Universal Multiple-Octet Coded Character Set
International Organization for Standardization
Organisation Internationale de Normalisation
Международная организация по стандартизации

Doc Type: **Working Group Document**
Title: **Cluster Formation Model for Khitan Small Script**
Source: **Andrew West, Viacheslav Zaytsev, Michael Everson**
Status: **Individual Contribution**
Action: **For consideration by JTC1/SC2/WG2 and UTC**
Date: **2018-05-19**

1. Introduction

The Khitan Small Script (KSS) is traditionally written in vertical columns running right to left, and comprises a mixture of standalone logographic characters and clusters of two to eight phonographic characters that form a rectangular block representing an individual word. There is no analogy for these rectangular clusters in any other script encoded in the UCS, and so various possible mechanisms for controlling cluster formation at the encoding level have been suggested.

Following on from the Meeting on Khitan Scripts held at Yinchuan, China in August 2016, the authors of "Final proposal to encode the Khitan Small Script in the SMP of the UCS" (WG2 N4738R2, L2/16-245R) proposed putting a single format character at the start of each cluster to indicate that the contiguous sequence of following characters formed a cluster. Two format characters (KHITAN SMALL SCRIPT SINGLE CLUSTER INITIAL and KHITAN SMALL SCRIPT DOUBLE CLUSTER INITIAL) would be required for encoding as there are two distinct cluster patterns (described in Section 2 below). Thus, a cluster of five KSS character would be represented as <-ABCDE> or <=ABCDE> (where – and = represent the single and double cluster initials respectively).

At WG2 Meeting 65 in San José in September 2016 the Ad Hoc Meeting on Khitan Small Script (see WG2 N4768, L2/16-338) decided not to accept the model agreed at Yinchuan, but rather to use two format characters (KHITAN SMALL SCRIPT HORIZONTAL JOINER and KHITAN SMALL SCRIPT VERTICAL JOINER) which would be placed alternatively between every graphic character in a cluster. Thus, a cluster of five KSS characters would be represented as <A*B:C*D:E> or <A:B*C:D*E> (where * and : represent the horizontal and vertical joiners respectively).

1.1 Summary of Proposed Model for Cluster Formation

This document proposes a new model for automatic cluster formation, which requires no new format characters to be encoded. Font design and OpenType implementation with respect to this model are also discussed.

In summary, this document proposes the following clustering model which:

- Automatically composes any sequence of two or more KSS characters into a cluster;
- Starts a default cluster with two adjacent characters;
- Inserts Combining Grapheme Joiner (CGJ) between the 1st and 2nd characters in a sequence to produce a cluster starting with a single centred character;
- Terminates a cluster sequence by EOL or any character that is not a KSS graphic character or CGJ;
- Separates clusters in the text stream from each other with a space character (U+0020 or any desired space character); and
- Separates single standalone characters in the text stream with a space character to prevent them forming a cluster.

With this model, caret placement, caret movement, selection, deletion and backspacing should work in exactly the same way as for decomposed Hangul displayed using a composing font. That is to say, it should not be possible to click into the middle of a composed Khitan cluster or visually select a partial section of a composed Khitan cluster; backspacing should delete one character at a time, with a new cluster forming each time; the number of presses of the left/right arrow keys required to move through a composed Khitan cluster should match the number of characters in the cluster (cf. a decomposed LVT Hangul syllable which requires three arrow presses to move past it). This has been tested using our prototype Khitan font on Notepad under Microsoft Windows 10, and caret placement, caret movement, selection, deletion and backspacing behaviour is as described above. With Word 2016 it is possible to place the caret in the middle of a composed cluster and to visually select part of a composed cluster, but we consider this to be a bug with Word.

When composing a Khitan cluster by entering one character at a time, a new cluster should form for each character entered. All intermediate clusters, which may not be valid final clusters, should be supported in the font. This is the case with our prototype Khitan font.

For a linear style Khitan font which is designed not to compose clusters but to display individual cluster components in linear layout, caret placement, caret movement, selection, deletion and backspacing should all be the same as for decomposed Hangul displayed using a font that does not support dynamic composition.

As is the case with decomposed Hangul, the user may not know where exactly the caret is positioned when using arrow keys to move through a cluster. There is no way of solving that at the encoding level, but it can be solved at the application level, for example BabelPad indicates the character at the current cursor position on the status bar.

1.2 Additional Comments in Response to L2/18-168

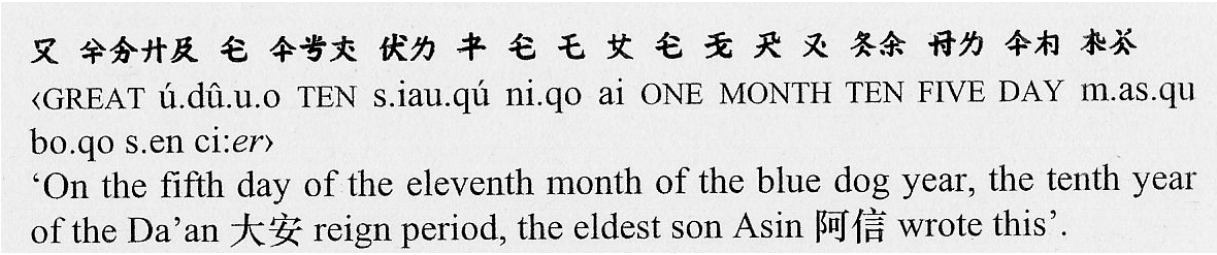
The following are additional comments in response to L2/18-168 "Recommendations to UTC #155 April-May 2018 on Script Proposals" (2018-04-28) section 14.

1. Regarding the use of Space character

- Original KSS texts show a visible narrow gap between clusters, and between clusters and standalone characters, and in many but not all cases there is also a narrow gap between adjacent standalone characters. Modern scholarly transcriptions of KSS text generally show a clear gap between all KSS units (whether clusters or standalone characters), as shown in the two examples below, one with handwritten characters and one with typeset characters. Therefore our model with spaces between adjacent KSS units reflects modern scholarly practice.

- If a user does not want a visible space between standalone characters then U+200B ZERO WIDTH SPACE can be used as the space character between standalone characters.
- Use of space characters to demarcate the start and end of a run of joining characters is not unusual, and is a feature of the Mongolian and Phags-pa scripts for example. Use of space characters in KSS is essentially the same as the use of space characters in other joining scripts such as Mongolian and Phags-pa.

- When KSS text is formatted in non-clustering linear layout, in order to be legible a space *must* be inserted between all KSS units (both sequences of characters that form a cluster and standalone characters), as seen in this example from Kane 2009 p. 145:


‹GREAT ú.dû.u.o TEN s.iau.qú ni.qo ai ONE MONTH TEN FIVE DAY m.as.qu bo.qo s.en ci:*er*›
'On the fifth day of the eleventh month of the blue dog year, the tenth year of the Da'an 大安 reign period, the eldest son Asin 阿信 wrote this'.

- With our proposed model it is possible to swap between clustering layout and linear layout *simply by changing the font, with no change to the sequence of encoded characters*. However, for an encoding model that has no spaces between clusters and standalone characters, it would be necessary to insert spaces when converting from clustering layout to linear layout, and to remove spaces when converting from linear layout to clustering layout. This would defeat one of the primary design goals for a KSS encoding model, which is to be able to represent KSS text in clustering layout and in linear layout with the same underlying encoding stream.

2. Regarding the use of CGJ character

- The script ad hoc report notes that "The current document recommends no format control characters, instead inserts Combining Grapheme Joiner (CGJ) between the first and second characters". This is not quite correct. KSS has two cluster patterns: the standard pattern starts with two adjacent characters (see Section 2.B below); and an alternative pattern starts with a single character (see Section 2.C below). In our model, CGJ is only required to be inserted between the first and second characters in the alternative cluster pattern (in order to distinguish it from the standard pattern), and no CGJ or other format character is required in the standard pattern. As the vast majority of KSS clusters conform to the standard pattern, CGJ would only occasionally be required.
- CGJ has the desired character properties, including inhibiting line-breaking before and after (i.e. "glue"), whereas other existing characters we considered do not have the desired character properties.
- We have produced a prototype KSS font that supports the desired behaviour with CGJ, and testing under Microsoft Windows 10 in Notepad and Microsoft Word 2016 shows that it works as expected with no side effects. We have also tested collation using the UCA on clusters with and without CGJ, and the collation results are as expected (clusters with CGJ sort before the same clusters without CGJ).
- Although we think that CGJ is suitable for our proposed usage, we can accept the encoding of a dedicated Khitan Small Script format character instead if the committees consider that would be more appropriate.

