.

2.1.5. Numbers

The numbers encode the ruler principle with characters. The ruler principle is built in automatically for scripts written sequentially in one dimension. The number characters are needed to specify the spatial relationship between symbols.

Between the various forms, the numbers exist in a restricted range of 500 and in a general range of 1,000. The more general definition simply defines 3 digits together with a potential range of 1000. A more explicit definition correctly restricts the numbers to 500 possibilities in the 250 to 749 range.

Cartesian Coordinates can be described with 2 tokens: number and number. These numbers represent the X and Y coordinates respectively.

Coordinate Tokens

Token Patterns	Description
n	Number from 250 to 749
nn	Coordinate with X and Y values as 2 numbers

Table 8

2.1.5.1. ASCII

There are 2 definitions for a number. The more general definition simply defines 3 digits together with a potential range of 1000. A more explicit definition correctly restricts the numbers to 500 possibilities in the 250 to 749 range. The general coordinate definition is adequate for processing.

General 3 digit number definition: [0-9]{3}

General coordinate definition: [0-9]{3}x[0-9]{3}

Explicit number definition from 250 to 749:
(2[5-9][0-9]|[3-6][0-9]{2}|7[0-4][0-9])

Explicit coordinate definition: (2[5-9][0-9]|[3-6][0-9]{2}|7[0-4][0-9])x(2[5-9][0-9]|[3-6][0-9]{2}|7[0-4][0-9])

2.1.5.2. Unicode Option 1

The 500 numbers of Formal SignWriting are uniquely idendified with Unicode characters in the range U+1D80C to U+1D9FF. The numbers of this set have a hard limit of 500. A coordinate is defined with 2 numbers together. Numbers and coordinates can be defined for UTF-8, UTF-16, and UTF-32.

2.1.5.3. Unicode Option 2

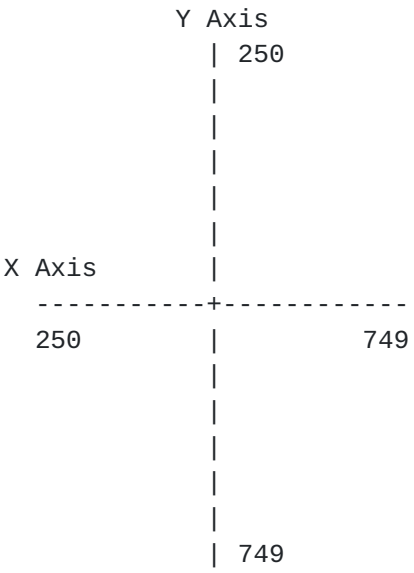
Each of the 500 numbers of Formal SignWriting is defined with 3 characters in the range U+1DAB0 to U+1DAB9. The numbers of this set have a soft limit of 500 and a hard limit of 1,000. A coordinate is

defined with 2 numbers together. Numbers and coordinates can be defined for UTF-8, UTF-16, and UTF-32.

2.1.6. Spatial SignBox

The visual image of a logographic sign is a 2-dimension arrangement of symbols inside of a SignBox. Each SignBox has a defined width, height, and 2-dimensional center that can be calculated from the plain text.

Each logographic sign exists on its own 2-dimensional SignBox. Each point on the SignBox is identified with an X and a Y coordinate. Each SignBox has a defined center. Formal numbers range from 250 to 749.



Symbols are placed on the SignBox with coordinates that represent the top-left of the symbol image. Symbol images may overlap.

The Spatial SignBox can be described with 8 tokens.

Spatial SignBox Tokens

Token Pattern	Description
B	SignBox Marker
L	Left Lane Marker
M	Middle Lane Marker
R	Right Lane Marker
w	Writing BaseSymbols
i	Fill Modifiers
o	Rotation Modifiers
n	Number from 250 to 749
wio	A writing symbol as 3 tokens of writing base, fill modifier and rotation modifier
nn	Coordinate with X and Y values as 2 numbers
wionn	A spatial symbol as 5 tokens, with 3 tokens for a writing symbol and 2 tokens for coordinates of top left placement
(wionn)*	Zero or more spatial symbols
Bnn(wionn)*	A SignBox with a preprocessed maximum coordinate and a list of spatial symbols used for horizontal writing
[LMR]	A lane marker: either left, middle or right.
[LMR]nn(wionn)*	A SignBox in either the left, middle, or right lane with a preprocessed maximum coordinate and a list of spatial symbols used for vertical writing

Table 9

The Spatial SignBox is assigned to a lane, has a preprocessed maximum coordinate and zero or more writing symbols with X and Y coordinates.

Symbol key definition: S[123][0-9a-f]{2}[0-5][0-9a-f]

Coordinate definition: [0-9]{3}x[0-9]{3}

SignBox definition: [BLMR]([0-9]{3}x[0-9]{3})(S[123][0-9a-f]{2}[0-5][0-9a-f][0-9]{3}x[0-9]{3})*

2-dimensional space does not have a normative 1-dimensional order. When symbols overlap, the relative order of the overlapping symbols is important. Otherwise, the exact string order of the spatial symbols is unpredictable.

