Technical White Paper v0.5

Peter Lawrey, 20 Feb 2018
Based on content from https://github.com/OpenHFT/Chronicle-Accelerate/tree/master/rfc

Block Message Workflow

Messages are passed around the platform to package up transactions into a block, vote on the order of those blocks, then choose and order.

Inbound messages

Commands and events are passed to the blockchain after being validated by the Fast Path component.

Queries and responses are never passed to the blockchain. They are only exchanged between a requesting client and a server.

Note	Servers act as client to other servers blurring the distinction. For the purpose of the discussion, the client queries and the server responds.

Time is divided into rounds of equal length, for example 10 milli-seconds. At the end of each round, new blocks are added to the end of the chain determining the order all servers should process the blocks.

This component takes all message types.

Fast Path Component

The Fast Path component validates commands and in some cases handles the command itself directly. Initially this will be very few commands, however the aim is to handle as much as 95% of commands in the fast path, avoiding the need to wait for the blockchain.

Once a command has been validated, a command is passed to the blockchain. The command might be different if the fast path handles it, or the same if not.

This component takes all message types.

Chainer

The Chainer takes commands from the Fast Path and events from the Post BlockChain Processor, then adds them to a transaction block.

This block is chained from the last block the same server produced. It is after:

- every other block this server produced.
- every block already added to the chain.

However, it's order relative to new blocks produced by other servers needs to be determined in the voting phase.

This component takes all message types and produces blocks.

Voting Component

The voting component looks at all the new blocks it has produced, or have been replicated to it, and proposes a new end of round block event.

The blocks in this proposed extension to the chain of blocks, come after all the blocks already in the tree, and in eventTime chronological order within this proposal.

Note	It is possible and likely some blocks will be out of order by time between rounds of voting.

This component only processes transaction blocks and produces a vote message.

VoteTaker Component

The vote taker listens to votes in a given round. Once a majority of servers vote on the same proposal, a new end of round block event is created. The round is abandoned if:

- a majority cannot be reached.
- a majority is reached on a later round.

This component only processes the vote messages and produces a EndOfRoundBlockNode.

BlockReplayer Component

The Block Replayer, replays the events in order to the Post BlockChain component which determines the commands outcomes.

This component processes Blocks and EndOfRoundBlocks and produces all the messages in the blocks.

PostBlockChainProcessor Component

This component processes the messages and can issue any message.

FinalMessageRouter Component

This component takes messages and passes them to the gateway, another server, or passes them to the Chainer.

XCL BlockChain

The XCL BlockChain is an ordered collection of messages as commands + events and queries + responses. Its structure contains:

- Signed Messages.
- A Transaction Block which is a list of messages produced by a single node.
- A End of Round Block which describes the order of these blocks.

All data is in little endian. All strings use UTF-8 encoding.

Note	Events must come from a trusted source. i.e. a recognised node in the current cluster, or a recognised node in the previous week's cluster.

Message types

Message types are defined in the MethodIds class.

Trailing zeros

All messages have notionally endless zeros at the end. If the schema changes and a field is added, it should be filled with 0 bytes; i.e. false, empty string, 0, or 0.0 as appropriate. If the message sent has more bytes than expected, they may be silently ignored.

Note	The signature will always include only the bytes actually written and not include the notional extra zero bytes.

Signed Messages

Messages written are either commands (an instruction to take an action which might be rejected), or events (something which has happened, might not need an action but cannot be rejected).

The message format looks like:

Byte offset	Field
0 - 63	Signature
64 - 71	Source address

72 - 79	Event timestamp
80 - 80	Protocol version == 1 (1.)
81 - 81	Message type (for version)
82 - 83	Week number
84 - 87	Block in week for source
84+	The rest of the message depending on the message type

1. In the future, two protocol versions might be supported while an old one is deprecated.

Transaction Block Event (message type = 0x01)

The transaction block is a signed message which contains zero or more messages.

