A Web Interface to Indian Languages

A Thesis Submitted
in Partial Fulfillment of the Requirements
for the Degree of
Master of Technology

by

Prasanna S

to the

Department of Computer Science & Engineering
Indian Institute of Technology, Kanpur

May, 1999

Certificate

This is to certify that the work contained in the thesis entitled "*A Web Interface to Indian Languages*", by *Prasanna S*, has been carried out under my supervision and that this work has not been submitted elsewhere for a degree.

May, 1999

(Dr.T V Prabhakar)
Department of Computer Science & Engineering,
Indian Institute of Technology,
Kanpur.

Abstract

The exponential growth of Internet in India in the recent past has led to the creation of a large amount of content in Indian languages on the Internet. But, when it comes to communicating with each other, many people find it difficult to put forth their thoughts in a foreign language like English. Typically, one would like to send e-mails in his/her native tongue. Moreover, if one were to fill forms in the HTML documents available on the Web, the user has to depend on Latin-1 encoding to submit the data. In this thesis we provide a solution, which would allow users to fill up forms and communicate with each other by sending e-mails in their native language. Though we have provided a solution for Hindi, it can be easily extended to any Brahmi-based script.

Acknowledgements

I would like to express my sincere thanks to my guide Dr. T V Prabhakar for his guidance and invaluable suggestions throughout this work.

I would like to thank my parents and sister for their love and affection. I'm indebted to my cousin Shyam Sundar for his support and guidance throughout my career. I would like to thank all my relatives for their moral as well as material help they provided throughout my life.

I would like to thank all my classmates and friends of Hall-5, particularly Srikar, Subhash, Uma, Varakur, Gopi, Girish, Arvind, Kousik, Hiran, well, the list is endless..., who made my stay in Hall-5 and IITK very memorable.

A special mention has to be made about the *Kannada Sangha*, which provided us some relief from the mess food.

I also thank all my Tamil Mandram friends for their memorable company at Kanpur.

Contents

Chapter 1

Introduction

Human beings all over the world speak a large number of languages. In a country like India, people of different states speak different languages. The Internet has grown enormously in the recent past. Moreover, there is an exponential growth in the number of Internet users in India. This has resulted in large amount of content being created in Indian languages on the Web. But, when it comes to communicating with each other, many people find it difficult to put forth their thoughts in a foreign language such as English. People would prefer to use their native tongue to communicate, for example sending e-mails in Hindi would be preferred to sending in English. Some software allow people to communicate in Indian languages, but they are mainly platform dependent and they require the user to install their package or a similar package. In this thesis we aim to provide a wrapper for Indian languages on the Web, which would help users to communicate in Indian languages without the explicit installation of any package. Our solution is also platform independent as it is written entirely in Java.

1.1 Motivation

The *iLeap* package [1], developed by CDAC, provides a solution to send e-mails in Indian languages, either by embedding the content in the form of GIF images or in HTML format, where the e-mail client should be able to provide a viewer for HTML document, or in ISCII format [3], which requires any ISCII compatible Indian language editor to view the message. None of the existing packages provide support to fill forms in Indian languages, which would be quite handy for the users, who are not so proficient in English.

Typically, one would like to provide a solution in which the software can be embedded in the User-Agent, so that the user need not bother about a viewer. Moreover, when it comes to filling of forms in a Web page, the solution provided by HTML is not sufficient for supporting Indian languages. The users have to type using the Latin-1 encoding scheme, which is not very convenient for typing in Indian languages. A typical scenario is that the designers of the Web page provide a transliteration scheme which would be used in the backend to process the user's request and give an appropriate response. Some of the disadvantages of this solution are:

- It is difficult for users to work with the transliteration scheme provided by the source.
- Different people can use different transliteration schemes.
- The user is never sure of whether the input he has given in the form is the same as he intended to.

Ideally, users should be able to type in their native language, preferably with convenient key mappings and the characters typed should displayed in the native language itself, so that the user need not worry about any ambiguity.

In this thesis, we provide a Java applet, that can be embedded into a HTML document and thus can be used to fill up forms. Further, the information given as input through the applet allows easy processing, as the server-side software can be easily written. As an example application of our applet, we use the applet to send e-mails in Hindi. Though we have provided a solution for Hindi, it can be easily extended to any Brahmi- based script. The solution requires the Java plugin to be installed on the client machine as our applet is written using JDK1.2 and also the font used by the applet, for rendering the glyphs, to be present on the client machine.

1.2 Organization of the Report

In chapter 2, we discuss the related work in the area of multilingual content creation and storage. The design and implementation of our approach is presented in chapter

3. Chapter 4 talks about a small application of our applet, namely communicating in Hindi by the means of e-mails. In chapter 5, we give the conclusions and work that can be done in the future. The Appendix A enumerates the ISCII character set. Appendix B lists a sample HTML file which is used to invoke the applet. In Appendix C, we present some sample configuration files.

Chapter 2

Related Work

Due to the rapid development of the Internet, and the variety of users it caters to, there is a great need for the Internet to support various languages. This change has not been so smooth, because of the poor standards that Internet initially started with. As a result of this, there has been a variety of new standards that have come up to support Multilingualism on the Web. ISCII, Indian Script Code for Information Interchange, is one such standard for Indian languages.

2.1 Multilinguism on the Web

The Internet which started as a small network in the USA, has become a global network and the number of people accessing the Internet has been on the rise. This has shown a number of limitations on the standards and protocols on which the Internet is based. HyperText Markup Language, HTML, is a markup language used to create documents on the Web. The biggest disadvantage of HTML is its dependence on Latin-1 (ISO-8859-1) character encoding, which is the character set of HTML. Latin-1 is an 8-bit character encoding scheme and can have a maximum of 256 characters. It can support only the Western European languages. Thus, having a large character set is a basic requirement of Multilingalism. The newer versions of HTML, uses Unicode as its document character set. But most of the User-Agents, i.e., browsers, cannot understand Unicode and documents have to provide appropriate fonts to render the document in the correct font. For example, if a document is written in three languages, say Hindi, Tamil and French, the fonts for these three languages should be present on the client machine to render the document in its original form.

2.2 Character Encodings

A document on the Internet can be written in various languages. The character encoding of a document represents a subset of the characters in the document. There are several character encodings that contain characters from a few scripts for example, ASCII (supports the characters in English), ISO-8859-1 (encodes most Western European languages), ISO-8859-5 (supports Cyrillic), SHIFT_JIS (a Japanese encoding). The Unicode character encoding is a 16-bit character encoding scheme, which can encode more than 65,000 characters. UTF-8 (Universal Character Set Transformation Format) encoding scheme is useful if English is used with other languages in the same document. When a document is transferred on the Web, the User-Agent on the other side should be informed what the character encoding of this document is.

2.2.1 ASCII

The English language has, until recently, dominated the Web. E-mail, gopher, ftp use the 7-bit ASCII (American Standard Code for Information Interchange) standard, allowing for 128 characters. ASCII codes 0 to 31 and 127 are control characters that are non printable, whereas the other codes represent printable characters. The codes 32 to 126 represent characters of the Roman alphabet mapped to the standard qwerty keyboard. ASCII characters are each represented by one byte according to a standard code. Use of ASCII for information interchange has been severely limiting for languages written in non-roman scripts. Thus authors have been forced to use various inadequate transliteration schemes to handle non-roman characters when digitizing text.