2.1.6.1. Bounding Box

The symbols do not have a consistent width or height. The center of a symbol can be safely assumed to be at half-width and half-height. A bounding box for a symbol is based on the symbol width and height. Each symbol has a defined width and height in a text file with 37,811 lines. Alternately, the symbol width and height can be calculated by analyzing the glyphs in a TTF font file, using JavaScript or other language.

The bounding box of a sign is a tight box around the symbols. The bounding box is used to determine the width and height of a sign. The center of a bounding box is coordinate 500,500.

The bounding box of a sign consists of four values: Minimum X, Minimum Y, Maximum X and Maximum Y. The values of the bounding box is taken straight from the coordinates in an Formal SignWriting word.

2.1.6.2. Maximum Coordinate

The maximum coordinate for a SignBox is pre-calculated to simplify layout for width, height, and center. For each symbol, the width of height of that symbol is added to the coordinate position of that symbol. These new coordinate values represent the bottom-right coordinate of each symbol bounding box. The maximum X value is joined with the maximum Y value to determine the maximum coordinate.

2.1.6.3. Centering a Sign

To simplify layout and improve 2-dimensional searching, every sign has a normalized center based on symbol type, size, and mathematical formula. The vertical center is based on the center of the bounding box around the head symbols. The horizontal center is based on the center of the bounding box around the head and trunk symbols. If a sign doesn't contain head or trunk symbols, then the bounding box of all symbols is used. For the symbol ranges see Table 6

Once the center of a sign has been determined, the symbols are moved so that the center is coordinate 500,500.

2.1.7. Temporal Sequence

Signs are written in 2-dimensional space which does not have a normative 1-dimensional order. Any 1-dimensional order of 2-dimensional space is subjective. Some 1-dimensional orders may be canonical according to a particular theory, but there are a variety of theories on setting a 1-dimensional order.

The temporal sequence describes a 1-dimensional order that is separate from the spatial SignBox, rather than ordering the 2-dimensional space directly. The temporal sequence is written as an optional prefix to a spatial SignBox. The temporal sequence will use the same symbols that are used in the spatial SignBox, but it does not need to use all of them and it is not limited to only those symbols. The temporal sequence is a list of writing symbols and/or detailed location symbols that identify temporal order and additional analysis. A valid sequence must contain at least one symbol and can not contain punctuation.

The temporal sequence allows for sorting that is universally supported through binary string comparison.

There are several theories on the best way to structure a temporal sequence. The most productive is based on the SignSpelling Sequence theory of Valerie Sutton. A temporal sequence is structured as a series of starting handshapes followed by optional movements, transitional handshapes, movement, and end handshapes. Only symbols of type "hand" and "movement" should be used in this first section. The last section of the temporal sequence should contain symbols of of type "dynamic", "head", "trunk", and "limb".

Detailed location symbols of type "location" can be used in a temporal sequence, but are rarely (if ever) needed for general writing.

A temporal sequence can be described with 5 tokens.

Temporal Sequence Tokens

Token Patterns	Description
A	Sequence Marker
w	Writing BaseSymbols
s	Detailed Location BaseSymbols
i	Fill Modifiers
o	Rotation Modifiers
(A([ws]io)+)?	An optional temporal sequence to be used as a prefix for a SignBox

Table 10

The temporal prefix starts with a sequence marker and includes an ordered list of writing symbols and detailed locations.

Temporal Sequence Definition

Regular Expression	Description
(A(S[123][0-9a-f]{2}[0-5][0-9a-f])+)?	An optional temporal sequence as a sequence marker followed by one or more symbols.

Table 11

2.1.8. Sentences

Signs are mixed with punctuation to form text. Punctuation is a single symbol and separates a series of signs into structured sentences. A punctuation symbol is always used alone and should not be used in a sign. Line breaks should not occur before punctuation.

When written vertically, SignWriting can use 3 different lanes: left, middle, and right. The middle lane is the default lane and

punctuation is always used in the middle lane. No matter the lane, the center of a sign is aligned with the center of the lane.

For body weight shifts to one side or the other, the center of the sign is aligned with a fixed horizontal offset from the middle lane into either the left or right lane.

The left and right lanes are used to represent body weight shifts and are represented by a horizontal offset from the middle lane. Body weight shifts are important to the grammar of sign languages, used for two different grammatical aspects: 1) role shifting during sign language storytelling, and 2) spatial comparisons of two items under discussion. One "role" or "item" is placed on the right side of the body (right lane), and the other on the left side of the body (left lane), and the weight shifts back and forth between the two, with the narrator in the middle (middle lane).

Sentence Token Patterns

Regular Expression	Description
Pionn	a punctuation symbol as a punctuation base symbol with a preprocessed minimum coordinate
(((A([ws]io)+)?Bnn(wionn)*)|Pionn)+	a sign text for horizontal writing as a string of SignBoxes (with optional prefixes) and punctuation
(((A([ws]io)+)?[LMR]nn(wionn)*)|Pionn)+	a sign text for vertical writing as a string of SignBoxes in lanes (with optional prefixes) and punctuation

Table 12

Sentences mix signs with punctuation to form text.