3. Regarding the UTC's preferred model

- We find it astonishing that the user community is being asked to accept the UTC's preferred model when this model is not described anywhere, and we have had to guess where the proposed format characters are intended to be positioned, and what behaviour they are supposed to have. We have described our proposed model in great detail, and have developed prototype fonts with OpenType features to implement clustering behaviour, whereas the advocates of an alternative model have written no documents describing how their model is intended to work, and have not developed any prototype fonts for this model.
- The UTC script group seem to think that IMEs can solve our objections to their model requiring a format character to be inserted between every adjacent character in a cluster. This is a very simplistic view, and ignores the question of who is going to develop and distribute an IME for a minor historic script. We have considerable experience in creating keyboard layouts and creating IMEs using Microsoft Windows Text Services Framework (TSF), and in our opinion it would be very difficult to design a usable IME for KSS clusters because a large proportion of KSS characters have no identified phonetic reading (and some characters have the same phonetic readings), and so the only way to allow users to enter KSS sequences as text using TSF would be to assign arbitrary text values to each KSS character (e.g. 3-letter mnemonics), but this would be very unlikely to be accepted by users as they would have to memorise hundreds of mnemonic sequences. As text-based IMEs are not practical for KSS, it is likely that users will enter KSS characters individually using a character picker or a stroke-based IME for individual characters, and so the burden of entering format characters cannot be easily removed from the end user.
- Difficulty of entering KSS text with about 40% format characters is certainly a problem, but it is not our main objection to the UTC's preferred model.
- We think that the main problem will come when editing KSS text, as cutting, pasting and deleting sequences of KSS characters interleaved with invisible format characters will result in a high probability of format characters becoming inadvertently misplaced or deleted, and it may not always be obvious to the user what has gone wrong and how to repair the problem.
- The UTC's preferred model allows for the construction of orthographically-invalid clusters (i.e. clusters that have arbitrary shapes with more than two adjacent characters which do not ever occur in real Khitan texts). How would the rendering system and font deal with clusters that are invalid according to Khitan orthography but are legal according to the encoding model? How would the rendering system deal with multiple format characters between two KSS graphic characters?
- Invisible format characters are problematic at the best of times, but having an encoding model that mandates about 40% of the text comprising invisible format characters will be a nightmare for users. We strongly believe that a model with as few format characters as possible serves the user community best. Our model only requires occasional use of CGJ in a fixed position (between 1st and 2nd character in a cluster), and no format characters for the vast majority of clusters, so there is very little scope for unexpected display or editing issues.

2. *Structure of the Khitan Small Script*

Khitan Small Script is written using a mixture of standalone characters and clusters of two to eight characters, as outlined below.

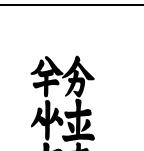
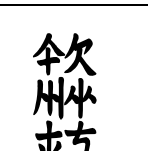
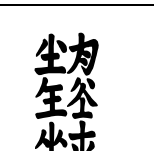
A. Standalone Characters

Only certain KSS characters can occur in isolation as standalone characters, and they are mostly logograms representing numbers, calendrical terms, kinship terms, etc. Dates usually include sequences of several standalone characters, but in other contexts standalone characters are much less frequent than clusters. Some standalone characters only occur by themselves, but others may also occur within a cluster. Examples of standalone characters are shown in Fig. 1 (p. 8 below).

B. Type A Clusters

Two types of cluster are attested in extant KSS texts. The standard cluster starts with two adjacent KSS characters and finishes with two adjacent characters or a single centred character. The seven patterns shown in Table 1 are attested.

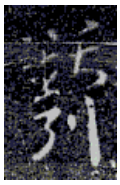
Table 1: Type A Cluster Patterns

2	3	4	5	6	7	8
①②	①② ③	①② ③④	①② ③④ ⑤	①② ③④ ⑤⑥	①② ③④ ⑤⑥ ⑦	①② ③④ ⑤⑥ ⑦⑧
						

C. Type B Clusters

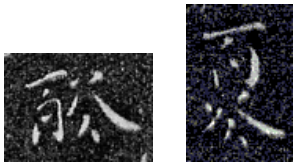
Occasionally an alternative cluster pattern, starting with a single centred character, is encountered. This pattern is most usually found with two or sometimes three characters, and only very rarely with more than three characters. Only a very small proportion of clusters conform to this pattern. Some examples of Type B clusters are shown in Table 2.

Table 2: Type B Cluster Patterns

2	2	2	2	3	3	4
① ②	① ②	① ②	① ②	① ②③	① ②③	① ②③ ④
						
土北	百爻	小尖	曲爻	一狀	小炎	小竹列

There do not seem to be any semantic differences between clusters that start with two adjacent characters (Type A) and clusters that start with a single centred character (Type B), and it seems to be an aesthetic choice. Cluster Type B is preferred for certain characters, for example 一 'north' and 小 'south' (小 is flattened at the start of a cluster).

Some two-character words are attested in both cluster patterns. For example, in the *Epitaph for Emperor Daozong* (1101) the word 百爻 [18B1D 18C66] *mó.er* 'mother, wife' is written four times with the two characters horizontally aligned (Cluster Type A), and once with the two characters vertically aligned (Cluster Type B):



An iteration mark (〻) also occurs, either by itself to stand for the preceding cluster or as the first component of a cluster where it indicates that the preceding cluster is repeated with the suffix attached to the iteration mark. For example 〻 stands for 〻 〻, and 〻 stands for 〻 〻.

Fig. 1: Start of the *Epitaph for Emperor Daozong* (1101)

Standalone characters highlighted in red

3. *Proposed Clustering Model*

Because the pattern of clusters in KSS is fixed, there is no need to put a format character between each character in a cluster in order to determine the cluster pattern and the position of glyphs in this pattern. If the position of the first character in a cluster is known (adjacent to the second character or centred by itself) then the relative positions of all following characters in the cluster can be trivially determined.

The original proposal for two format characters was based on the idea that every cluster would need to be prefixed by a format character in order to specify the pattern type (starting with two adjacent characters or starting with a single centred character). However, after further consideration and consultation with users, we now consider that there is no need for any dedicated format characters for KSS layout and rendering.

We consider that the simplest solution for end users, as well as for the layout system and font, is to have automatic cluster formation of any string of contiguous KSS characters without the intervention of format characters. In order for automatic clustering to occur, sequences of characters that form a cluster and standalone characters should be separated by whitespace. This reflects the behaviour seen in original KSS texts where there is usually visible whitespace between clusters and/or standalone characters. The space character used can be left to the user to decide, but we recommend using the ordinary U+0020 space but with a relatively narrow glyph width in a KSS font. Users could use ZWSP if no gap is desired between clusters and/or standalone characters.

The vast majority of clusters in KSS texts are Cluster Type A, so the proposed automatic clustering mechanism would produce clusters starting with two adjacent characters by default. For Cluster Type B starting with a single centred character, we propose inserting U+034F COMBINING GRAPHEME JOINER (CGJ) after the first character in the cluster to indicate this cluster pattern. We considered using ZWNJ, but CGJ is more appropriate as it inhibits line breaking before and after, which is the desired behaviour for KSS. Using CGJ will also cause the desired sorting behaviour with the Unicode Collation Algorithm, which is to sort Cluster Type B sequences before Cluster Type A sequences where both sequences comprise the same graphic characters, e.g. <18B1D 034F 18C66> should sort immediately before <18B1D 18C66>.[1]

[1] In this document KSS clusters embedded in English text are typeset with the Khitan Small Rotated font which renders the text in rotated orientation. The Khitan Small Vertical font is not used as it would result in huge gaps between lines.

4. *Presentation Format and Font Implementations*

Modern scholars of Khitan generally typeset or handwrite Khitan Small Script text in one of the three presentation formats described below.