Byte offset	Field
0 - 81	Message header
82 - 83	Week number
84 - 87	Block in week for source (1.)
88 - 89	The first message length
90+	The first message
Later	Additional messages

1. Allows for 4 billion blocks per week, per source.

Within the transaction block, each message is preceded by a message length which is 2 byte unsigned length.

Transaction Block Gossip Event (message type = 0x02)

The message publishes which blocks this node has received.

Transaction Block Vote Event (message type = 0x03)

The message publishes a vote on what the next End of Round Block Event should be.

End Of Round Block Event (message type = 0x04)

The transaction block is a signed message which contains zero or more messages.

Byte offset	Field
0 - 81	Message header
82 - 83	Week number
84 - 87	Block in week for source
88 - 95	The first source address
96 - 99	The first block number
100 - 103	The first block status (1.)
104+	Any additional block entries

1. The block status records how many times it has been replicated and whether it was timed out or not.

Opening Balance Event (0x05)

This event holds the opening balance for an address at the start of the week. It is also the event produced at the end of the week to record the balance.

This message:

- is an event to set the initial state of an address.
- is dumped as part of the snap shot at the end of each week and loaded at the start of the next week.

- must come from a trusted source or it will be rejected.

The message contains:

- the list of currencies and balances

Fees Event (0x06)

This event records the fee structure determined by the AI at the start of the week.

Exchange Rates Event (0x07)

This event lists all the weekly exchange rates with mid price. It is used to calculate fees when the balance is in other currencies.

Service Nodes Event (0x08)

This event lists all the nodes in a cluster for a service.

Block Subscription Query (0x0f)

Requests a block be sent from a given block number in a week, for that node.

Note: If the requested block number is -1, all the checkpoint blocks (Weekly Events written) from the previous week will also be sent.

Runtime Events

These events can occur at any time and on any chain.

Application Message Event (0x10)

This event records that something went wrong, most likely unable to be handled automatically.

When an event fails to be processed, it could trigger one of these events.

Command Failed Message Event (0x11)

This event records that an error occurred processing a command in a way which might be processed automatically.

When a command fails to be processed it could trigger one of these events.

Query Failed Response (0x12)

This response occurs when there is an error in processing a query in a way which might be processed automatically.

When a query fails to be processed it could trigger one of these events.

Note	Responses are not written to a chain, only the connection sending the query.

Main Chain Command/Queries

Main Chain Commands are commands which are used to execute Global operations; i.e. operations which apply to all regions in the world. The results of executing these commands are new transactions being executed and then persisted to the Main Blockchain. The Main Blockchain holds cross-regional data like …

- Create New Address Command (0x20)
- Cluster Transfer Value Step1 Command (0x21)
- Cluster Transfer Value Step2 Command (0x22)
- Cluster Transfer Value Step3 Command (0x23)

Range 0x20 - 0x2f

Create New Address Command (0x20)

This message is a command to request that a new account be created. This includes the public key and the region in which to create the address:

Success	Error
Address Information Event (0x30)	Command Failed Event (0x11)

Cluster Transfer Value Step1 Command (0x21)

This message is a command to transfer value from one cluster to another, via the main chain.

The first step is to approve money be taken out of an account in one region/cluster.

Success	Error
Cluster Transfer Value Step2 Command (0x22)	Command Failed Event (0x11)

Cluster Transfer Value Step2 Command (0x22)

This message is a command to transfer value from one cluster to another, via the main chain.

The second step is to pass the transfer main chain can reject it if a node or cluster fails risk checks; e.g. transfers too much money, too quickly.

Success	Error
Cluster Transfer Value Step3 Command (0x23)	Command Failed Event (0x11)

Cluster Transfer Value Step3 Command (0x23)

This message is a command to transfer value from one cluster to another via the main chain.

The last step is to notify the target cluster to add to the balance of an address.

Success	Error
Cluster Transfer Value Step3 Event (0x33)	Application Message Event (0x10)

Clusters Status Query (0x2f)

This message is a query for all the known clusters and the services they provide.