2.2.2 ISCII

The Department of Electronics came up with a standard for Indian code chart. The Indian Script Code for Information Interchange, ISCII[3], specifies a 7-bit code table which can be used in 7 or 8-bit ISO compatible environment. It allows English and

Indian script alphabets to be used simultaneously. ISCII retains the ASCII character set in the lower half and provides the Indian script character set in the upper 96 characters in the 8-bit code. Some of the main features of ISCII are as follows:

- ISCII character set is a super-set of all the characters required in the 10 Brahmi-based languages.
- The ISCII code contains only the basic alphabet required by Indian scripts. The composite characters are formed by combining these basic characters.
- ISCII code has the advantage that there is only one way of typing a word. Any word can only be typed in the phonetic order of the constituent basic characters.
- All words have unique spellings and are not affected by the display rendition.

Eight-bit ISCII code

ISCII-8 is an 8-bit encoding standardized by DOE in 1986. The lower 128 characters contains the ASCII character set, while the upper half of the table is used for Indian script code. The first two columns in the upper half are reserved for control characters as per the recommendation of the ISO(International Standards Organization). This encoding allows free mixing of Roman characters with Indian scripts.

Seven-bit ISCII code

Seven bit ISCII code is recommended for those computers and packages which do not allow the use of 8-bit codes. In 7-bit coding schemes, 128 positions are available for representing all characters of the script.

In this encoding scheme, control codes of ASCII are retained and all other codes are used to represent Indian scripts. The disadvantage of this code is that Roman scripts cannot be mixed with Indian scripts.

2.2.3 Unicode

Unicode[5] is a 16-bit character encoding that allows 65,535 characters compared to the 256 characters provided by 8-bit ASCII. The Unicode Character Standard is designed to support the interchange, processing and display of texts in various languages of the modern world. Unicode is the first plane of ISO-10646, which is also called BMP (Basic Multilingual Plane) or Plane Zero. ISO-10646 is a 32-bit encoding. It is divided into 32,000 planes, each of 64,000 characters capacity. This permits 2,080 million characters. This form is also called UCS-4, Universal Character Set 4-bytes. Only the first plane of UCS-4 encoding, Unicode, is in use.

The Devanagari block of the Unicode standard is based on ISCII-1988. The Devanagari characters are encoded in the range U+0900–U+097F.

2.2.4 UTF-8

Character encoding standards define not only the identity of each character and its numeric value, or code position, but also how this value is represented in bits. The Unicode standard endorses two forms that correspond to ISO 10646 *transformation formats*, UTF-8 (Universal Character Set Transformation Format)[4] and UTF-16.

UTF-8 is a way of transforming all Unicode characters into a variable length encoding of bytes. It has the advantages that the Unicode characters corresponding to the familiar ASCII set end up having the same byte values as ASCII, and that Unicode characters transformed into UTF-8 can be used with much existing software without extensive software rewrites. Any Unicode character expressed in the 16-bit form can be converted to the UTF-8 form and back without loss of information. Some salient features of the UTF-8 encoding are as follows:

- All UCS characters > 0x7f are encoded as a multibyte sequence consisting only of bytes in the range 0x80 to 0xfd.
- The lexicographic sorting order of UCS-4 strings is preserved.
- All possible 2^{31} UCS codes can be encoded using UTF-8.
- The bytes 0xfe and 0xff are never used in the UTF-8 encoding.
- UTF-8 encoded UCS characters can be upto six bytes long.
- The first byte of a multibyte sequence which represents a single non-ASCII UCS character is always in the range 0xc0 to 0xfd and indicates how long this multibyte sequence is.

Encoding

The following byte sequences are used to represent a character encoded in UCS. The sequence to be used depends on the UCS code number of the character:

```
0x00000000 -- 0x0000007F:
0xxxxxxx

0x00000080 -- 0x000007FF:
110xxxxx 10xxxxxx

0x00000800 -- 0x0000FFFF:
1110xxxx 10xxxxxx 10xxxxxx

0x00010000 -- 0x001FFFFF:
11110xxx 10xxxxxx 10xxxxxx 10xxxxxx

0x00200000 -- 0x03FFFFFF:
111110xx 10xxxxxx 10xxxxxx 10xxxxxx 10xxxxxx

0x04000000 -- 0x07FFFFFF:
1111110x 10xxxxxx 10xxxxxx 10xxxxxx 10xxxxxx 10xxxxxx
```

The xxx bit positions are filled with bits of the character code number in binary representation. Only the shortest possible multibyte sequence which can represent the code number of the character can be used.

2.3 Nature of Indian Alphabet

All the Indian scripts have originated from the ancient Brahmi script which is phonetic in nature. Thus all the Indian languages are phonetic in nature, but there are subtle differences in the alphabet. The differences in Indian languages are mainly in their written form. Different combination rules get used in generating the appropriate sounds.

- **The Consonants**
  Indian script consonants[3] have an implicit अ vowel included in them. These consonants have been categorized according to their phonetic properties. There are 5 Vargs (Groups) and non-Varg consonants. Each Varg in turn consists of 5 consonants, out of which the last one is a nasal consonant.

Varg Consonants				
ka क	kha ख	ga ग	gha घ	"na ङ
ca च	cha छ	ja ज	jha झ	~na ञ
.ta ट	.tha ठ	.da ड	.dha ढ	.na ण
ta त	tha थ	da द	dha ध	na न
pa प	pha फ	ba ब	bha भ	ma म

Table 2.1: Varg Consonants in Devanagari

Non-Varg Consonants							
य	र	ल	व	श	ष	स	ह
ya	ra	la	va	"sa	.sa	sa	ha

Table 2.2: Non-Varg Consonants in Devanagari

- **Anuswar (ं)**
  Anuswar[3] indicates a nasal consonant sound. When it appears before a consonant belonging to any of the 5 Vargs, it represents the nasal consonant belonging to that Varg. However, if it appears before a non-Varg consonant, it represents a different nasal sound.

- **Nasalization Sign: Chandrabindu (ँ)**
  The chandrabindu (ँ) [3] represents nasalization of the preceding vowel. In Devanagari script, it often gets substituted with Anuswar, as the latter is more convenient for writing. However, in some words, the Anuswar and Chandrabindu can give different meanings as in the case of हँस, which means to laugh and हंस, which means a swan.

- **Vowels and Vowel Signs (Matras)**
  All the Vowels [3] of Indian languages are pronounced independently and they have different symbols. The consonants of Indian scripts have an implicit vowel sound. The vowel-sign (Matra) is attached to the consonant to indicate a vowel sound other than the implicit one. Thus every vowel, excepting the अ vowel, has an equivalent Matra.