A. Running Vertical Format

In this format, KSS text is laid out in vertical columns running right to left across the page, reproducing the format used in original Khitan sources. This format is often used for transcribing KSS texts, as shown in the example given in Fig. 2.

Fig. 2: Transcription of the Epitaph for Court Attendant Yelü Yongning (1088) in Zheng 2002, p. 66

B. Embedded Vertical Format

In this format, KSS standalone characters and clusters are written horizontally in vertical orientation, either on a line by themselves or embedded in horizontal text in a different language.

An example of vertical KSS clusters embedded in horizontal Chinese text is shown in Fig. 3. In this example the clusters are vertically aligned at the bottom, but other alignments are also found. For example, Fig. 4 shows vertical KSS text embedded in horizontal Japanese with top alignment of KSS clusters, and Fig. 5 shows a Russian paper with vertical KSS text aligned vertically along the centre.

Fig. 3: Vertical KSS clusters embedded in horizontal Chinese text in Bao 2002, p. 18

字形	业及 仝	业及 子圭 亏	业及 子圭 丰	业及 闲屮 关	业及 闲屮 伏	业及 子叐 孖	业及 闲屮 叐	总数
出现行数	9,16,17,18,19	6,27	6,9,22,24	8	9	11,22	12	12
出现次数	5	2	4	1	1	3	1	17
百分比	29.4	11.8	23.5	5.9	5.9	17.6	5.9	100%

由上表可以看出“除”意的 23.5% 义的词当中，“业及 仝”形式出现的最多。占总数的 29.4%。其次是 业及 子圭 丰 占 23.5%。出现次数最少的是 业及 闲屮 关，业及 闲屮 伏，业及 闲屮 叐 分别只出现过一次。契丹小字中的动词附加成分，有如 业及 闲屮 叐 中的“叐[ən]”动词陈述式附加成分和 业及 子圭 丰 中的 丰[ai]副动词附加成分以及 业及 子叐 孖 中的“孖[su～sun]”形动词中附加成分，还有如 业及 子圭 亏 中的“亏[al]”之类及物与不及物等的区别。如上所述，我们可以进一步得知契丹小字的动词附加成分具有各种不同的形式以及不同的变体。而出现频率根据需要有所不同。这一点与其他语言的情况类似。

在此文的写作过程中，陈乃雄 教授，清格尔泰教授，

Fig. 4: Vertical KSS clusters embedded in horizontal Japanese text in Aisin Gioro 2006, p. 48

冠するものは、みな奚可汗帳に属し、漢姓は「蕭」氏である。

「令爿／公爿」を「迪輦」と推測する説がある[*31]が、契丹小字墓誌に出現する漢文と対応する人
　　丬东　丬东

名「迪輦」（あるいは「敵輦」）は均しく「令用／公用」*dirənに作り、「令爿／公爿」*dəliæとは
　　当　　与　　丬东　丬东

明らかに同じ単語ではない。

以下に契丹小字墓誌でこの姓氏が出現する箇所を列挙しよう。

(1)『永寧郎君墓誌銘』の墓主は耶律氏で、その女性親族は多く「令爿」*dəliæの姓氏を冠する奚
　　丬东

人である。

　　屮冋　九火　丙　几　令爿　六夫　尤　口北　巫立　冊爻……
　　丬东　　伏　　夭　　朴

　　（令　公　妻　迭剌　達里乙林免　奚　可汗　帳の）『永5』

　　又及　令用　今犇　九亦　丙　几　令爿　发冋　戈谷　尔火　口北　巫立　冊爻　朴丹　王丟　又土　火关
　　令　　丬东　　伏　夭　朴　企　九

　　（長子　迪烈德　將　軍　妻　迭剌　迎　日　娘子　奚　可汗　帳の　帝室己　守　期
　　尔火　圣朴　冋力
　　娘子　二人の　子）『永12～13』

Fig. 5: Vertical KSS clusters embedded in horizontal Russian text in Zaytsev 2011, p. 144

Таблица 4

Наименование киданьского государства

Н 176 л. 9, стк. 6:	*Наименование киданьского государства, записанное большим киданьским письмом*: 夾 平弓 朴英 宋刘 国
	То же, записанное малым киданьским письмом: 又 令兮 分 玉刘 关关 几亦
	транскр. mos diau-d hulʤi[29] kitai gur
	кит. 大 中央 胡里只 契丹 國
	перевод Великое центральное *hulʤi* киданьское государство (Лю Фэн-чжу, 2003, с. 75–77, 89; Лю Фэн-чжу, 2006, с. 52–54; Айсинь Гиоро, 2009a, с. 166–167, 192–193).

C. Horizontal Linear Format

In horizontal linear format KSS characters are written horizontally left-to-right, with clusters decomposed into linear strings of characters. Sequences of characters that form a cluster in the original Khitan source are separated by whitespace. KSS text is presented this way in Daniel Kane's 2009 book, *Kitan Language and Script* (see Fig. 6), in Wu & Janhunen's 2010 book *New Materials on the Khitan Script*, and in some works by Aisin Gioro Ulhicun published after 2010.

Fig. 6: KSS text written in horizontal linear format in Kane 2009, pp. 186–187

6.3 Kitan text of the Langjun inscription

[1] 又 山 九夾村 丞立夬 与 而礼 重今 九用 小考永 戊为夫
GREAT GOLD g.úr:en qa.ha:an deu cau.ji hur.ú g.iŋ l.iau.u ś.a.rí[2]

---

[2] Wang Jingru identified 又 as 大 'big, great' and 山 as 金 'gold, Jin dynasty'. *Research* identified 九夾村 ‹g.úr:en› 'of the country' and 丞立夬 ‹qa.ha:an› 'of the khan'. 与 is 弟 *di* 'younger brother'. 而礼 重今 ‹cau.ji hur.ú› corresponds to 都統 *dutong* 'military commissioner'. 而礼 ‹cau.ji› is 'war', related to 而夾 ‹cau.úr› 'war, battle, army ' and 重今 ‹hur.ú› is 'controller'. 九用 小考永 ‹g.iŋ l.iau.u› corresponds to 經略 *jinglüe* 'military commissioner'. 戊为夫 ‹ś.a.rí› is 郎君 *langjun* 'court attendant'. 又 山 clearly corresponds to 大金 Da Jin 'The Great Jin [State]'; in Jurchen the word for 'gold' was **alčun*. There has been considerable discussion on the graph 山 and its variant 山. Wu Yingzhe has proposed that the 'dotted forms' indicate grammatical gender i.e. they are used with masculine nouns. 金 also means 'metal', which is one of the five elements, but 山 does not correspond to 'metal'. 九夾村 ‹g.úr:en›: 九夾村 ‹g.úr›

TEXTS IN THE KITAN SMALL SCRIPT 187

[2] 未木 乃今 兆为夬 火 与伏
c.ar am.se REGION.a.an ui eu.iń[3]

[3] 小丑 戊为夬 刃尘(当?) 兴 又出又 今仕兴
l.iáŋ ś.a.an ja.cẻn.i m.án.ún se.mu.i[4]

Different font implementations will be required in order to realise these three different presentation formats. Every KSS font should be able to handle the same input stream of encoded KSS characters, but will render the KSS characters in particular ways.

Andrew West has developed working prototypes of three types of KSS font, corresponding to the three presentation formats described above:

- **Khitan Small Rotated** — a prototype font for running vertical format;
- **Khitan Small Vertical** — a prototype font for embedded vertical format;
- **Khitan Small Linear** — a prototype font for horizontal linear format.

These fonts are derived from a font designed by Jing Yongshi, and can be provided to WG2 and UTC committee members on request.

These fonts have been verified as working as expected on Microsoft Windows 10 with BabelPad, Notepad, Microsoft Word 2016, and Microsoft Edge browser (see Appendix 2). A test page using a WOFF version of Khitan Small Rotated is available at http://www.babelstone.co.uk/Fonts/KSS_Test.html.

These three types of font are described below, and a sample image of the same KSS text using the provisional UCS code points is given for each font.