Punctuation definition: S38[7-9ab][0-5][0-9a-f][0-9]{3}x[0-9]{3}

Formal SignWriting text definition: ((A(S[123][0-9a-f]{2}[0-5][0-9a-f])+)?[BLMR]([0-9]{3}x[0-9]{3})(S[123][0-9a-f]{2}[0-5][0-9a-f][0-9]{3}x[0-9]{3})*|S38[7-9ab][0-5][0-9a-f][0-9]{3}x[0-9]{3})((A(S[123][0-9a-f]{2}[0-5][0-9a-f])+)?[BLMR]([0-9]{3}x[0-9]{3})(S[123][0-9a-f]{2}[0-5][0-9a-f][0-9]{3}x[0-9]{3})*| S38[7-9ab][0-5][0-9a-f][0-9]{3}x[0-9]{3})*

2.2. Styling String

The styling string of Formal SignWriting uses a lite markup to define a variety of styling options. The entire sign can be customized for padding, coloring, and size. Individual symbols within a sign can be customized for coloring and size. For SVG output, class names and IDs can be defined. A styling string can be added to the end of any Formal SignWriting string to style a particular sign.

Colors can be written as CSS color names or as color hex values.

CSS Color Names: [a-zA-Z]+

Color Hex Values: [0-9a-fA-F]{3}([0-9a-fA-F]{3})?

The styling string is divided into 3 sections: one for the entire sign, one for individual symbols, and one for SVG class names and ID. The styling string starts with a single dash, after which is the section about the entire sign. A second dash, if present, marks the start of the section about the individual symbols. A third dash, if present, marks the start of the section about the SVG class names and ID. The order of the styling options is important.

Styling String: -C?(P[0-9]{2})?(G_([0-9a-fA-F]{3}([0-9a-fA-F]{3})?|[a-zA-Z]+)_)?(D_([0-9a-fA-F]{3}([0-9a-fA-F]{3})?|[a-zA-Z]+)(,([0-9a-fA-F]{3}([0-9a-fA-F]{3})?|[a-zA-Z]+))?_)?(Z([0-9]+(\.[0-9]+)?|x))?(-(D[0-9]{2}_([0-9a-fA-F]{3}([0-9a-fA-F]{3})?|[a-zA-Z]+)(,([0-9a-fA-F]{3}([0-9a-fA-F]{3})?|[a-zA-Z]+))?_)*(Z[0-9]{2},[0-9]+(\.[0-9]+)?(,[0-9]{3}x[0-9]{3})?)*)?(--?[_a-zA-Z][_a-zA-Z0-9-]{0,100}(-?[_a-zA-Z][_a-zA-Z0-9-]{0,100})*!([a-zA-Z][_a-zA-Z0-9-]{0,100}!)?)?

2.2.1. Entire Sign

There are several options for styling an entire sign.

C Colorize

P Padding

G Background

D Detail colors

Z Zoom level

2.2.1.1. Colorize

Colorizing a sign will set the color of each symbol based on its classification.

Hand 0000CC

Movement CC0000

Dynamic FF0099

Head 006600

Body 000000

Detailed Location 884411

Punctuation FF9900

Styling String	Description
-C	Colorize the symbols of the sign

Table 13

2.2.1.2. Padding

Padding is applied around the entire sign. A two-digit number is used to set the padding.

Styling String	Description
-P01	A padding of 1 around the sign

Table 14

2.2.1.3. Background

By default, the background of a sign is transparent. The background color can be set with a CSS color name or with a color hex value. The color name or value must be surrounded by underscores.

Styling String	Description
-G_lightblue_	Background color of light blue.
-G_f00_	Background color as 3 hex values.
-G_ff0000_	Background color as 6 hex values.

Table 15

2.2.1.4. Detail Colors

By default, each symbol has a line color of black and a fill color of white. The line color for all of the symbols can be set with a CSS color name or with a color hex value. The color name or value must be surrounded by underscores. Setting the fill color is optional. To set the fill color, put a comma and the fill color after the line color but before the closing underscore.

Styling String	Description
-D_red_	Line color of red.
-D_red,yellow_	Line color of red with a fill color of yellow.

Table 16

2.2.1.5. Zoom Level

By default, a sign is set to zoom level 1. The zoom level can be set with an integer or a decimal number.

Alternatively, the zoom level can be set to lower-case 'x', for extendable. The SVG created will not specify the width or height, so that the sign image will fill whatever container it is placed inside.

Styling String	Description
-Z2	Zoom level of 2
-Z15.7	Zoom level of 15.7
-Zx	Zoom level of extendable

Table 17

2.2.2. Individual Symbols

There are two options for styling individual symbols. Individual symbols are identified by a two-digit number, which identifies the order the symbol appears in the SignBox.

D Detail colors

Z Zoom level

2.2.2.1. Detail Colors

By default, each symbol has a line color of black and a fill color of white. The line color for an individual symbol can be set with a CSS color name or with a color hex value. The color name or value must be surrounded by underscores. Setting the fill color is optional. To set the fill color, put a comma and the fill color after the line color but before the closing underscore.