Success	Error
Cluster Transfer Value Event (0x31)	Query Failed Response (0x12)

Main Chain Events

Main Chain Events are events which are sent as a result of executing Main Chain Commands. These events are confirmation of transactions performed at the global level; i.e. transactions which apply to all regions in the world. These events are published after the transactions have been executed, confirmed and persisted to the Main Blockchain.

Range 0x30 - 0x3f

Create New Address Event (0x30)

This message:

- is an event from the main chain to set the reference information of an address.
- is dumped as part of the snap shot at the end of each week and loaded at the start of the next week.
- must come from the main chain or it will be rejected.

The message includes:

- the public key of the address.
- the list of verifiable facts about the account.

Cluster Transfer Step3 Event (0x33)

Value was successfully added to an address after transferring it from another cluster.

Clusters Status Response (0x3f)

A message detailing all the known clusters, their services and their host connection details.

Regional Chain Commands

Regional Chain Commands are commands that are relevant to a specific region, as opposed to Main Chain Commands that apply globally to all regions across the world.

The results of these commands are persisted to the Regional Blockchain for the appropriate region.

Range 0x40 - 0x4f

Transfer Value Command (0x40)

This message is a command to transfer value from one address to another in the same chain.

The first step is to approve money be taken out of an account in one region/cluster.

Success	Error
Transfer Value Event (0x50)	Command Failed Event (0x11)

Subscription Query (0x4c)

Success	Error
Subscription Success Response (0x5c)	Query Failed Response (0x12)

Current Balance Query (0x4d)

The Current Balance Query message is a query for the current balance of an account address. An account address has a number of balances, one for each currency or asset held in the account. This message is a request for all balances for all currencies held within a given account.

Success	Error
Current Balance Response (0x5d)	Query Failed Response (0x12)

Exchange Rate Query (0x4e)

This message is a query for the latest Mid rate between 2 currencies, for example, the XCL/USD exchange rate.

The Mid rates are calculated by the appropriate Exchange service for that currency/asset pair. For example, the XCL/USD exchange rate will be calculated by the XCL/USD Exchange service. This service will return the Mid rate for 1,000,000 USD (?)

Success	Error
Exchange Rate Response (0x5e)	Query Failed Response (0x12)

Cluster Status Query (0x4f)

The Cluster Status Query message is a query for the status of the nodes in the current cluster. A node can have one of these status:

- WAITING_FOR_APPROVAL
- APPROVED_AND_NEVER_RUN
- RUNNING
- RUNNING_AND_DOING_ROUND_PROCESSING
- NOT_RUNNING
- DISABLED

Success	Error
Cluster Status Response (0x5f)	Query Failed Response (0x12)

Regional Chain Events

Regional Chain Events are events which are published in response to Regional Chain Commands.

These are the Regional Chain Events:

- Transfer Value Event (0x50)

Range 0x50 - 0x5f

Note	Events and Responses are always assumed to be successfully processed or ignored unless an Application Message Event (0x10) is produced.

Transfer Value Event (0x50)

The Transfer Value Event message is an event which is sent in response to a Transfer Value Command message.

This event message contains the details of the transfer:

- sender’s account address
- receiver’s account address
- amount transferred
- currency

- timestamp

Regional Chain Responses

Regional Chain Responses are commands which are sent in response to Regional Chain Queries. These response messages are:

- Subscription Success Response (0x5c)
- Current Balance Response (0x5d)
- Cluster Status Response (0x5f)
- Clusters Status Response (0x3f)
- Exchange Rate Response (0x5e)

Subscription Success Response (0x5c)

The Subscription Success Response message is a message which is sent in response to a Subscription Command message.

Current Balance Response (0x5d)

The Current Balance Response message is the message sent in response to the Current Balance Query (0x4d) message.

This message will contain, for a given account address, the list of currencies/assets and the current balance for each of them. For example:

Table 1. Accelerate Account Address: @gb1ndar3bfw93

Currency	Amount
XCL	2,434
GBP	3,493,343
USD	34,893

Exchange Rate Response (0x5e)

The Exchange Rate Response message is sent in response to the Exchange Rate Query (0x4e) message.