Vowels and Vowel Signs											
Vowels	अ	आ	इ	ई	उ	ऊ	ऋ	ए	ऐ	ओ	औ
Vowel-sign	–	ा	ि	ी	ु	ू	ृ	े	ै	ो	ौ
Roman	a	A	i	I	u	U	.r	e	ai	o	tt au

Table 2.3: Vowels and Vowel signs in Devanagari

- **Vowel Omission Sign: Halant(्)** All the consonants in Indian scripts are assumed to have an implicit vowel sound, अ, within them unless a Matra is attached to the consonant, in which case it has a different vowel sound. To indicate that the consonant does not have the implicit vowel अ in it, the special sign Halant(्) is used.

- **Conjuncts**

  There are numerous conjuncts[3], which essentially are clusters of upto four consonants without the intervening implicit vowels, in Indian scripts. The shape of these conjuncts can differ from those of the constituent characters. These conjuncts are formed in the ISCII code by putting the Halant(्) character, between the constituent consonants. Examples:

  क ् ष = क्ष
  प ् र = प्र

- **Diacritic Mark: Nukta**

  The Nukta is used in deriving 5 other consonants in the Devanangari and Punjabi scripts, required for Urdu. It is also used for ड़ and ढ़ in some Northern scripts.

- **Punctuation**

  The punctuation marks used in Indian scripts are borrowed from English. Instead of the full-stop a Viram(।) is used in Northern scripts.

- **Numerals**

  The Devanagari script has its own set of numerals corresponding to the international numerals.

Numerals									
०	१	२	३	४	५	६	७	८	९
0	1	2	3	4	5	6	7	8	9

Table 2.4: Numerals in Devanagari

2.4 Font Encodings

The encoding of a font specifies the mapping from character codes (an integer, typically between 0 and 255) to the characters. There are no particular standards for encoding fonts and almost every vendor comes up with an encoding scheme that is convenient for use in their software.

Though character encoding is different from font encoding, the font encoding could be the same as the character encoding and this would greatly reduce the standardization problem of font encodings.

In the case of Indian language fonts, this becomes quite complicated as ISCII has to be different from the font encoding, because of the presence of conjuncts in Indian languages. Most fonts for Devanagari generally provide a glyph for the danda, ा, and glyphs for half consonants like च्, which have to be combined together to form the ligature च. Further, in words like अच्छा, the half consonant च् is used, which makes it essential for the font designers to provide half consonants.

2.5 Inscript Keyboard Layout

The Inscript overlay[2, 3] is the same for all the Indian scripts and contains the basic characters required for all of them. The Figure 2.1 is for the Devanagari script. The characters which are not used in Devanagari are not shown in the overlay.

Figure 2.1: Inscript Keyboard Layout

The left half of the keyboard contains the Vowels, Nasal signs and Nasal consonants. A matra is given in the Unshift position and the corresponding Vowel is given in the Shift position. The Vowel अ has no corresponding Matra as its sound is inherently present in each consonant. A Halant ् sign is used to remove the inherent अ sound from a consonant. Halant sign is also used quite often to combine consonants, and is thus given in the Unshift position.

The consonant pad is in the right half of the keyboard. The 5 consonant Vargs in Devanagari have 5 consonants each. The Vargs are shown in the Table 2.1. The two pairs of each Varg are kept in two vertical keys. The consonant in the Shift position of a key is the "Aspirated" counterpart of the consonant in the Unshift position. An Aspirated consonant is one that has an extra "h" sound in it. For example, ख is an aspirated counterpart of क.

The Non-Varg consonants in Devanagari are also accomodated in the consonant pad on the right half of the overlay.

Viram is provided on the Shift position of the full-stop. For convenience of typing, some Conjunct characters are provided in the Shift positions of the digits. The action of these keys is the same as typing two or three characters comprising each of the corresponding conjuncts.

2.6 iLeap

iLeap is a multilingual Indian language word processor, developed by CDAC, available for the Windows operating system. The word processor allows users to type in any Indian language. The following are the salient features of *iLeap*:

- It uses the Inscript keyboard layout.
- If one learns typing in any Indian language, it is easy to type in any other language, as all Indian languages have similar keyboard layouts in the Inscript scheme.
- It provides variety of fonts for each of the Indian languages.
- We can use different languages in the same document.

- It allows to export text to HTML, RTF, BMP, JPEG which makes it easy to use for publishing, creating Web pages.
- Support for complete WYSIWYG display and printing.
- Design templates for routine documents, re-use existing documents as templates.

One can send e-mails in Indian languages using *iLeap*. It is capable of converting the Indian language document into a GIF imaga, a JPEG image, or as a HTML document, which can be attached with the mail. Moreover, it can also send e-mail using the ISCII character encoding, which requires an ISCII interpreter to be present on the recipient's machine.

Chapter 3

Design and Implementation

The main problem of communicating in various languages across the Internet are the insufficient standards and protocols. The current standards allow transfer of content only in UTF-8 encoding, which is safe for transmission across various networks. Moreover, most of the computers have the “qwerty” keyboard, using which typing in different languages is not possible unless there is a keyboard driver that allows the user to do this. Users have to depend on Latin-1 to fill forms. This restricts Multilingual features of the Web in a big way. Our design provides a way to get around this problem of feedback in Latin-1 by providing a Java applet that allows users to type in Indian languages. The applet can then communicate with the server and transfer the information provided by the user.

3.1 Our Approach

In our approach, we provide a Java applet, that acts as a keyboard driver and allows users to type in Indian languages. The language in which the user can type and the font dependent information can be provided to the applet through the various configuration files.

3.1.1 Why Java Applet?

We have chosen to provide a Java applet as it is platform- and operating system-independent and it is downloaded across the network and interpreted by the browser. Further, applets written in Java do not pose any security threats to the client system. Java applets can be easily embedded in HTML documents. They can be used to create smart forms, that are interactive, and can do input verification at the client side, rather than verifying on the server, which can save bandwidth.

3.1.2 Configuration Files

The Java applet requires the following configuration files while startup. The place where the applet can find these files on the server is specified in the `<PARAM>` tag in the HTML file that embeds the Java applet. The following have to be specified in the HTML file:

- `ISCIIFILE`, the file containing the character encoding information
- `KEYBOARDFILE`, the file containing information about the layout of various keys.
- `FONTTABLEFILE`, the file that contains information on the conjuncts in the language.
- `FONTFILE`, the file that contains information of the glyph positioning in the corresponding font that is used.
- `UNICODEFILE`, is the file that contains the mapping between Unicode and ISCII. The internal representation of characters are in ISCII form for processing, but in Unicode format when stored or when sent across the network.

Character Encoding Format

This file is provided so that the other configuration files can be easily written and interpreted. This file has the format:

```
STRING_REPRESENTATION_OF_CHARACTER   CHARACTER_CODE
```

The character code is an integer. It can be an 8-bit character code or a 16-bit character code. As this string representation of characters is used in all the other files, this helps in using any character encoding the user prefers. The file name should be set as the value for `ISCIIFILE` parameter. In this file the string representation of a character as well as the character code are one word each. The string representation is not standard, but it is used internally by the applet in reading other configuration files.