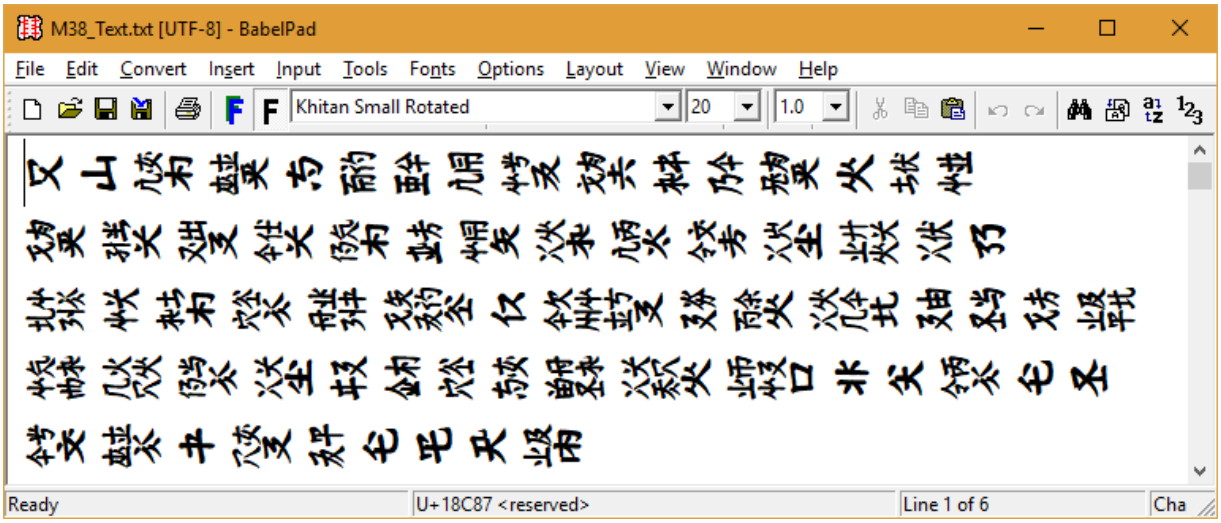
A. Rotated Vertical Font

The typical KSS font will be intended to display KSS text in vertical layout, as is usually the case in original Khitan sources. However, in order to display running vertical text on computers, the font needs to provide glyphs for individual characters and clusters that are rotated counterclockwise. In plain text applications the font will produce horizontal lines of rotated glyphs, but in rich text environments the horizontal lines can be rotated clockwise to produce vertical columns of KSS text. For example, Microsoft Word 2016 supports vertical rotation of text (Layout > Text Direction), and web pages support vertical layout using CSS stylesheets.

This is the same mechanism that is used in Unicode fonts for other vertical scripts such as Mongolian and Phags-pa.

Fig. 7 shows the display of KSS text using the Khitan Small Rotated font in a plain text application, where the lines are displayed horizontally, and the glyphs are rotated counterclockwise. Examples of rotated KSS text displayed in running vertical format with HTML/CSS are given in Appendix 2.

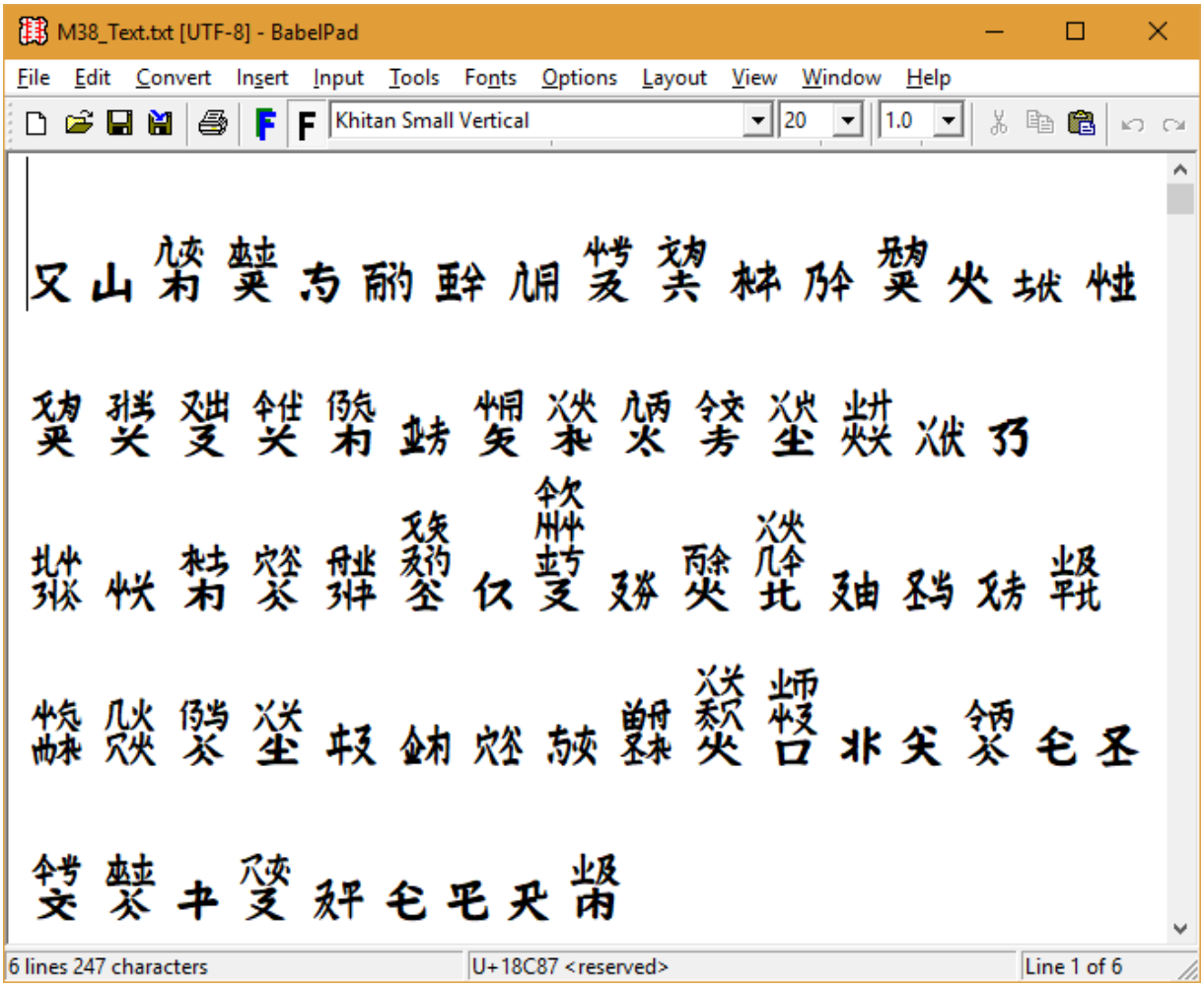
Fig. 7: Sample of KSS text set with the Khitan Small Rotated font



B. Embedded Vertical Font

A KSS font could also provide glyphs for individual characters and clusters in vertical orientation. This would allow individual characters or clusters to be embedded in horizontal text with vertical orientation, either in a plain text or rich text environment. However, this font could not be used to produce running vertical text. Moreover, the line spacing of the font has to accommodate the tallest glyph in the font, which means that when even a single standalone character is embedded in a paragraph of English or Chinese text there will be a huge line gap between the line above and the line with embedded KSS characters.