Styling String	Description
--D01_red_	First symbol line color of red.
--D01_red,yellow_	First symbol line color of red with a fill color of yellow.
--D01_red_D02_green_	First symbol line color of red and second symbol line color of green.

Table 18

2.2.2.2. Zoom Level

By default, each symbol is set to zoom level 1. The zoom level of individual symbols can be set with an integer or a decimal number.

Additionally, an offset coordinate can be specified with an individual symbol's zoom level. The offset coordinate of 500x500 is considered no offset for either the x or y value.

Styling String	Description
--Z03,2	Third symbol zoom level of 2
--Z04,15.7	Fourth symbol zoom level of 15.7
--Z04,1.5,480x500	Fourth symbol zoom level of 1.5 with a -20 offset applied to the X value of the symbol's placement coordinate.

Table 19

2.2.3. SVG Class Names and ID

When using SVG, there are two additional styling options of class names and ID.

{class names}! SVG Class Names

{ID}! SVG ID

Both class names and ID use a restricted ASCII subset.

class names -?[_a-zA-Z][_a-zA-Z0-9-]{0,100}(-?[_a-zA-Z][_a-zA-Z0-9-]{0,100})*

ID [a-zA-Z][_a-zA-Z0-9-]{0,100}

Each SVG can be created with a list of class names separated by spaces, ending in an exclamation (!) mark. After the class names exclamation mark, an ID can be written followed by another exclamation mark.

Styling String	Description
---glowing!	A class name of "glowing"
---flashing primary!	Two class names of "flashing" and "primary".
---!cursor!	SVG created with an ID of "cursor"
---flashing!cursor!	SVG created with a class name of "flashing" and an ID of "cursor"

Table 20

2.3. Query Language

The query language is a lite ASCII markup similar to Formal SignWriting. Any Formal SignWriting string can easily be converted into several different query string, depending on the search parameters.

The query string is a concise representation for a much larger and detailed set of regular expressions. The regular expressions can be used to quickly and accurately search large files and databases containing Formal SignWriting.

A filter and repeat pattern of searching is used as a series of match criteria. A file, database, or text input is searched using a sequence of steps. Each step applies a single match criteria. Matching results are collated and the next search criteria is applied. The pattern of searching the previous results continues until all regular expressions have been used.

There are two main sections of a query string. The first searches the spatial SignBox. The second searches the temporal sequence. Both sections use the same definition for a symbol or a range. The symbol search can match an exact symbol, or a set of related symbols. For the fill and rotation modifiers, the "u" character is a wildcard. The "u" stands for unknown and will match all values rather than a specific character. The range search can match a range of base symbols. The base symbol range consists of 2 values: the starting base symbol and the ending base symbol. Every symbol between these 2 base symbols will be matched.

Symbol Search: S[123][0-9a-f]{2}[0-5u][0-9a-fu]

Range Search: R[123][0-9a-f]{2}t[123][0-9a-f]{2}

At the end of the query string is an optional styling string flag represented by a dash (-). If present, the Formal SignWriting strings will include any styling strings. If the styling string flag isn't included, the query string will only find plain text Formal SignWriting strings without the styling string.

The full query string definition allows for the possibility of searching the temporal sequence and the spatial SignBox at the same time.

Query String: Q((A(S[123][0-9a-f]{2}[0-5u][0-9a-fu]|R[123][0-9a-f]{2}t[123][0-9a-f]{2})+)?T)?(S[123][0-9a-f]{2}[0-5u][0-9a-fu]([0-9]{3}x[0-9]{3})?|R[123][0-9a-f]{2}t[123][0-9a-f]{2}([0-9]{3}x[0-9]{3})?)*(V[0-9]+)?-?

2.3.1. Searching the Spatial SignBox

The spatial SignBox is a list of symbols with 2-dimensional placement. The query "Q" will find all signs regardless of the symbols used or their placement.

It is possible to specify one or more symbols (or ranges of symbols) that must be included in the SignBox to indicate a match. The order of the symbols is not important. Each symbol (or range) can include an optional coordinate. The coordinate is a restriction on the match, such that a symbol must be used within a certain variance of the coordinate to qualify as a match.

The variance is a number value, 0 or greater with a default value of 20. A variance of 0 will only find symbols used at an exact coordinate. A variance of 5 will match the symbols used at a coordinate, plus or minus 5 for both X and Y numbers.

Symbol Search with Optional Coordinate: S[123][0-9a-f]{2}[0-5u][0-9a-fu]([0-9]{3}x[0-9]{3})?

Range Search with Optional Coordinate: R[123][0-9a-f]{2}t[123][0-9a-f]{2}([0-9]{3}x[0-9]{3})?

Variance: (V[0-9]+)?

Spatial SignBox Search Query: Q(S[123][0-9a-f]{2}[0-5u][0-9a-fu]([0-9]{3}x[0-9]{3})?|R[123][0-9a-f]{2}t[123][0-9a-f]{2}([0-9]{3}x[0-9]{3})?)*(V[0-9]+)?