Keyboard Map

The layout of various keys on the keyboard is specified in the "keyboard_map" file. This file name is specified in the HTML file, which embeds the Java applet as a parameter using the `<PARAM>` tag. The default keyboard layout used by the applet is the Inscript keyboard layout. By changing the file, the user can give a keyboard layout of his choice. This file should be written in the following format:

```
ASCII_value_of_the_key  A_set_of_ISCII_codes_in_string_format
```

The set of ISCII codes represent the codes corresponding to the character in that layout. If a key corresponds to more than one ISCII code, then the codes are written in the same line separated by whitespace. This file should be specified as the value of the parameter `KEYBOARDFILE`. In this file, ASCII value of a key is a single word, but the ISCII codes can be any number of codes mapped to that key.

Conjuncts

Conjuncts are clusters of upto four consonants without the intervening implicit vowels in Indian scripts. The shape of these conjuncts can differ from those of the constituent characters. Different Indian scripts have different conjuncts and hence have to be defined in the configuration file, so as to make the applet language independent.

The format of the conjunct file is as follows:

```
STREAM_OF_ISCII_CHARACTERS  INTERMEDIATE_GLYPH_CODE
```

The ISCII characters are separated by whitespace. The intermediate glyph code is an integer which is used to represent each possible character that can be generated from the font. This file can be specified as the parameter `FONTTABLEFILE`. The intermediate glyph code is a unique number and is the last word of each line. The rest of the words correspond to the ISCII characters that are represented by the glyph code. The glyph code is not a standard code, but used internally by the applet.

Font Information

The font information file, specified in the parameter `FONTFILE`, gives the mapping between the intermediate glyph code and the glyph code used in the font. This file is dependent on the font that is used as every font has its own encoding scheme. This file also gives us the information about how many glyphs in the font correspond to a particular character in the alphabet.

This file should be in the format:

```
INTERMEDIATE_GLYPH_CODE  GLYPH_CODES_CORRESPONDING_TO_THE_CHARACTER
```

As in the other configuration files, the glyph codes corresponding to the character are separated by whitespace. All possible glyph sequences have unique intermediate glyph number. These intermediate glyph numbers can map any number of glyphs in the font.

3.2 Devanagari Text Area

JDK1.2 provides `JTextComponent`[6] class for text editing features. The `JTextComponent` class is a super class of a set of classes. The most commonly used components for editing purposes are the `JTextField`[6] and `JTextArea`[6]. The `JTextComponent` provides these customizable features for all its descendents:

- A separate model, known as the *document*, to manage the component's content.
- A separate view, which is in charge of displaying the component on the screen.
- A separate controller, known as an *editor kit*, that can read and write text and that implements editing capabilities with actions.

- Customizable keymaps and key bindings.
- Support for infinite undo and redo.
- Pluggable caret and support for caret change listeners.

3.2.1 Documents

Like all Swing components, a text component separates its data (known as model) from its view of the data. A text component's model is known as a *document*. A document [7] provides these services for a text component:

- A document stores the textual content, which can represent any logical text structure, such as paragraphs, text runs that share styles, and so on.
- Provides support for editing the text through the `remove` and `insertString` methods.
- Notifies document listeners and undoable edit listeners of changes to the text.

The `DevTextArea`, designed by us, is a subclass of the `JTextArea`, which is inherited from the `JTextComponent` class. The `DevTextArea` has its own document, which is used to display the variable number of glyphs associated with each character in Devanagari. For example, the character च constitutes the glyphs च् and ा.

3.3 Keyboard to ISCII encoding

The `DevTextArea` uses the inscript keyboard layout for the input of characters. The code of the typed key is converted into a corresponding ISCII code. The following subsections describe the process in detail.

3.3.1 Keyboard Map

The "keyboard_map" specifies the mapping between the "qwerty" keyboard and the inscript keyboard. The "keyboard_map" file consists of the key codes and the corresponding ISCII characters mapped to the key. For example, the the key '&' is mapped to क्ष, which comprises of the ISCII codes for क, ्and ष. Whenever a key is pressed the corresponding ISCII codes for the key are generated and this serves as the input to the routine which generates the display codes. Instead of using the '&' key to get क्ष, it can be generated by typing the keys 'k', 'd' and 'M', which represent the keys for 'क', '्', and 'ष' respectively. As a key pressed generates a set of ISCII codes which is independent of the font code, it is possible to generate the same character to be displayed for all the key sequences required to generate a particular character.

We use the inscript keyboard layout, in which the vowels and the consonants of the Indian language alphabet are arranged on the left and right side of the keyboard respectively.

3.3.2 ISCII

The "iscii" file contains the mapping of symbols to the 8-bit ISCII codes. The appendix lists the table of ISCII codes. This mapping decreases the complexity of other files as other configuration files are written using these symbols and makes the configuration files readable.

3.4 Formation of Devanagari Characters

Characters in Devanagari are represented as a series of ISCII codes. To obtain a character in Devanagari, the constituent ISCII characters are combined based on the information in the "font table" file, which specifies the set of ISCII characters that correspond to the display code.

3.4.1 Generation of Characters

We construct a hash table, in which a set of ISCII characters are mapped to a display code. If the input pattern consists of an ISCII character sequence which is a key in the hashtable, the corresponding value is stored in the display buffer. The display buffer thus consists of a set of display codes corresponding to a series of glyphs. For example, when the characters प, ्and र are typed, the ISCII codes for these characters combine to form the code for conjunct character प्र as specified in the "font table" file.

3.4.2 Generating Glyphs from Intermediate Display Sequence

All possible characters in Devanagari are given a unique numbering used for internal representation. This numbering is different from the font code as the font contains separate glyphs च् and ा which are combined to form the character च. The display code numbering corresponds to logical characters in Devanagari like च.

Once the display code is obtained from the ISCII code, the characters to be displayed are generated using the "`FONTFILE`" file, which contains the mapping between the display code and the font code. The `insertString` method of `DevTextArea`'s document model, does the conversion between display code to font code. Thus characters which comprise of one or more glyphs are generated and these glyphs are displayed in the `DevTextArea`.

3.5 Position of Matras

Matras are vowel signs that are added to consonants to indicate a vowel sound other than the implicit one. All matras in Devanagari, except the Matra ि are positioned after the consonant. The Matra ि is positioned before the consonant. Because of the phonetic nature of ISCII encoding, the code of Matras appear after the code of the consonants.

For example, when the word प्रिय is written, the sequence of ISCII codes generated would be the ISCII codes corresponding to the characters प ् र ि and य. As the conjunct character प्र is present in our set of conjuncts, the characters प ् र would be converted to the character प्र. As the Matra ि is the next character in the ISCII stream, it has to be positioned before the character प्र and the display sequence would consist of the code for the characters ि and प्र. The next character य can be added to the display sequence without any modifications to the display sequence.

3.6 Position of Reph

The Reph (र्), is generated by typing र and ्. While displaying र् has to be placed after the next syllable. For example, in the word बर्फी, the ISCII encoding would correspond to the characters ब र ् फ and ी. The resulting display sequence would be बर्फी. Notice that the (र्) has to be positioned beyond ी to obtain the appropriate word.

3.7 Caret Positioning

Whenever a character is added in the text area, the caret position has to be changed to show the user a logical position where the next character would be added. As a display code might represent more than one glyph, the caret has to be moved as many places as the number of glyphs inserted, as otherwise the caret may be positioned between the glyphs constituting to the character.

The "font.info" file contains a mapping between display code and font code. Thus from this file we can find out the number of glyphs corresponding to a particular display code. With this information the caret can be positioned at the appropriate place.