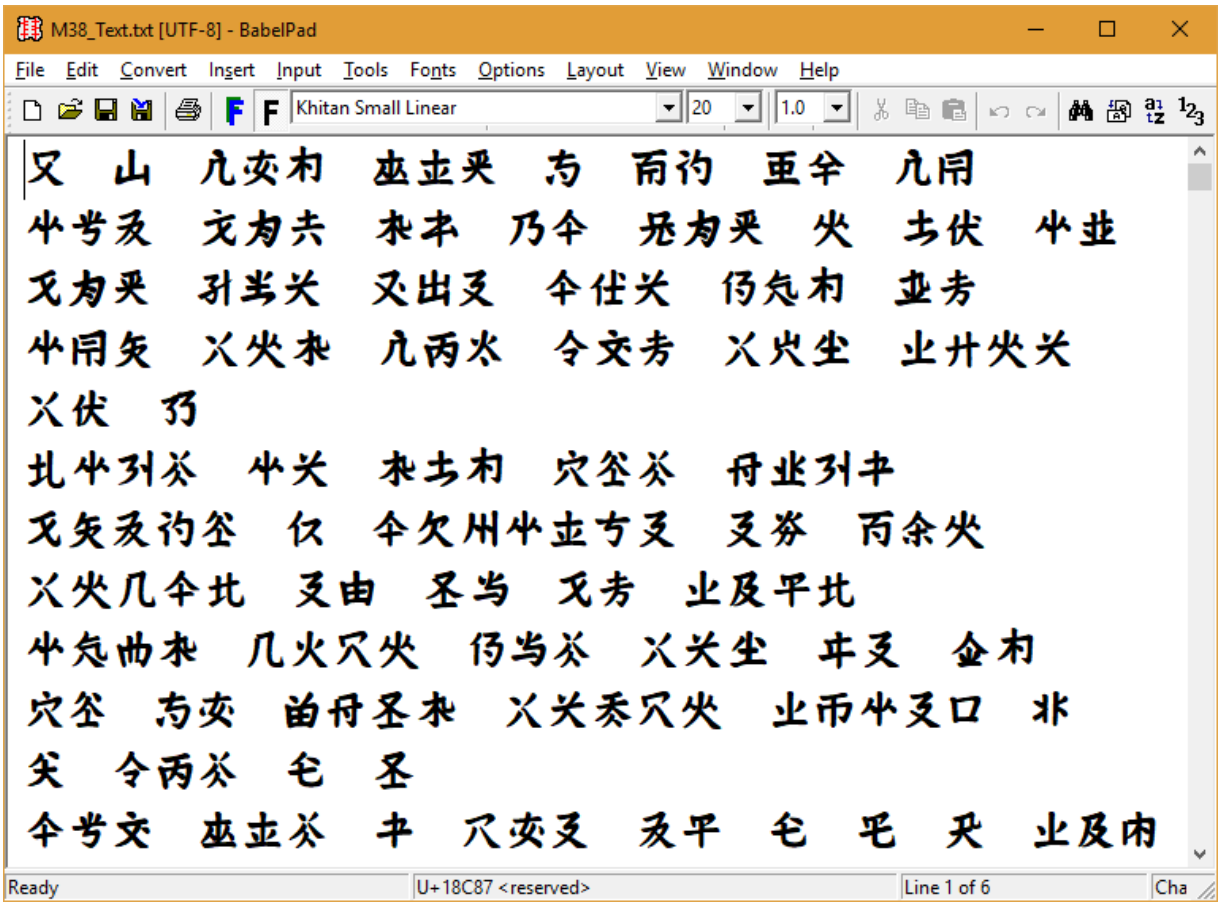
Fig. 8: Sample of KSS text set with the Khitan Small Vertical font



C. Linear Font

A linear font simply provides a single vertically-oriented glyph mapped to each encoded KSS character. No clustering of sequences of KSS characters is enabled, and no OpenType features are required to be supported. Because KSS clusters are separated by whitespace, linear strings of KSS characters that would form a cluster in a rotated vertical font or embedded vertical font are separated by whitespace in a linear font.

Fig. 9: Sample of KSS text set with the Khitan Small Linear font



5. *OpenType Implementation*

For the rotated vertical and embedded vertical fonts, which combine sequences of KSS characters to form clusters, two possible OpenType implementations can be considered. These are not relevant to a linear font as it has a one-to-one mapping between characters and glyphs, and no OpenType transformations are required.

In both implementations a run of KSS characters is processed by the rendering system. A KSS run comprises a contiguous sequence of KSS graphic characters, including zero or many CGJ characters (valid KSS clusters should only have zero or one CGJ after the first character in the run, but the implementation should be able to handle runs with more than one CGJ or a single CGJ in a position other than between the first two graphic characters). The boundaries of the run are marked by SOL, EOL, any space character, any punctuation character, or any character that is not a KSS graphic character or CGJ.

A run comprising a single KSS character uses the default glyph mapped to the character in the *cmap* table. For a run of two or more characters, cluster-forming glyphs are substituted for the default glyphs using OpenType features.

A. Precomposed Implementation

For this implementation the KSS font provides precomposed glyphs for all supported clusters, including incomplete clusters required during user input of KSS text. As there are only about 2,000 to 3,000 attested cluster sequences, a font would need about 4,000 to 5,000 precomposed glyphs for partial and complete clusters, which is a large but manageable number.[2]

The font would have a single GSUB table using the `ccmp` "Glyph Composition / Decomposition" feature that substitutes sequences of KSS characters for the precomposed glyphs.

The advantages of this implementation are that is:

- Is simple to implement in the font;
- Works the same for both rotated vertical and embedded vertical fonts;
- Does not require any special assistance from the rendering engine, and so should work with rendering systems that have no knowledge of the Khitan Small Script;
- Allows the font designer to design cluster glyphs where the various components are harmoniously integrated together (e.g. with strokes from one component crossing into the space occupied by another component without clashing).

The disadvantages are that it:

- Only supports defined clusters, and cannot deal with arbitrary cluster sequences;
- Requires a large font size with thousands of glyphs.

[2] At present there is no single source that lists all attested KSS clusters.

An extract of the OpenType table design used in the prototype fonts Khitan Small Rotated and Khitan Small Vertical is shown below (the same OpenType lookups are used in both fonts):

```
script DFLT {
  feature "Glyph Composition/Decomposition";
}

feature "Glyph Composition/Decomposition" ccmp {
  lookup Polygrams;
}

lookup Polygrams {
  sub u18B01 cgj u18C2D u18C40 -> u18B01.C2D.C40.centred;
  sub u18B1D cgj u18C66 -> u18B1D.C66.centred;
  sub u18B54 cgj u18B62 -> u18B54.B62.centred;
  sub u18BA0 cgj u18C64 -> u18BA0.C64.centred;
  sub u18B1C u18B5E u18C37 -> u18B1C.B5E.C37;
  sub u18B1C u18B5E -> u18B1C.B5E;
  sub u18B1C u18CCD u18C04 -> u18B1C.CCD.C04;
  sub u18B1C u18CCD -> u18B1C.CCD;
  sub u18B1D u18C66 -> u18B1D.C66;
  sub u18B25 u18C5B -> u18B25.C5B;
  sub u18B25 u18CCD u18C04 -> u18B25.CCD.C04;
  sub u18B25 u18CCD -> u18B25.CCD;
  sub u18B39 u18BFA u18BA2 -> u18B39.BFA.BA2;
  sub u18B39 u18BFA -> u18B39.BFA;
  sub u18B50 u18C31 -> u18B50.C31;
  sub u18BA1 u18C46 -> u18BA1.C46;
  sub u18BFA u18B85 u18B42 u18BE2 -> u18BFA.B85.B42.BE2;
  sub u18BFA u18B85 u18B42 -> u18BFA.B85.B42;
  sub u18BFA u18B85 -> u18BFA.B85;
}
```

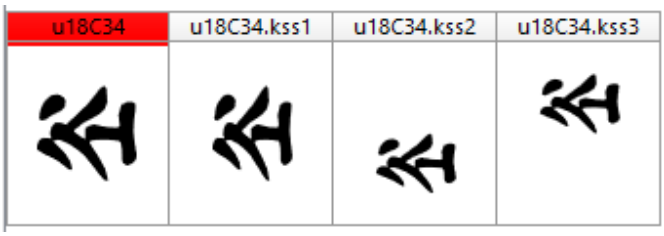
B. Dynamic Composition Implementation

This implementation is only applicable to the rotated vertical font. It would be very difficult to implement dynamic composition of arbitrary clusters in an embedded vertical font, and such an implementation is not discussed here. As this implementation depends on OpenType features that do not yet exist, a prototype font has not been developed.

For this implementation the KSS font provides three positional glyph forms for each KSS character (in addition to the basic full-sized glyph):

A. Centred glyph used for the initial character in Type B clusters or for the final character where there are an odd number of characters (Type A cluster) or an even number of characters (Type B cluster);
B. Bottom-aligned glyph (in font orientation) used for odd-numbered non-final characters in Type A clusters or even-numbered non-final characters in Type B clusters;
C. Top-aligned glyph (in font orientation) used for even-numbered characters in Type A clusters or odd-numbered non-initial characters in Type B clusters.