Spatial SignBox Query Examples

Query	Description
Q	All signs
QS100uu	Signs with the index handshape in the spatial SignBox
QS100uu480x480	Signs with the index handshape in the spatial SignBox used near coordinate (480,480)
QS100uu480x480V0	Signs with the index handshape in the spatial SignBox used at the exact coordinate (480,480)
QS100uuR2fft36c	Signs with the index handshape and a symbol from the head & face range

Table 21

2.3.2. Searching the Temporal Sequence

The temporal sequence is a list of symbol keys. The query "QT" will find all signs that include a temporal sequence.

It is possible to specify the start of the temporal sequence by identifying a series of symbols and/or ranges. The query will start with an "QA" and end with a "T", such as "QA...T". Between the "QA" and "T", a series of symbol searches and/or range searches will specify the desired start of the temporal sequence. The order of the symbols and ranges is important.

Temporal Sequence Search Query: Q((A(S[123][0-9a-f]{2}[0-5u][0-9a-fu]|R[123][0-9a-f]{2}t[123][0-9a-f]{2})+)?T)?

Temporal Sequence Query Examples

Query	Description
QT	All signs that include the temporal sequence
QAS100uuT	Signs with a temporal sequence that starts with the index handshape
QAS100uuR100t204S20500T	Signs with a temporal sequence that starts with the index handshape, followed by any handshape, followed by the single contact

Table 22

2.3.3. Including the Styling String

At the end of the query string is an optional styling string flag represented by a dash (-). If present, the Formal SignWriting strings will include any styling strings. If the styling string flag isn't included, the query string will only find plain text Formal SignWriting strings without the styling string.

Styling String Search Query: Q-

Styling String Search Only: -

Styling String Query Examples

Query	Description
Q-	All signs including the styling strings when present
-	Only find styling string without including the Formal SignWriting

Table 23

2.4. Transformations

Formal SignWriting and the surrounding technologies have been created to facilitate easy transformations between the various forms.

2.4.1. Formal SignWriting to Query String

Formal SignWriting strings have several natural transformations to query string. The transformation can use the temporal sequence and/or the spatial SignBox. For each symbol, the query can include the exact symbol key, or the query can use a general symbol key where the fill and rotation modifiers are not explicitly defined. Consider the Formal SignWriting string "AS14c20S27106M518x529S14c20481x471S27106503x489".

Exact Temporal Sequence Symbols: QAS14c20S27106T

General Temporal Sequence Symbols: QAS14cuuS271uuT

Exact Spatial SignBox Symbols: QS14c20S27106

General Spatial SignBox Symbols: QS14cuuS271uu

Exact Spatial SignBox Symbols with Location:
QS14c20481x471S27106503x489

General Spatial SignBox Symbols with Location:
QS14cuu481x471S271uu503x489

2.4.2. Query String to Regular Expression

The transformation from query string to regular expressions has been fully implemented in the Sutton SignWriting JavaScript Library and the SignWriting Server.

The query language to regular expressions generator uses the following regular expression structures as building blocks.

Temporal Sequence Prefix: (A(S[123][0-9a-f]{2}[0-5][0-9a-f])+)

SignBox Prefix: [BLMR]([0-9]{3}x[0-9]{3})

Spatial Symbols: (S[123][0-9a-f]{2}[0-5][0-9a-f][0-9]{3}x[0-9]{3})*

The Temporal Sequence Prefix is a structural marker followed by one or more symbols. For the query string "QT", the prefix is required. For the general "Q", the prefix is optional so "?" is appended to the Temporal Sequence Prefix regular expression.

The SignBox Prefix is a combination of structural marker and preprocessed maximum coordinate. Every constructed regular expression will include the SignBox Prefix.

The Spatial Symbols is zero or more symbol definitions and associated coordinates. The Spatial Symbols regular expression is used for every search. For both "Q" and "QT", it is the only symbol matching used. When searching for specific symbols and ranges, the general Spatial Symbols definition will sandwich the specific search definitions.

Searching for number ranges with regular expressions requires a unique technique. This technique requires five steps.

Find a number between 122 and 455

1) 10's don't match and the min 1's are not zero (last number to 9):

   Match 12[2-9]

2) Bring up the 10's if hundreds are different: Match 1[3-9][0-9]

3) Bring up the 100's if different: Match [2-3][0-9][0-9]

4) Bring up the 10's: Match 4[0-4][0-9]

5) Bring up the 1's: Match 45[0-5]

Final Match (12[2-9]|1[3-9][0-9]|[2-3][0-9][0-9]|4[0-4][0-9]|45[0-5])

For the styling string regular expression, see Section 2.2.

3. Technology Integration

Formal SignWriting has been specifically designed to integrate with standard technology on the phone, tablet, and desktop. Four main components make this integration possible: 1) Font Technology, 2) Scalar Vector Graphics, 3) HTML and CSS, and 4) a JavaScript Library.

3.1. Fonts

The Sutton SignWriting Fonts are available as source SVG and as two TrueType Font files.