Caret Movement

In English, there is a one to one correspondence between the input character and the symbol displayed. So movement of the caret corresponds to positioning the caret according to the number of glyphs present. However, in Indian scripts many input characters may combine to form one display symbol. For example, क्ष is a combination of क, ् and ष. When this kind of input occurs, the caret is moved to the end of previous syllable when the user presses the left arrow key. Movement of the caret is done using the `setCaretPosition` of the `JComponent` class. The position of the caret is calculated and the `setCaretPosition` method is called with the appropriate caret position as its parameter. As a different number of ISCII characters may combine to produce a particular glyph, a separate buffer called the `isciiBuffer` maintains the ISCII string input through the key board. When the user moves in the text, the corresponding movement in the `isciiBuffer` also has to be obtained. This would result in the proper insertion of the characters as the user might want. The change in the caret position leads to a corresponding change in all the buffer positions.

Text Manipulation

The `DevTextArea` serves as a Hindi text editor. Characters can be inserted at the current caret position. The caret positions are demarcated by syllables. One can insert text before or after syllables by not in between the characters in the syllable. For example, in the word बर्फी, which comprises of the characters ब, र, ्, फ and ी new insertions can be performed only before or after the syllables ब and र्फी. Characters cannot be inserted between the constituent characters of a syllable.

To insert a character between the constituent characters of a syllable, one has to break the syllable by pressing the `Backspace` key by positioning the caret after the syllable and rewriting the word from that position.

For example, if one has to change the word पिया to प्रिया, the caret should be moved between पि and या, the sequence of keys to be pressed by the user can be summarized as shown in Table 3.1.

To delete a character, the user has to position the caret after the particular character and press the `Backspace` key. This results in deletion of component characters if the character is a conjunct character. The Table 3.2 shows the sequence of characters displayed when `Backspace` is pressed after the conjunct character क्ष, which comprises of the characters क, ् and ष.

Keys to be pressed	Codes Displayed
Backspace	पया
्	प्या
Backspace	प्य
Backspace	प्
र	प्र
ि	प्रि
य	प्रिय
ा	प्रिया

Table 3.1: Sequence of Keys to be typed for changing पिया to प्रिया

Codes Displayed
क्ष
क्
क

Table 3.2: Deletion of a Conjunct Character

Since we modify the underlying *document* of the `TextArea` after every key pressed, other features of any `JComponent` like scrolling can be easily supported by enclosing the `DevTextArea` in a `JScrollPane`[6].

Chapter 4

An Application: E-mail in Hindi

In this chapter we describe an application of the `DevTextArea` in the form an applet that is capable of sending e-mails in Hindi. Initially we look at some of the problems in sending e-mails in Hindi and how these problems are overcome. We also look at the some of the issues in sending and receiving mails.

4.1 Problems with Hindi E-mails

The problems in sending e-mails in Hindi can be summarized as follows:

- Different fonts have different encoding schemes.
- Most of the existing UNIX systems use 7-bit encoding schemes to transfer information through e-mail.
- Most systems come with the "qwerty" keyboard, using which we cannot type mails in Hindi.
- Even if the font is present on the system, there is usually no regular mapping between keyboard and font code. So getting the appropriate characters would be difficult.
- If we use the font code for encoding characters, it would not be possible to encode all the characters using the 7-bit code.

4.2 Sending Hindi E-mails

In this section we discuss about how the Hindi mail is sent using the existing resources.

4.2.1 Typing the Mail in Hindi

We use the `DevTextArea`, which serves the purpose of an editor, to type the mails in Hindi. As we have discussed earlier, this editor uses the Inscript keyboard layout, in which the consonants are on the right side of the keyboard and the vowels on the left side. Hence it is very conveinient to type the mail in Hindi using this editor.

The `DevTextArea` represents the mail typed as a stream of ISCII characters. These ISCII characters are converted to Unicode characters. The Unicode characters are sent over the network by encoding the characters in UTF-8.

4.2.2 Encoding the Mail in UTF-8

Characters in Unicode are of 2-bytes length. Hence, they cannot be sent over the network directly. So they are encoded into the UTF-8 encoding, in which the Unicode characters are represented as a set of two separate bytes. The `StringEncoder` class has a method, `unicodeToUTF8`, which converts the given Unicode string into a set of UTF-8 bytes. These bytes are ready to be sent over the network.

4.2.3 SMTP

Our e-mail client uses the Simple Mail Transfer Protocol, SMTP[8], to send mails. A client delivers a mail message by opening socket connection to port 25 on the server. Once the connection is established the client sends a series of commands, delivers the message and disconnects.

All commands and responses to and from the SMTP server are terminated by CRLF. For every command sent by the client, the server responds with a text response, that starts with a 3-digit code. The server response codes are categorized in Table 4.1

Response Code	Explanation
200-299	Successful command execution
300-399	Command was initially successful but it needs more information to complete
400-499	There was an error on the server (server down, file system full)
500-599	Indicate an error on the client side (Invalid password, unknown command)

Table 4.1: Response codes in SMTP

The sequence of commands sent from the client to the server is:

- `HELO host_name` - a simple greeting from the client to the server. The host name field is optional, but the sendmail programs in our lab require the host name field.
- `MAIL FROM: sender's address` - identifies the user account which is sending this mail. No check is performed by the server on to who is sending this message. So it is virtually possible to send a mail as anyone in the world
- `RCPT TO: receiver's address` - identifies the address where this mail will be sent. Some of the mail servers are configured so that this address need not be necessarily on the server itself.
- `DATA` - tells the server that the client is ready to send the message body. It waits for a server response code, starting with digit 3, which means that the server is ready for accepting the message body. The end of the message is signified by a single "." on a newline. If a line in the message body starts with a ".", the client adds another "." before the start of that line. The "." also signifies the end of the session and hence closes the connection.

The `SMTPSession` class implements a session with the SMTP server. It is used by our applet to send the mail. The `sendMessage` method essentially opens a socket on the SMTP port and writes the headers and other information essential for initiating the SMTP session. Then the body of the message is written on the socket. Any error occurred during the sending of mail is reported by the `SMTPSession` class.

4.3 Receiving Hindi E-mails

Our e-mail client uses POP3 to fetch the mail box from the server. So the applet should be fetched from a webserver, that also acts as the POP3 server for the local network. The following sections describe in detail how the mail is being fetched and displayed to the user.

4.3.1 POP3

Post Office Protocol version 3, POP3[9], is a popular protocol used to remotely access mails on the Internet. By using the POP3 protocol, the user can read his/her mails without having to copy his/her entire mail box on to the local machine. POP3 responses start with a "+" for a successful command and a "-" in the case of an error. POP3 responses could span only a single line or many lines. If the response is a multiple line response, it is terminated with a line containing a single ".".

After connecting to a POP3 server, which usually listens on port 110, you have to send a `USER user_name` command, followed by a `PASS` command giving the user's password.

Once we have logged on to the POP3 server, we can do the following:

- Retrieve a count of the number of messages using the `STAT` command.
- Get a list of the active message numbers with the `LIST` command.
- Examine the beginning of a message using the `TOP` command. This is an optional command according to the standard, but most servers implement it.
- Read an entire message using the `RETR` command.
- Delete a message using the `DELE` command.