This message will contain the latest exchange Mid rates between 2 assets; e.g. currencies, crypto currencies, etc.

For example, a standard response will look something like:

Currencies	mid	spread
USD/XCL	2.3493	0.0011

The Mid rates are calculated by the appropriate Exchange service for that currency/asset pair. For example, the XCL/USD exchange rate will be calculated by the XCL/USD Exchange service. This service will return the Mid rate for 1 million USD

Cluster Status Response (0x5f)

The Cluster Status Response message is the message sent in response to the Cluster Status Query (0x4f) message.

This message contains the status of all the nodes in the current cluster. A node can have one of these status:

- WAITING_FOR_APPROVAL
- APPROVED_AND_NEVER_RUN
- RUNNING
- RUNNING_AND_DOING_ROUND_PROCESSING
- NOT_RUNNING
- DISABLED

Service Chain Commands

Service Chain Commands are commands which are used to perform service type operations like:

- depositing and withdrawing funds.
- placing Market/Limit orders in the Accelerate market.

These commands are:

- Deposit Value Command (0x60)
- Withdraw Value Command (0x61)
- Market Order to Buy/Sell XCL (0x62)
- Limit Order to Buy/Sell XCL (0x63)

- Cancel Order to Buy/Sell XCL (0x64)

These commands are typically issued by a user from the Accelerate Website page.

Service Chain Commands are requests to perform transactions which are persisted to the appropriate regional blockchain; i.e the blockchain for that region.

Range 0x60 - 0x6f

Deposit Value Command (0x60)

This message is a command to make a deposit for an amount of standard real currency, e.g. Sterling Pounds or Dollars, into an Accelerate address account.

A user would typically make this deposit on the Accelerate Website page by making a standard Debit/Credit card payment into their personal Accelerate account address.

Success	Error
Deposit Value Event (0x70)	Command Failed Event (0x11)

Withdraw Value Command (0x61)

The Withdraw Value Command message is a command to make a withdrawal from an Accelerate account address and deposit the funds into a user's personal retail bank account.

A user would typically issue this withdrawal on the Accelerate Website page by requesting a standard bank transfer from their Accelerate account address to their personal retail bank account.

If the user is withdrawing funds held in XCL, then this withdrawal will involve a currency conversion from XCL to the user's home currency; e.g. Sterling Pounds or US Dollars.

Success	Error
Withdraw Value Event (0x71)	Command Failed Event (0x11)

Market Order to Buy/Sell XCL (0x62)

The Market Order Command message is a command which is used to place a Market Order to buy or sell an amount of XCL on the Accelerate market at the current live market price.

A Market Order is a type of Order that executes immediately at the best available price in the market.

A user would typically place their Market Order on the Accelerate Website page.

The result of placing a Market Order is a Execution Report (0x72) which will be executed immediately after the market order is placed at the best available price at the time.

Success	Error
Execution Report (0x72)	Command Failed Event (0x11)

Limit Order to Buy/Sell XCL (0x63)

The Limit Order Command message is a command which is used to place a Limit Order to buy or sell an amount of XCL on the Accelerate market at a specified price.

A Limit Order is a type of order to execute a trade at a given maximum price, if buying, or at a given minimum price, if selling. This given price is called the Limit Price.

A Limit Order may never be executed, but it guarantees that if it is executed, it will be at the specified limit price or better.

A user would typically place their Limit Order on the Accelerate Website page.

After placing a Limit Order, a trade might be executed immediately or at some point in the future or not at all, depending on how aggressive the specified Limit Price is and the currently available live prices in the market.

The result of placing a Limit Order is a Execution Report Event (0x72) but only, if and when, a trade takes place at the requested Limit Price.

Success	Error
Execution Report (0x72)	Command Failed Event (0x11)

Cancel Order to Buy/Sell XCL (0x64)

The Cancel Order Command message is a command to cancel a Market or Limit Order that a user previously placed in the Accelerate market.