> Sutton SignWriting Fonts
>

3.1.1. Installing the TrueType Fonts

The Sutton SignWriting TrueType fonts are available for download and installation. The fonts have been tailored for the Sutton SignWriting JavaScript library. Please ignore warning and unusual font previews during installation as these will not affect the fonts utility.

Installing the fonts using the instructions below is not required, but it will improve the user experience. If the fonts are not installed on the system, CSS declarations will install the fonts in the browser cache.

3.1.1.1. Windows, Linux, and Mac

Installation is straight forward for Windows, Linux and Mac. Simply download the 2 TrueType fonts and install as usual.

Sutton SignWriting TrueType Font [SuttonSignWritingFont]

Sutton SignWriting Fill TrueType Font [SuttonSignWritingFontFill]

3.1.1.2. iOS

Installation is possible on iOS with a configuration profile that includes the 2 TrueType fonts. Simply download the configuration profile and install.

Sutton SignWriting Configuration Profile [SuttonSignWritingProfile]

3.1.1.3. Android

Android can not install the fonts directly onto the system. The CSS declarations below will install the fonts in the browser cache.

3.1.2. Using the Fonts without Installation

The TrueType Fonts can be used without installing the fonts on any platform by defining two font-face statements. Simply include the following CSS in any HTML page to access the fonts. Make sure to replace the URLs with the fully qualified links for both fonts.

```
@font-face {
  font-family: "SuttonSignWriting";
  src:
    local('SuttonSignWriting'),
    url('https://.../SuttonSignWriting.ttf') format('truetype');
}
@font-face {
  font-family: "SuttonSignWritingFill";
  src:
    local('SuttonSignWritingFill'),
    url('https://.../SuttonSignWritingFill.ttf') format('truetype');
}
```

If the fonts are installed, then the system fonts will be used. If the fonts are not installed when a SignWriting Font page is opened, the CSS will cause the fonts to be automatically downloaded to the browser's cache on the first visit. Once the fonts are installed in the browser cache, they will remain there until the browser cache is emptied. Any webside that uses this CSS can access the browser installed font without requesting a new copy. The fonts are 10 MB, so the first install make take a few seconds or longer depending on your download speed and processor.

3.2. Scalar Vector Graphics

Sutton SignWriting is a 2-dimensional script. The sign images are composed using Scalar Vector Graphic (SVG).

3.2.1. Font Based SVG

The conversion of Formal SignWriting to Scalar Vector Graphics requires three parts: header, text, and symbols. Consider the FSW string "M518x533S1870a489x515S18701482x490S20500508x496S2e734500x468".

3.2.1.1. SVG Header

The header section contains the SVG definition along with the width, height, and viewbox. The viewbox is a combination of the minimum X, minimum Y, width, and height.

Minimum X: 482

Maximum X: 518

Width: 36

Minimum Y: 468

Maximum Y: 533

Height: 65

```
<svg version="1.1" xmlns="http://www.w3.org/2000/svg"
  width="36" height="65" viewBox="482 468 36 65">
```

If the width and height properties are not included, then the resulting SVG will automatically expand in size to fill the containing element on the screen.

```
<svg version="1.1" xmlns="http://www.w3.org/2000/svg"
  viewBox="482 468 36 65">
```

3.2.1.2. SVG Text

The SVG text section is included to make it possible to copy and paste Formal SignWriting strings. The font-size is set to zero to make the text invisible.

```
<text style="font-size:0%;">
M518x533S1870a489x515S18701482x490S20500508x496S2e734500x468
</text>
```

3.2.1.3. SVG Symbols

Each symbol in the SignBox is a combination of the symbol key and the positioning coordinate.

Symbol 1: S1870a 489x515

Symbol 2: S18701 482x490

Symbol 3: S20500 508x496

Symbol 4: S2e734 500x468

Each spatial symbol is written as an SVG group and positioned by the transformation translate.

```
  <g transform="translate(489,515)">...</g>
  <g transform="translate(482,490)">...</g>
  <g transform="translate(508,496)">...</g>
  <g transform="translate(500,468)">...</g>
```

Inside of each group, 2 text elements are written. The symbol fill is written first using the SuttonSignWritingFill font with a plane 16 character. The symbol line is written second using the

SuttonSignWriting font with a plane 4 character. See Section 4.2.1 for the formula to convert symbol keys to codepoints.

```
<text class="sym-fill"
 style="font-family:'SuttonSignWritingFill';font-size:30px;fill:white;">
 {plane 16 codepoint}
</text>
<text class="sym-line"
 style="font-family:'SuttonSignWriting';font-size:30px;fill:black;">
 {plane 4 codepoint}
</text>
```

3.2.2. Stand Alone SVG

It is possible to request completed SVG images from the SignWriting Server [SignWritingServer]. The SVG images created by the SignWriting Server are stand-alone graphics that do not use the TrueType Fonts. The SVG images use path elements to define the symbol lines and curves.

The SVG header and SVG text for the server-side images are the same as the standard FSW to SVG transformation. See Section 3.2.1

The SVG symbols section is structured differently. Multiple SVG elements are contained within each sign SVG image. Each sub-SVG element uses X and Y coordinates to place each symbol. Consider the FSW string "M518x533S1870a489x515S18701482x490S20500508x496S2e734500x468".