The `POP3Session` class acts as a wrapper to the application, whereby it takes care of the connection establishment, retrieving a mail, converting the raw sequence of characters into a `Message`, the data structure that represents a message.

4.3.2 Fetching the Mail

The `POP3Session` class implements a session with a POP3 server. Our e-mail client uses this interface to retrieve mails stored on the server. First the number of messages in the mail box of the user is found using the `STAT` command. The `TOP` command is used to fetch the headers of all messages, from which the sender's name, subject, date and time when the message is sent can be found out.

When a user selects a particular mail message, it is retrieved from the server using the `RETR` command.

4.3.3 Decoding the Mail

The mail fetched using the `RETR` command contains the headers as well as the body of the message. These headers are stripped from the message using **find the method name** in the `Message` class. The `body` member of the `Message` class contains the body of the message after the headers are stripped off. The body of the message is what we are interested in as it consists UTF-8 encoded Unicode characters, which represents the Hindi string.

We convert the UTF-8 encoded characters into Unicode using the `UTF8ToUnicode` method of the `StringConverter` class. This sequence of Unicode characters is converted into ISCII using `UnicodeToIscii` method of the `StringConverter` class.

4.3.4 Displaying the Mail

The method `setISCIIString` of `DevTextArea` converts the given sequence of ISCII codes into a sequence of display codes, by using the longest matching sequence of conjuncts as specified in the "font" file. The display code corresponds to a set of font codes. When a particular display code is inserted into the text area, the underlying document of the `DevTextArea` replaces this display code with a sequence of font codes. Thus the text displayed in the e-mail consists of a sequence of Hindi characters.

Due to applet security restrictions, which state that an applet can make network connections only to the host from which it is downloaded, it is required that the server from which the applet is downloaded has to be the SMTP server as well as the POP3 server.

4.3.5 Mails in other Indian Languages

The application can be extended to display any other Indian language as all the Indian languages have a similar structure. To send mails in other Indian languages, one has to ensure the following things:

- The appropriate Indian language font has to be installed on the client machine.
- The keyboard layout of the particular language has to be specified in the "keyboard_map" file.
- Though the ISCII codes for all Indian languages are the same, the characters have different Unicode values. The configuration file which contains the mapping table between ISCII and Unicode has to be changed appropriately.
- The conjuncts in the appropriate Indian language have to mentioned in the configuration file which has information about conjuncts.

Chapter 5

Conclusions and Future Work

In this thesis, we have presented a Java applet that can be embedded into a HTML page, which is used to get user input in Indian languages. The solution is secure, portable and transparent to the end user. The solution was implemented for Devanagari script, which forms the basis for documents in Hindi, Marathi and Sanskrit. This solution can be extended to other languages just by providing the configuration files required at startup. We have also implemented an e-mail client that allows users to transparently send and receive e-mails in Hindi.

5.1 Limitations

Our solution has the following limitations:

- The solution requires the Java plugin 1.2 to be installed on the client machine.
- At present, only one language can be used in the applet.
- Moreover, the applet does not allow dynamic change of fonts in the input.
- It requires the font to be installed on the client machine.
- The startup time of the applet is quite high. This is mainly because of the high startup time of Java applets.

- The e-mail applet requires the Web server to be support SMTP as well as POP3 protocols, as it cannot make network connections to other hosts in the network.
- The e-mail applet also requires addresses to be typed using Latin-1 encoding.
- The current text area does not provide mouse support.

5.2 Future Work

- Our applet supports only one language at a time. This can be extended to support multiple languages.
- The applet can be extended to support more than one font in the same input area.
- It can be extended to support styled text as input.
- The e-mail applet can be used to exploit the security model provided by Java and connect to any server on the network. This can be done by changing the default security model provided by Java.
- An interactive addressbook can be provided, which would allow users to give email addresses in Hindi.
- The e-mail applet can be slightly modified to allow creating and maintaining folders for incoming mails.
- Threads in Java can be used to provide improvement in performance.
- The applet can be enhanced to allow users to enter URLs in Hindi. This can be done by providing a small dictionary interface between the ISCII code for the URL and the absolute URL in Latin-1.
- Mouse support can be provided and block actions like cut, copy, paste can be supported.

Bibliography

[1] *iLeap - Intelligent Internet Ready Indian Language Word Processor* `http://www.cdac.org/gist/ileap1.html`

[2] *Gist Multi-lingual Card user's guide*, Quark Computers Pvt. Ltd., C-1, Sarvodaya Nagar, Kanpur, 1996

[3] *Indian Script Code for Information Interchange - ISCII*, IS 13194 : 1991, Bureau of Indian Standards, Manak Bhavan, 9 Bahadur Shah Zafar Marg, New Delhi 110002.

[4] F. Yergeau, *UTF-8, a transformation format of Unicode and ISO 10646*, Request for Comments: 2044, October 1996

[5] *The Unicode Standard, Version 2.0*, The Unicode Consortium, Addison-Wesley Developers Press, Chapter 6 pp33-34, Chapter 7 pp72-74

[6] *The Java 2 Platform API Specification* `http://java.sun.com/products/jdk/1.2/docs/api/index.html`

[7] *The Java Tutorial* `http://java.sun.com/docs/books/tutorial/index.html`

[8] Jonathan B. Postel, *Simple Mail Transfer Protocol*, Request for Comments: 821, August 1982

[9] J. Myers, M. Rose, *Post Office Protocol - Version 3*, Request for Comments: 1725, November 1994

[10] Mark Wutka, et. al. *Hacking Java: The Java Professional's Resource Kit*, Macmillan Publications, 1997

Appendix A

A.1 8-bit ISCII Code

A.2 7-bit ISCII code

Appendix B

B.1 HTML file in which the applet is embedded

In the following file we specify the parameters to the applet using the `PARAM` tag. If these parameters are not specified in this file, the applet assumes default values for these files. The initial part of the file is a small Java script, that enables us to generate the appropriate HTML code for a given browser. The script also specifies where the user can get the Java plugin for running JDK 1.2 applets.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2//EN">
<!--NewPage-->
<HTML>
<HEAD>
<TITLE>A Simple Applet</TITLE>
</HEAD>

<BODY>
<SCRIPT LANGUAGE="JavaScript"><!--
var _info = navigator.userAgent; var _ns = false;
var _ie = (_info.indexOf("MSIE") > 0 && _info.indexOf("Win") > 0 &&
_info.indexOf("Windows 3.1") < 0);
//--></SCRIPT>
<COMMENT><SCRIPT LANGUAGE="JavaScript1.1"><!--
var _ns = (navigator.appName.indexOf("Netscape") >= 0 &&
```

```
((_info.indexOf("Win") > 0 && _info.indexOf("Win16") < 0 &&
java.lang.System.getProperty("os.version").indexOf("3.5") < 0) ||
(_info.indexOf("Sun") > 0) || (_info.indexOf("Linux") > 0)));
//--></SCRIPT>