Success	Error
Execution Report (0x72)	Command Failed Event (0x11)

Service Chain Events

Service Chain Events are events which are sent confirming that a Service Chain Command was executed successfully. Service Chain Events contain the details of the transaction that was executed.

These events are:

- Deposit Value Event (0x70)
- Withdraw Value Event (0x71)
- Execution Report to Buy/Sell XCL (0x72)

Range 0x70 - 0x7f

Note	Events and Responses are always assumed to be successfully processed or ignored unless an Application Message Event (0x10) is produced.

Deposit Value Event (0x70)

A Deposit Value Event message is a message which is sent as a response to a Deposit Value Command message.

The Deposit Value Event message contains confirmation of all the deposit details including:

- the amount deposited
- the currency
- the user’s account address

Withdraw Value Event (0x71)

A Withdraw Value Event is an event which is sent as a response to a Withdraw Value Command message.

The Withdraw Value Event contains confirmation of the withdrawal details including:

- the amount withdrawn
- the currency
- the user's account address

Execution Report Event (0x72)

An Execution Report message is a message which is sent as a confirmation message of a trade executed in the Accelerate market. The Execution Report Event message contains:

- the amount bought or sold
- the currency pair
- the price of the trade

Execution Reports are sent as a result of placing a Market or Limit Order on the Accelerate market.

Region Codes

Each service is designated to a region, or a set of regions.

Region Codes

Region codes are based on ISO-3166 region codes with some additional services:

- codes starting with a letter are regional country codes.
- codes starting with a 2-currency are exchange "regions".
- codes starting with 3, 4, 5, 6 or 7 are reserved for additional services.

Region Specification

Each region service manages a range of regions based on the following format, as a 32-bit unsigned value:

Bits	Description
31 - 5	Region base

4 - 0	Mask bits

e.g. The region `gbldn/25` could be London in the UK, with a mask of 25 bits.

Region splitting

Initially the only region is 0x00000000 with match every region.

However once this is split in two you get:

```
0x00000001
 - 0x00000002
 - 0x40000002
0x80000001
 - 0x80000002
   - 0x80000003
     - 0x80000004
     - 0x90000004
   - 0xA0000003
- 0xC0000002
  - 0xC0000003
    - 0xC0000004
      - 0xC0000005
      - 0xC8000005
        - 0xC8000006
        - 0xCC000006
    - 0xD0000004
  - 0xE0000003
```

Block Distribution

Transaction are distributed in three stages:

1. They are packed together by each node into blocks.
2. The nodes gossip about which blocks they have received from other nodes.

3. The blocks are assembled into a rounds of blocks from the chains of each node.

Once a node has assembled a rounds of blocks it can process the transactions in order to determine whether they are successful or not.

Reaching Consensus

Consensus is reached by going through stages with the purpose of:

- determining the order transactions will be processed.
- reaching a collective decision in a reasonable time frame.
- reaching consensus even when some nodes are not running or mis-behaving.

Building Blocks

Transactions are assembled into batches called blocks. These blocks are assembled in parallel on each node. These blocks in turn are ordered into a chain of blocks. At this point, all the blocks produced by a node can be said to be in order, however the blocks in these chains could be interlaced in any order.

Proposing An Order For A Round

From the start of the week, the building of the blocks into a single ordered rounds of blocks occurs in rounds.

At the start of a round, each node which has a block that has not be assembled into the rounds of blocks, proposes an order of the blocks to be added and to be processed. This is called gossip about the blocks to be added.

Voting Of The Order Of New Blocks

Each node votes of the proposals for a given round. There are three possible states:

1. At least one node can see the majority of nodes agree on a proposal.
2. At least one node can see that a majority cannot be achieved.
3. No node sees a majority.

In case 1, each node which sees a majority can publish a round of blocks recording the order blocks should be processed.

In case 2, a node can start the next round, though whether it should is not clear.