Symbol 1: S1870a 489x515

Symbol 2: S18701 482x490

Symbol 3: S20500 508x496

Symbol 4: S2e734 500x468

```
<svg x="489" y="515">...</svg>
<svg x="482" y="490">...</svg>
<svg x="508" y="496">...</svg>
<svg x="500" y="468">...</svg>
```

Inside of each sub-SVG element is a group (g) element with one or two path elements. This inside information can only be requested from the SignWriting Server or some other source of the symbol image data.

```
  <g transform="translate(0.146473559361,17.7697467366) ... ">
    <path class="sym-fill" fill="white" d="M700 1493 ... "/>
    <path class="sym-line" fill="black" d="M1826 1480 ... "/>
  </g>
```

3.3. HTML and CSS

Basic HTML structures and CSS rules can be used with Sutton SignWriting for customization and layout.

3.3.1. Centering and Sizing

It is possible to center a symbol or sign within a div with a few CSS rules. The symbol or sign will automatically shrink in size if the containing div is smaller than the SVG image. Additionally, if the SVG is created with the zoom level of extendable (styling string "-Zx"), the symbol or sign will grow in size to fill as much of the containing div as possible.

```
<div class="centered">
  <svg version="1.1" xmlns="http://www.w3.org/2000/svg" ...
</div>

div.centered {
  position: relative;
  width: 10%;
  height: 10%;
  border: 1px solid black;
}

div.centered svg {
  position: absolute;
  display: block;
  top:2.5%;
  bottom: 2.5%;
  left: 2.5%;
  right: 2.5%;
  margin: auto;
  max-width: 95%;
  max-height: 95%;
}
```

3.3.2. Coloring Symbols and Signs

Individual signs can be colored with CSS rules. The individual classes of 'sym-line' and 'sym-fill' can be used to isolate each part of a symbol, both positive and negative spaces, or the classes can be

ignored to create the shadow of a symbol that includes both aspects of a symbol.

```
<svg class="primary" ...
<svg class="success" ...
<svg class="info" ...
<svg class="warning" ...
<svg class="danger" ...
<svg class="shadow" ...
<svg class="inverse" ...

svg.primary g text.sym-line { fill: #337ab7 !important; }
svg.success g text.sym-line { fill: #5cb85c !important; }
svg.info g text.sym-line { fill: #5bc0de !important; }
svg.warning g text.sym-line { fill: #f0ad4e !important; }
svg.danger g text.sym-line { fill: #d9534f !important; }
svg.shadow g text { fill: grey !important; }
svg.inverse g text.sym-line { fill: white !important; }
svg.inverse g text.sym-fill { fill: black !important; }
```

3.3.3. Other Effects

Other CSS rules can be used for other effect. Please note that transform property does not effect the document flow and should not be used for general layout.

```
svg.shadowed {
  text-shadow: -1px -1px 1px #fff, 1px 1px 1px #000;
}
svg.rotate {
  transform: rotate(0.5turn);
}
svg.bigger {
  transform: scale(2);
}
svg.skewed {
  transform: skewX(30deg);
}
```

3.3.4. Sentences

SignWriting is written vertically using the vertical writing mode of CSS. To create the center lane and to visually divide the columns of text, several span elements are used. Each sign is contained in a div with a width and height that matches the enclosed sign. To properly align each sign with the center of its lane, the containing div will either use "margin-right" or "border-left". With "border-left", the rule must include "solid transparent" after the size.

```
<div class="signtext">
 <span class="outside"><span class="middle"><span class="inside">
  <div style="width:42px;height:77px;margin-right:2px;"><svg ...
  <div style="width:38px;height:48px;margin-right:2px;"><svg ...
  <div style="width:25px;height:9px;border-left:7px solid transparent;">
```

```
   div.signtext {
     -webkit-writing-mode: vertical-lr;
     writing-mode: vertical-lr;
     font-size: 0%;
     border-left: 1px solid blue;
     height: 100%;
   }

   span.outside { border-left: 1px solid blue; vertical-align: top; }
   span.middle { vertical-align: bottom; }
   span.inside { border-left: 1px dashed red; }

   div.signtext div {
     writing-mode: horizontal-tb;
     display: inline-block;
     vertical-align: middle;
     padding: 20px;
     box-sizing: content-box;
   }
```

3.4. JavaScript

The Sutton SignWriting JavaScript Library leverages the Sutton SignWriting Fonts, without additional dependancies.

The Sutton SignWriting JavaScript Library is part of the Sutton SignWriting Project [SuttonSignWritingProject]

```
          Sutton SignWriting JavaScript Library
          Copyright (c) 2007-2016, Stephen E Slevinski Jr
          Licensed under the MIT License
```

4. Unicode Considerations

4.1. UTF-8

Formal SignWriting as ASCII characters is compatible with and optimized for UTF-8.

Formal SignWriting as ASCII characters is the standard that is currently used by the Center for Sutton Movement Writing and the sign

language Wikipedia projects on Wikimedia Incubator. The ASCII form is fully supported with the fonts, styling string, query string, and the various transformations.