A rendering engine would be required to apply specific OpenType features to the font in order to substitute the correct positional glyph form for each KSS character in a run. There are no existing OpenType features that can be used, so three new OpenType features would need to be registered, which we refer to as `kss1`, `kss2` and `kss3`. The default glyph form and three positional forms for U+18C34 𘐴 in the prototype font are shown below (note that the glyphs are rotated counterclockwise in the font to enable vertical layout in applications that support vertical orientation of text):



The kss1 and kss2 glyphs are spacing, but the kss3 glyph has no advance width, with the result that a kss3 glyph is rendered immediately above the preceding kss2 glyph.

The following pseudocode shows how a rendering engine might process a run of KSS characters (including one or more CGJ characters). For runs of a single KSS character no action is required as the normal full-sized glyph in the *cmap* mapping should be displayed. For runs of two or more characters, the `kss1`, `kss2` or `kss3` feature will be applied to the KSS characters to produce the expected shaping.

```
if ( run.length > 1 ) {
{
      for ( int i = 0; i < run.length; i++ )
      {
            if ( i % 2 == 1 )
            {
                  font.applyFeature( run[i], kss3 );
            }
            else
            {
                  if ( ( ( i + 1 ) == run.length ) || ( run[i+1] == CGJ ) )
                  {
                        font.applyFeature( run[i], kss1 );
                        i++;
                  }
                  else
                  {
                        font.applyFeature( run[i], kss2 );
                  }
            }
      }
}
```

This implementation supports arbitrary clusters of arbitrary length, and does not require the font designer to create thousands of precomposed glyphs for clusters. However, creating clusters by piecing together positional components does not give an ideal visual result, and in a high-quality font additional substitutions and/or positioning may need to

be applied to ensure that the components forming a cluster are harmoniously integrated. The following points need to be considered:

A. The relative widths of adjacent components may vary depending on the components involved. For example, when 氵 + 叅 form a cluster, the left component should occupy less than half the cluster width, and the right component should occupy more than half the cluster width (滲), but in other cluster combinations the right component would need to be thinner in order to accommodate the left component.
B. In many cases the best visual results are achieved by allowing strokes from one component to cross into the space occupied by another component, and to do this without clashing would require individual tailoring of adjacent glyphs.
C. The length of components in a cluster may vary depending on the number of components in a cluster. Clusters comprising just two adjacent components are normally about the same length as a single standalone character, but in clusters with three or more components each component is shorter than a standalone character, and as a rule of thumb the longer a cluster is the shorter each component (or pair of parallel adjacent components) is. It would be very difficult to mimic this behaviour using dynamic composition.

For these reasons, a high-quality KSS font may need to use a mixture of precomposed glyphs and dynamic composition. Precomposed glyphs could be used for important and frequently-occurring clusters, whereas dynamic composition could be used for all other arbitrary clusters.

Note that the above glyph design issues would still be present with a model that inserts a horizontal joiner or vertical joiner format character between each graphic character in a cluster.

6. *Line Breaking Considerations*

There should be no line break opportunity within a KSS cluster. As the proposed automatic clustering model requires standalone characters to be separated by a space character, all sequences of two or more KSS characters necessarily form a cluster. Therefore there should be no line break opportunity between any two adjacent KSS characters.

Unicode Standard Annex #14 "Unicode Line Breaking Algorithm" should be updated with a new rule to specify that there is no line break opportunity between KSS characters. This would require a new line breaking class for KSS characters, which we provisionally call KS, and the following rule: KS × KS. There would be a line break opportunity between KS and any other non-gluing class.

CGJ may also occur in a cluster, but CGJ prohibits line breaking between it and the preceding character and following character, so it does not provide a line break opportunity when used in a KSS cluster.

Even though line-breaking might be acceptable within cluster sequences rendered in horizontal linear format, because the line-breaking algorithm is independent of the font used to display the text, KSS text displayed with a linear font would also have no line breaks within sequences corresponding to a cluster. If desired, line break opportunities could be manually inserted into linear clusters using ZWSP.

7. *Bibliography*

N4725R. Andrew West, Viacheslav Zaytsev, Michael Everson. *Towards an Encoding of the Khitan Small Script*. ISO/IEC JTC1/SC2/WG2 N4725R (L2/16-113R). 2016-05-21. http://www.unicode.org/wg2/docs/n4725r-khitan-small-script.pdf

N4736. Deborah Anderson, SEI, UC Berkeley. *Summary of Meeting on Khitan Scripts, 20 August 2016 (Yinchuan, China) - Ad Hoc Report #1*. ISO/IEC JTC1/SC2/WG2 N4736 (L2/16-243). 2016-08-20. http://www.unicode.org/wg2/docs/n4736-khitan-meeting-1.pdf

N4737. Deborah Anderson, SEI, UC Berkeley. *Summary of Meeting on Khitan Scripts, 22 August 2016 (Yinchuan, China) - Ad Hoc Report #2*. ISO/IEC JTC1/SC2/WG2 N4737 (L2/16-244). 2016-08-22. http://www.unicode.org/wg2/docs/n4737-khitan-meeting-2.pdf

N4738R2. Sun Bojun, Wu Yingzhe, Jing Yongshi, Jiruhe, Viacheslav Zaytsev, Andrew West, and Michael Everson. *Final proposal to encode the Khitan Small Script in the SMP of the UCS*. ISO/IEC JTC1/SC2/WG2 N4738R2 (L2/16-245R). 2016-09-17. http://www.unicode.org/wg2/docs/n4738r2-khitan-small.pdf

N4768. Lisa Moore, Unicode Consortium. *Summary of Ad Hoc Meeting on Khitan Small Script, 28 September 2016*. ISO/IEC JTC1/SC2/WG2 N4768 (L2/16-338). 2016-09-28. http://www.unicode.org/wg2/docs/n4768-KhitanSmallScriptAdhoc.pdf

N4775. Andrew West, Michael Everson, Viacheslav Zaytsev. *Discussion of Cluster Formation in Khitan Small Script*. ISO/IEC JTC1/SC2/WG2 N4775 (L2/16-296). 2016-10-27. http://www.unicode.org/wg2/docs/n4775-khitan-small-model.pdf

L2/18-168. Deborah Anderson, Ken Whistler, Roozbeh Pournader, Lisa Moore, Liang Hai, Chris Chapman, and Richard Cook. *Recommendations to UTC #155 April-May 2018 on Script Proposals*. L2/18-168. 2018-04-28. http://www.unicode.org/L2/L2018/18168-script-rec.pdf

Aisin Gioro 2006. Aishingyoro Uruhichun 愛新覚羅 烏拉熙春 [Aisin Gioro Ulhicun]. *Kittanbun boshi yori mita ryōshi* 契丹文墓誌より見た遼史 [History of the Liao as seen from Khitan Epitaphs]. Kyōto 京都: Shōkadō shoten 松香堂書店, 2006. ISBN 4-87974-601-0

Bao 2002. Bāo Liánqún 包联群. "«Nánshànbùzhōu Dà Liáo guó Gù Díliè wáng mùzhìwén» de bǔchōng kǎoshì" 《南赡部洲大辽国故迪烈王墓志文》的补充考释 [A Supplementary Study of the "Epitaph of the Late Prince Dilie of the Great Liao in the Southern Continent of Jambudvīpa"]; *Nèiménggǔ dàxué xuébào* (*rénwén shèhuì kēxué bǎn*) 内蒙古大学学报 (人文社会科学版) [*Journal of Inner Mongolia University* (*Humanities and Social Sciences*)] vol. 34 no. 3 (May 2002): 15–19.