<SCRIPT LANGUAGE="JavaScript"><!--
if (_ie == true) document.writeln
('<OBJECT classid = "clsid:8AD9C840-044E-11D1-B3E9-00805F499D93"
WIDTH = "500" HEIGHT = "500" ALT = "Your browser understands the
applet tag but isn\'t displaying any applet."
CODEBASE=
http://java.sun.com/products/plugin/1.2/jinstall-12-win32.cab#Version=1,2,0">
<NOEMBED><XMP>');
else if (_ns == true) document.writeln
('<EMBED type="application/x-java-applet;version=1.2"
code = "HindiMail.class" java_CODE = "HindiMail.class"
WIDTH="500" HEIGHT="500"
pluginspage="http://java.sun.com/products/plugin/1.2/plugin-install.html">
<NOEMBED><XMP>');
//--></SCRIPT>
<APPLET CODE = "HindiMail.class" WIDTH = "500" HEIGHT = "500"
ALT = "Your browser understands the applet tag but isn't displaying
any applet."></XMP>

<PARAM NAME = "code" VALUE = "HindiMail" >

<PARAM NAME = "type" VALUE = "application/x-java-applet;version=1.2" >
<PARAM NAME = "ISCIIFILE" VALUE = "resources/iscii">
<PARAM NAME = "KEYBOARDFILE" VALUE = "resources/keyboard_map">
<PARAM NAME = "FONTTABLEFILE" VALUE = "resources/font">
<PARAM NAME = "FONTFILE" VALUE = "resources/font.info">
<PARAM NAME = "UNICODEFILE" VALUE = "resources/unicode">
```