4.2. Option 1

Formal SignWriting as an alternate encoding that replaces the Sutton SignWriting block in Unicode. Option 1 focuses on small size, simple design, and ease of use. This option can complicate processing with UTF-8 and UTF-16 considerations.

4.2.1. Symbols

In order to support Sutton SignWriting, you must support all of the Sutton SignWriting Symbols. This set has been designed as a 16-bit coded character set. Using these derived codepoints makes it easier to reference the individual Sutton SignWriting Symbols in a variety of ways. For inside of a font file, an entire Unicode plane can be used to reference these symbols. Inside of the font files, plane 4 is used to identify each glyph, but these characters have not been officially proposed to Unicode. Alternately, plane 16 can be used, but this creates a font filled with private use area characters. Finally, the glyphs can be added to a font file without assigning a unique Unicode codepoint to each, but this results in each glyph being given a unique glyph ID that is only meaningful for that specific font file.

A simple formula transforms a symbol key into a codepoint. Given a symbol key as variable "key", in JavaScript the function is defined as:

```
var code = ((parseInt(key.slice(1,4),16) - 256) * 96) +
((parseInt(key.slice(4,5),16))*16) + parseInt(key.slice(5,6),16) +
1;
```

4.2.2. Other Characters

In addition to the Sutton SignWriting symbols, the structural markers and number characters need to be supported. These other characters overwrite the Sutton SignWriting block with a viable character design.

Description	Formal SignWriting	x-Character-SignWriting
Sequence Marker	A	U+1D800
SignBox Markers	B, L, M, R	U+1D801 to U+1D804
Numbers	250 to 749	U+1D80C to U+1D9FF

Table 24

4.3. Option 2

Formal SignWriting as an encoding that is 97.5% official Unicode with defined characters in the Sutton SignWriting block. Option 2 focuses on augmenting the Unicode 8 standard with 17 new control characters for a design that is compatible with Formal SignWriting. This option is three times larger than Option 1 and requires the support of the ligature feature "ccmp" for Glyph Composition/Decomposition. Support in older software is limited. In the browser, "ccmp" ligatures support is possible, but it will require extra css to define the font famiily and may require extra css to enable the "ccmp" feature.

4.3.1. Official Characters

In 2015, the symbols of Sutton SignWriting were added to Unicode version 8.

See Section 1.2 and Section 2.1.4

Description	Unicode Range
Base Charcters	U+1D800 to U+1DA8B
Fill Modifiers 2 to 6	U+1DA9B to U+1DA9F
Rotation Modifiers 2 to 16	U+1DAA1 to U+1DAAF

Table 25

Each symbol key can be rewritten as 3 Unicode characters of a base, a fill, and a rotation. Given a symbol key as variable "key", in JavaScript the 3 characters can be derived with the following statements:

```
var base = parseInt(key.substr(1,3),16) + parseInt('1D700',16);

var fill = parseInt(key.substr(4,1),16) + parseInt('1DA9A',16);

var rotation = parseInt(key.substr(5,1),16) +
parseInt('1DAA0',16);
```

4.3.2. 17 New Characters

The addition of 17 Unicode characters to the official characters will complete the script encoding and cover 2-dimensional layout.

Description	Formal SignWriting	Proposed Unicode
Fill Modifier 1	0	U+1DA9A
Rotation Modifier 1	0	U+1DAA0
Numbers	0 to 9	U+1DAB0 to U+1DAB9
Sequence Marker	A	U+1DABA
SignBox Markers	B	U+1DABB
Left Lane Markers	L	U+1DABC
Middle Lane Markers	M	U+1DABD
Right Lane Markers	R	U+1DABE

Table 26

Fill Modifier 1 and Rotation Modifier 1 are included to fix sorting and simplify processing.

The 10 number characters express the concept of distance, important for use with 2-dimensional scripts.

The 5 structural markers define cohesive units of the script.

5. IANA Considerations

None.

6. Security Considerations

None.

7. References

[SignWritingServer]
Slevinski, S., "SignWriting Server",
<https://signpuddle.net>.

[SuttonSignWritingFont]
Slevinski, S., "Sutton SignWriting TypeType Font",
<https://cdn.rawgit.com/Slevinski/SuttonSignWriting/master/assets/SuttonSignWriting.ttf>.

[SuttonSignWritingFontFill]
Slevinski, S., "Sutton SignWriting Fill TypeType Font",
<https://cdn.rawgit.com/Slevinski/SuttonSignWriting/master/assets/SuttonSignWritingFill.ttf>.

[SuttonSignWritingProfile]
Slevinski, S., "Sutton SignWriting Configuration Profile",
<https://cdn.rawgit.com/Slevinski/SuttonSignWriting/master/assets/SuttonSignWriting.mobileconfig>.

[SuttonSignWritingProject]
Slevinski, S., "Sutton SignWriting Project",
<https://github.com/Slevinski/SuttonSignWriting>.

Author's Address

Stephen E Slevinski Jr
Center for Sutton Movement Writing

Email: slevinski@signwriting.org