Kane 2009. Kane, Daniel. *The Kitan Language and Script*. Leiden ; Boston: Brill, 2009. (Handbook of Oriental Studies = Handbuch der Orientalistik. Section 8: Central Asia; Volume 19). ISBN 978-90-04-16829-9

Wu & Janhunen 2010. Wu Yingzhe [Wú Yīngzhé 吴英喆] and Juha Janhunen. *New Materials on the Khitan Small Script: A Critical Edition of Xiao Dilu and Yelü Xiangwen*. Folkestone: Global Oriental, 2010. (Corpus Scriptorum Chitanorum; I). (Languages of Asia Series; Volume 9). ISBN 978-1-906876-50-0

Zaytsev 2011. Zaytsev, Viacheslav (Зайцев, Вячеслав Петрович). "Рукописная книга большого киданьского письма из коллекции Института восточных рукописей РАН" [A manuscript codex written in the Khitan Large script from the collection of the Institute of Oriental Manuscripts, Russian Academy of Sciences]; *Письменные памятники Востока* [Written Monuments of the Orient] no. 2(15) (autumn–winter 2011): 130–150. ISSN 1811-8062 http://www.orientalstudies.ru/eng/images/pdf/a_zaytsev_2011.pdf

Zheng 2002. Zhèng Xiǎoguāng 郑晓光. "Qìdān xiǎozì «Yēlǜ Yǒngníng lángjūn mùzhìmíng» kǎoshì" 契丹小字《耶律永宁郎君墓志铭》考释 [A Study of the "Epitaph for Court Attendant Yelü Yongning" in the Khitan Small Script]; *Mínzú yǔwén* 民族语文 no. 2 (2002): 63–69.

8. *Appendix 1: Examples of KSS Text Input Sequences*

Examples of text input using the precomposed OpenType implementation for the Khitan Small Rotated font are shown below. Table 3 and Table 4 show input of cluster sequences that are supported in the font, whereas Table 5 and Table 6 show input of unsupported cluster sequences. The unsupported clusters are not rendered correctly, but they do provide a reasonable fallback that should be understandable to a user who is familiar with the Khitan Small Script.

Table 3: Text Input for Type A Cluster

Step	Enter	Backing Store	Display
1	18C31 尘	18C31 尘	尘
2	18BFA 为	18C31 18BFA 尘为	尘为
3	18C02 生	18C31 18BFA 18C02 尘为生	尘为生
4	18C34 尘	18C31 18BFA 18C02 18C34 尘为生尘	尘为生尘
5	18C44 屮	18C31 18BFA 18C02 18C34 18C44 尘为生尘屮	尘为生尘屮
6	18B42 圡	18C31 18BFA 18C02 18C34 18C44 18B42 尘为生尘屮圡	尘为生尘屮圡
7	18BFA 为	18C31 18BFA 18C02 18C34 18C44 18B42 18BFA 尘为生尘屮圡为	尘为生尘屮圡为
8	18BA2 出	18C31 18BFA 18C02 18C34 18C44 18B42 18BFA 18BA2 尘为生尘屮圡为出	尘为生尘屮圡为出

Table 4: Text Input for Type B Cluster

Step	Enter	Backing Store	Display
1	18BA6 小	18BA6 小	小
2	034F ◌͏	18BA6 034F 小◌͏	小
3	18B42 丈	18BA6 034F 18B42 小◌͏丈	小丈
4	18B85 方	18BA6 034F 18B42 18B85 小◌͏丈方	小丈方
5	18B42 丈	18BA6 034F 18B42 18B85 18CB2 小◌͏丈方刂	小丈方刂

Note that under Microsoft Windows an unused CGJ is stripped out of the rendering stream, and so in Step 2 the CGJ is invisible even when the font has a visible glyph mapped to U+034F. This is the same behaviour that would be seen for any formatting character.

Table 5: Text Input for an Unsupported Cluster

Step	Enter	Backing Store	Display
1	18C3A 令	18C3A 令	令
2	18C17 仕	18C3A 18C17 令仕	令仕
3	18C34 㐅	18C3A 18C17 18C34 令仕㐅	令仕㐅
4	18B5E 扎	18C3A 18C17 18C34 18B5E 令仕㐅扎	令仕㐅扎

In this example, neither the final cluster nor the intermediate clusters are supported in the font, so no substitutions are applied, and the cluster is rendered as a sequence of separate glyphs.

Table 6: Text Input for an Unsupported Cluster

Step	Enter	Backing Store	Display
1	18B42 𠮷	18B42 𠮷	𠮷
2	18B78 𠮷	18B42 18B7B 𠮷𠮷	𠮷𠮷
3	18B1D 丙	18B42 18B7B 18B1D 𠮷𠮷丙	𠮷𠮷丙
4	18C66 𠀀	18B42 18B7B 18B1D 18C66 𠮷𠮷丙𠀀	𠮷𠮷丙𠀀

In this example, the complete cluster is not supported in the font, but a cluster for the last two characters in the sequence is supported, so the first two characters are rendered as separate glyphs and the last two characters are rendered as a cluster.

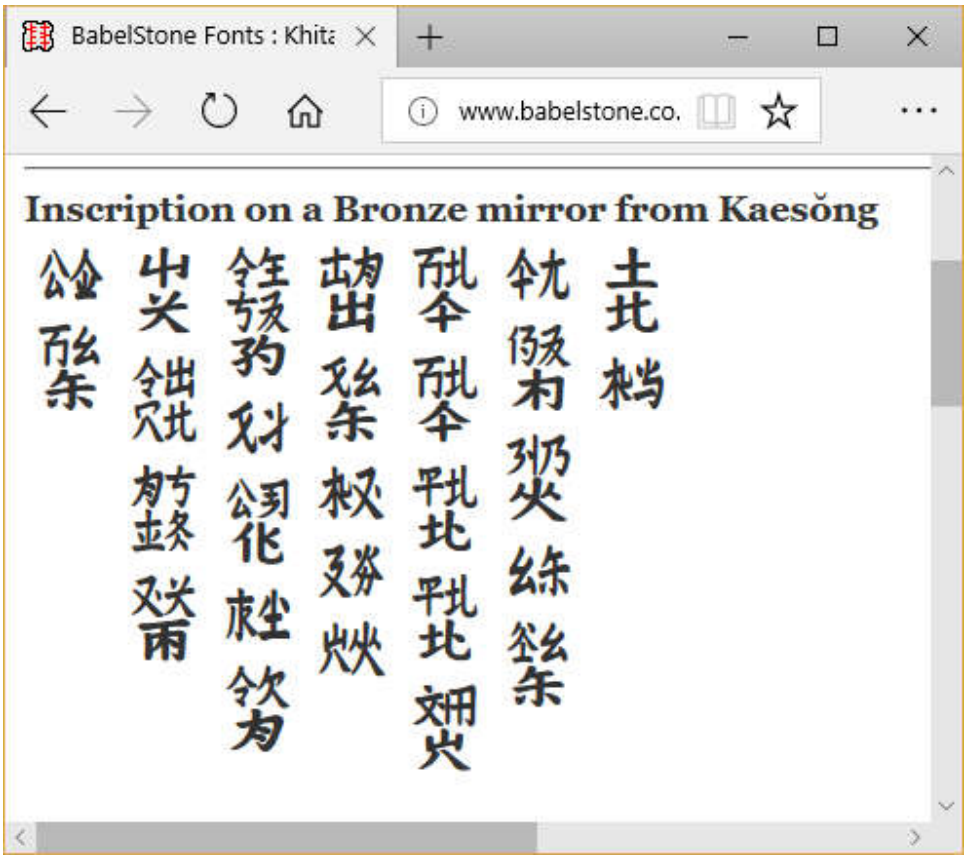
9. *Appendix 2: Samples for Prototype Font*

This appendix shows two examples of well-known Khitan Small Script texts coded in an HTML page, and rendered using the Khitan Small Rotated font in the Edge browser on Microsoft Windows 10 (see http://www.babelstone.co.uk/Fonts/KSS_Test.html). A CSS stylesheet is used to rotate the paragraphs containing each line of KSS text, so that the KSS text is displayed in vertical columns running from right to left on the web page. All clusters in these two texts are supported in the prototype font, so the result should closely reflect the source texts.