```
</APPLET>
</NOEMBED></EMBED></OBJECT>
</BODY>
</HTML>
```

Appendix C

C.1 Sample ISCII Code File

The configuration file `ISCIIFILE` has information as shown in the following table[1]. The glyphs in the last column are not part of the configuration file.

String Representation	ISCII Code	Symbol
CHANDRABINDU	161	ँ
ANUSWAR	162	ं
VISARG	163	ः
A	164	अ
AA	165	आ
I	166	इ
II	167	ई
U	168	उ
UU	169	ऊ
RI	170	ऋ
EY	172	ए
AYE	174	ऍ
AI	173	ऐ
OW	176	ओ
AWE	178	ऑ

[1] Characters for which there were no Devang font glyphs have been excluded.

String Representation	ISCII Code	Symbol
AU	177	अउ
KA	179	क
KHA	180	ख
GA	181	ग
GHA	182	घ
NGA	183	ङ
CHA	184	च
CHHA	185	छ
JA	186	ज
JHA	187	झ
JNA	188	ञ
HARD_TA	189	ट
HARD_THA	190	ठ
HARD_DA	191	ड
HARD_DHA	192	ढ
HARD_NA	193	ण
SOFT_TA	194	त
SOFT_THA	195	थ
SOFT_DA	196	द
SOFT_DHA	197	ध
NA	199	न
PA	200	प
PHA	201	फ
BA	202	ब
BHA	203	भ
MA	204	म
YA	205	य
RA	207	र
LA	209	ल
VA	212	व
SHA	213	श
HARD_SHA	214	ष
SA	215	स
HA	216	ह

String Representation	ISCII Code	Symbol
MATRA_AA	218	ा
MATRA_I	219	ि
MATRA_II	220	ी
MATRA_U	221	ु
MATRA_UU	222	ू
MATRA_RI	223	ृ
MATRA_EY	225	े
MATRA_AYE	227	ै
MATRA_AI	226	ॅ
MATRA_OW	229	ो
MATRA_AWE	231	ौ
MATRA_AU	230	ॉ
HALANT	232	्
0	241	०
1	242	१
2	243	२
3	244	३
4	245	४
5	246	५
6	247	६
7	248	७
8	249	८
9	250	९

C.2 Sample Keyboard Map File

The configuration file `KEYBOARDMAPFILE` has information as shown in the following table[2]. The keys in the first column and glyphs in the last column are not part of the configuration file.

Key	ASCII Value	ISCII Codes Mapped	Character Represented
!	33	AYE	ऍ
”	34	HARD_THA	
#	35	HALANT RA	्र
$	36	RA HALANT	र्
%	37	JA HALANT JNA	ज्ञ
&	38	KA HALANT HARD_SHA	क्ष
’	39	HARD_TA	ट
*	42	SHA HALANT RA	श्र
+	43	RI	ऋ
/	47	YA	य
0	48	0	०
1	49	1	१
2	50	2	२
3	51	3	३
4	52	4	४
5	53	5	५
6	54	6	६
9	57	9	९
:	58	CHHA	छ
;	59	CHA	च
<	60	HARD_SHA	ष
=	61	MATRA_RI	ृ
>	62	VIRAM	।
@	64	MATRA_AYE	ॅ
A	65	OW	ओ
C	67	HARD_NA	ण
D	68	A	अ

[2]Characters for which there were no Devang font glyphs have been excluded.

Key	ASCII Value	ISCII Codes Mapped	Character Represented
E	69	AA	आ
F	70	I	इ
G	71	U	उ
H	72	PHA	फ
I	73	GHA	घ
J	74	HARD_RA	र.
K	75	KHA	ख
L	76	SOFT_THA	त
M	77	SHA	श
O	79	SOFT_DHA	ध
P	80	JHA	झ
Q	81	AU	औ
R	82	II	ई
S	83	EY	ए
T	84	UU	ऊ
U	85	NGA	ङ
W	87	AI	ऐ
X	88	CHANDRABINDU	ँ
Y	89	BHA	भ
[	91	HARD_DA	ड
\	92	MATRA_AWE	ॉ
]	93	NUKTA	़
^	94	SOFT_TA HALANT RA	त्र
_	95	VISARG	ः
a	97	MATRA_OW	ो
b	98	VA	व
c	99	MA	म
d	100	HALANT	्
e	101	MATRA_AA	ा
f	102	MATRA_I	ि
g	103	MATRA_U	ु
h	104	PA	प
i	105	GA	ग
j	106	RA	र
k	107	KA	क
l	108	SOFT_TA	त
m	109	SA	स
n	110	LA	ल

Key	ASCII Value	ISCII Codes Mapped	Character Represented
o	111	SOFT_DA	द
p	112	JA	ज
q	113	MATRA_AU	ौ
r	114	MATRA_II	ी
s	115	MATRA_EY	े
t	116	MATRA_UU	ू
u	117	HA	ह
v	118	SOFT_NA	न
w	119	MATRA_AI	ै
x	120	ANUSWAR	ं
y	121	BA	ब
{	123	HARD_DHA	ढ
—	124	AWE	आॅ
}	125	JNA	ञ

C.3 Sample Font Table File

The configuration file `FONTTABLEFILE` has information as shown in the following table[3]. The glyphs in the last column are not part of the configuration file.

ISCII Characters	Intermediate Glyph No.	Character Represented
CHANDRABINDU	1	ँ
ANUSWAR	2	ं
VISARG	3	ः
A	4	अ
AA	5	आ
I	6	इ
II	7	ई
U	8	उ
UU	9	ऊ
RI	10	ऋ
EY	12	ए
AI	13	ऐ
AYE	14	ऍ
OW	16	ओ
AU	17	औ
AWE	18	ऑ
KA	19	क
KHA	20	ख
GA	21	ग
GHA	22	घ
NGA	23	ङ
CHA	24	च
CHHA	25	छ
JA	26	ज
JHA	27	झ
JNA	28	ञ
HARD_TA	29	ट
HARD_THA	30	ठ

[3]Characters for which there were no Devang font glyphs have been excluded.

ISCII Characters	Intermediate Glyph No.	Character Represented
HARD_DA	31	ड
HARD_DHA	32	ढ
HARD_NA	33	ण
SOFT_TA	34	त
SOFT_THA	35	थ
SOFT_DA	36	द
SOFT_DHA	37	ध
SOFT_NA	38	न
PA	40	प
PHA	41	फ
BA	42	ब
BHA	43	भ
MA	44	म
YA	45	य
RA	47	र
LA	49	ल
VA	52	व
SHA	53	श
HARD_SHA	54	ष
SA	55	स
HA	56	ह
KA HALANT	58	क्
KHA HALANT	59	ख्
GA HALANT	60	ग्
GHA HALANT	61	घ्
NGA HALANT	62	ङ्
CHA HALANT	63	च्
JA HALANT	65	ज्
JHA HALANT	66	झ्
JNA HALANT	67	ञ्
HARD_TA HALANT	68	ट्
HARD_THA HALANT	69	ठ्
HARD_DA HALANT	70	ड्
HARD_DHA HALANT	71	ढ्

ISCII Characters	Intermediate Glyph No.	Character Represented
HARD_NA HALANT	72	ण्
SOFT_TA HALANT	73	त्
SOFT_THA HALANT	74	थ्
SOFT_DA HALANT	75	द्
SOFT_DHA HALANT	76	ध्
SOFT_NA HALANT	77	न्
PA HALANT	79	प्
PHA HALANT	80	फ्
BA HALANT	81	ब्
BHA HALANT	82	भ्
MA HALANT	83	म्
YA HALANT	84	य्
LA HALANT	88	ल्
VA HALANT	91	व्
SHA HALANT	92	श्
SA HALANT	94	स्
HA HALANT	95	ह्
MATRA_AA	97	ा
MATRA_I	98	ि
MATRA_II	99	ी
MATRA_U	100	ु
MATRA_UU	101	ू
MATRA_RI	102	ृ
MATRA_EY	104	े
MATRA_AI	105	ै
MATRA_AYE	106	ॅ
MATRA_OW	108	ो
MATRA_AU	109	ॉ
MATRA_AWE	110	ौ
KA NUKTA	111	क़
KHA NUKTA	112	ख़
GA NUKTA	113	ग़
JA NUKTA	114	ज़
HARD_DA NUKTA	115	ड़
HARD_DHA NUKTA	116	ढ़
PHA NUKTA	117	फ़
HALANT	125	्

ISCII Characters	Intermediate Glyph No.	Character Represented
NUKTA	126	़
RA HALANT	86	र्
HALANT RA	127	्र
VIRAM	129	।
0	130	०
1	131	१
2	132	२
3	133	३
4	134	४
5	135	५
6	136	६
7	137	७
8	138	८
9	139	९
KA HALANT HARD_SHA HALANT	140	क्ष्
KA HALANT HARD_SHA	141	क्ष
JA HALANT JNA	142	ज्ञ
SOFT_DA HALANT GHA	143	द्घ
SOFT_DA HALANT SOFT_DHA	144	द्ध
GA HALANT RA	145	ग्र
SOFT_DA HALANT VA	146	द्व
PA HALANT RA	147	प्र
SOFT_TA HALANT RA	149	त्र
RA MATRA_UU	150	रू
RA MATRA_U	151	रु
SOFT_TA HALANT RA	166	त्र
SOFT_TA HALANT RA HALANT	167	त्र्
SOFT_NA HALANT SOFT_NA	171	न्न
PA HALANT RA	173	प्र
PHA HALANT RA	175	फ्र
SHA HALANT RA	183	श्र
SHA HALANT RA HALANT	184	श्र्

C.4 Sample Font File Table

The configuration file `FONTTABLEFILE` has information as shown in the following table[4]. The glyphs in the last column are not part of the configuration file.

Intermediate Glyph No.	Actual Glyph Codes	Glyph Produced
1	196	ँ
2	198	ं
3	38	ः
4	43	अ
5	43 201	आ
6	60	इ
7	60 199	ई
8	61	उ
9	62	ऊ
10	64	ऋ
12	66	ए
13	66 228	ऐ
14	66 236	ऍ
16	43 201 228	ओ
17	43 201 232	औ
18	43 201 236	ऑ
19	69	क
20	74 201	ख
21	77 201	ग
22	80 201	घ
23	82	ङ
24	83 201	च
25	85	छ
26	86 201	ज
27	90 201	झ
28	92 201	ञ
29	93	ट

[4]Characters for which there were no Devang font glyphs have been excluded.

Intermediate Glyph No.	Actual Glyph Codes	Glyph Produced
30	96	ठ
31	98	ड
32	102	ढ
33	104 201	ण
34	105 201	त
35	108 201	थ
36	110	द
37	118 201	ध
38	120 201	न
40	123 201	प
41	161	फ
42	164 201	ब
43	166 201	भ
44	168 201	म
45	170 201	य
47	174	र
49	177 201	ल
52	180 201	व
53	182 201	श
54	185 201	ष
55	186 201	स
56	189	ह
58	67	क्
59	74	ख्
60	77	ग्
61	80	घ्
62	82 194	ङ्
63	83	च्
65	86	ज्
66	90	झ
67	92	ञ्
68	93 194	ट्
69	96 194	ठ्
70	98 194	ड्
71	102 194	ढ्

Intermediate Glyph No.	Actual Glyph Codes	Glyph Produced
72	104	ण
73	105	त
74	108	श
75	110 194	द्
76	118	ङ
77	120	न
79	123	ट
80	125	फ
81	164	ढ
82	166	ष
83	168	म
84	170	ए
86	199	्
88	177	ल
91	180	ठ
92	182	श्र
94	186	स
95	189 194	ह्
97	201	ा
98	202	ि
99	210	ी
100	214	ु
101	218	ू
102	222	ृ
104	228	े
105	232	ें
106	236	ै
108	201 228	ो
109	201 232	ों
110	201 236	ौ
111	70	क़
112	75 201	ख़
113	78 201	ग़
114	87 201	ज़
115	99	ड़
116	103	ढ़
117	162	फ़
125	194	्

Intermediate Glyph No.	Actual Glyph Codes	Glyph Produced
126	195	ृं
86		
127	197	^
129	42	ा
130	48	०
131	49	१
132	50	२
133	51	३
134	52	४
135	53	५
136	54	६
137	55	७
138	56	८
139	57	९
140	73	क्ष्
141	73 201	क्ष
142	89 201	ज्ञ
143	116	द्य
144	114	द्ध
145	79 201	ग्न
146	117	द्ध
147	124 201	प्र
149	106 201	त्र
150	176	रू
151	175	रु
166	106 201	त्र
167	106	त्र्
171	122 201	न्न
173	124 201	प्र
175	163	फ्र
183	184 201	श्र
184	184	श्र्