Fig. 10: Bronze mirror with KSS inscription found in Kaesŏng, Korea

[https://commons.wikimedia.org/wiki/File:Khitan_mirror_from_Korea.jpg cc-by-sa-2.0 John S Y Lee]

Fig. 11: Display on Microsoft Windows 10 with Edge browser

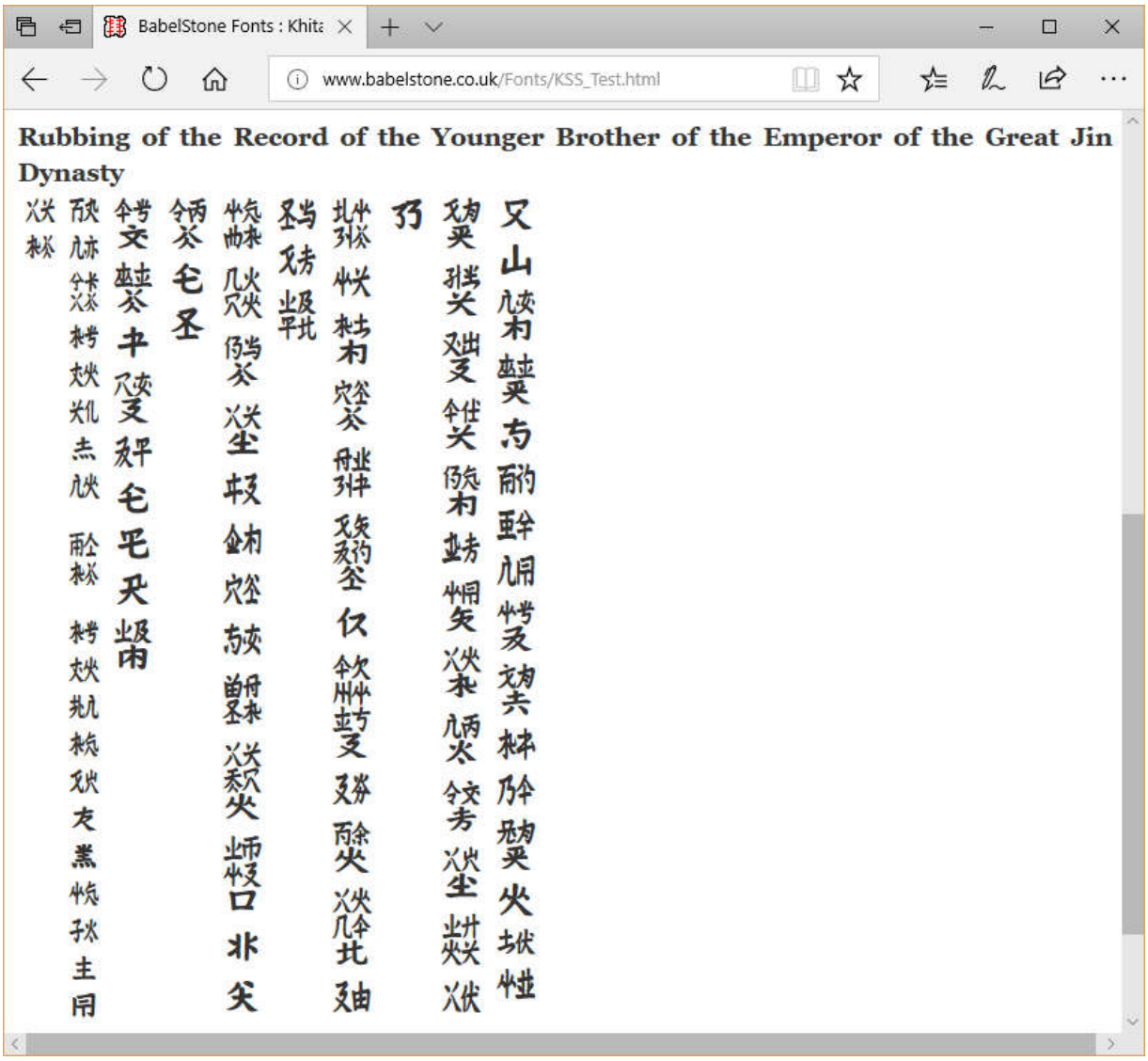


Encoding Stream:

```
<18B54 034F 18B62 0020 18CBD 18C78>
<18C37 18B74 0020 18C1D 18C9A 18CA6 0020 18CB2 18BF4 18C46 0020 18CCD 18C04
      0020 18C34 18CCD 18C04>
<18B1C 18B5E 18C37 0020 18B1C 18B5E 18C37 0020 18C80 18B5E 18B5F 0020 18C80
      18B5E 18B5F 0020 18C53 18BAF 18BA1>
<18B39 18BFA 18BA2 0020 18B25 18CCD 18C04 0020 18CBD 18C9C 0020 18CAB 18C6E
      0020 18BA1 18C46>
<18C3A 18C02 18B85 18C9A 18CB4 0020 18B25 18C5B 0020 18C2D 18CA3 18C25 0020
      18B50 18C31 0020 18C3A 18BDC 18BFA>
<18BA0 034F 18C64 0020 18C3A 18BA2 18C86 18B62 0020 18BFA 18B85 18B42 18BE2
      0020 18C9C 18C64 18B1A>
<18C2D 18C40 0020 18B1C 18CCD 18C04>
```

Fig. 12: ***Record of the Journey of the Younger Brother of the Emperor of the Great Jin Dynasty*** **(1134)**

Fig. 13: Display on Microsoft Windows 10 with Edge browser



Note that the HTML for this page has six paragraphs of KSS text, but as the lines are very long the browser has automatically broken the 2nd, 3rd, 4th and 6th lines. This automatic line breaking could be inhibited by using No-Break Space instead of ordinary spae.

Encoding Stream:

```
<18C87 0020 18B9D 0020 18C5A 18B92 18CA6 0020 18B45 18B42 18B10 0020 18B89
      0020 18B1E 18C5E 0020 18B13 18C2B 0020 18C5A 18BB0 0020 18C44 18C79
      18C9A 0020 18C56 18BFA 18B57 0020 18CBD 18C91 0020 18BF4 18C37 0020
      18C85 18BFA 18B10 0020 18C46 0020 18B55 18C15 0020 18C44 18B4F>
<18B25 18BFA 18B10 0020 18CB1 18C76 18C64 0020 18C9C 18BA2 18CAB 0020 18C37
      18C17 18C64 0020 18C1D 18C06 18CA6 0020 18CBE 18B5B 0020 18C44 18BB0
      18C0D 0020 18C65 18C46 18CBD 0020 18C5A 18B1B 18C6B 0020 18C3A 18C53
```

18B5B 0020 18C65 18BA1 18C31 0020 18BA7 18B72 18C46 18C64 0020 18C65 18C15 0020 18CAD>

<18B5E 18C44 18CB2 18C66 0020 18C44 18C64 0020 18CBD 18B55 18CA6 0020 18C58 18C34 18C66 0020 18BBD 18B98 18CB2 18C90 0020 18B25 18C0D 18C9A 18C5E 18C34 0020 18C1B 0020 18C37 18BDC 18BF9 18C44 18B42 18B85 18CAB 0020 18CAB 18C6E 0020 18B1D 18C39 18C46 0020 18C65 18C46 18BE5 18C37 18B62 0020 18CAB 18BCC 0020 18C9D 18C78 0020 18B25 18B5B 0020 18BA7 18BF6 18C80 18B62>

<18C44 18C06 18BC6 18CBD 0020 18BE5 18C6A 18C86 18C46 0020 18C1D 18C78 18C66 0020 18C65 18C64 18C31 0020 18B3F 18CAB 0020 18C40 18CA6 0020 18C58 18C34 0020 18B89 18B92 0020 18C68 18BBD 18C9D 18CBD 0020 18C65 18C64 18B2A 18C86 18C46 0020 18BA7 18B17 18C44 18CAB 18BCF 0020 18B97 0020 18C32 0020 18C3A 18B1B 18C66 0020 18C2F 0020 18C9D>

<18C37 18C79 18C53 0020 18B45 18B42 18C66 0020 18C90 0020 18C8C 18B92 18CAB 0020 18C9A 18C80 0020 18C2F 0020 18C82 0020 18CBA 0020 18BA7 18BF6 18BCE>

<18B1C 18CBC 0020 18C5A 18C55 0020 18C29 18B37 18C65 18C66 0020 18CBD 18C79 0020 18B84 18C46 0020 18C64 18C23 0020 18B59 0020 18C5A 18C46 3000 18B18 18C3D 18CBD 18C66 3000 18CBD 18C79 0020 18B84 18C46 0020 18B56 18C5A 0020 18CBD 18C06 0020 18B25 18BA1 0020 18B6F 0020 18C77 0020 18C44 18C06 0020 18CB0 18C6B 0020 18B5D 0020 18BB0 0020 18C65 18C64 0020 18CBD 18C66>