In case 3, after the round has expired, a new proposal can be made. Once the new proposal has been accepted, it replaces all previous unsuccessful rounds.

Note	After a new round has been proposed but before it has been accepted, an old round might be accepted. When this happens, all the blocks in the old round must come before the blocks proposed in the new round.

Sorting Blocks Within A Round.

Each message has a microsecond resolution timestamp. Within a round, blocks are sorted by this timestamp to minimise how much difference the round process makes when everything is running correctly.

When some nodes are not running ideally, the rounds will ensure the blocks can be placed in order even if:

- the clock of one node is way off.
- nodes are down.
- nodes are very slow.

Each node published the blocks it has and proposes an order for the outstanding blocks it has.

Chain vs Rounds of Blocks

Each node produces a chain of signed blocks containing transactions. However, the order of these blocks relative to one another can be important, especially if an address is attempting to double spend by transferring from two chains at once.

To prevent double spend, each node broadcasts the chain it is producing and gossips about all the chains it has received.

Once a node has detected that a super majority of nodes have gossiped that they have received a block, it can be placed in order.

The order will be based on:

- the order blocks confirmed.
- the block eventTime.
- the signature for the block.

Periodically, once a node has decided the order it broadcasts an End of Round Block Event containing the order of blocks it will be processing.

A process might be need to reconcile what happens if different nodes don’t agree on the order or blocks, especially if they are not running the same code.

Block Activity

A Transaction Block could be produced around every 1 milli-seconds to 1 second. Transaction Blocks are expected to be ~200 B to 100 KB. The soft limit size in the implementation is 1 MB and protocol limit is around 4 GB or ~40 million transactions, however blocks this size are unlikely to be practical. Therefore it would be more efficient to send more, smaller blocks more often as volumes increase.

Gossip about block could occur after every block received and could be 10 - 100x more often, but should be much smaller.

An End Of Round Block could be produced around every 2 ms to 10 seconds and is expected to be less often than the Transaction Blocks.

Note	Transaction Blocks and End of Round Block Event both have block numbers and are part of the same chain for replication purposes.

XCLBase32 specification

XCLBase32 is an extension of hexadecimal for encoding a string of bytes.

There are Base32 specifications such as IETF - RFC4648 for encoding a string of bytes, which starts at the highest bits and includes the length of bytes; the byte length of any training zeros is not important. As such, encoding is aligned to the highest bits.

One of the aims is to produce numbers which can also be used to form words.

Extended Hexadecimal

The encoding is the same as Base16, with the rest of the alphabet except:

- I and O as they can be confused with 1 and 0.
- q and x as these are rare letters.

0, o, O = 0	1, I, L = 1	2 = 2	3 = 3
4 = 4	5 = 5	6 = 6	7 = 7
8, x, X = 8	9, q, Q = 9	a, A = a	b, B = b
c, C = c	d, D = d	e, E = e	f, F = f
g, G = g	h, H = h	i, I = i	j, J = j
k, K = k	m, M = m	n, N = n	p, P = p
r, R = r	s, S = s	t, T = t	u, U = u
v, V = v	w, W = w	y, Y = y	z, Z = z

Note	When decoding an encoded string, multiple characters map to the same value. e.g. A and a are 10, 9, q and Q are 9

Non ASCII extensions

For unicode characters above 128, the character is romanized as either 1 or two encoded tokens as above. If the romanized character would be more than 2 letters, the first and last letters are taken.

(For breverity, not all translations are included here)

See https://github.com/OpenHFT/Chronicle-Accelerate/blob/master/rfc/XCLBase32.adoc for a complete list.

Table 1. ALPHABETIC_PRESENTATION_FORMS

ﬠ = yodh	ﬡ = alta	ﬢ = wida	ﬣ = widd
ﬤ = widh	ﬥ = widk	ﬦ = widl	ﬧ = widf
ﬨ = widr	﬩ = widt	שׁ = altp	ﬠ, ﬩ = shis
﬩, ﬩, ﬩ = shid	ﬡ = alep	ﬡ = aleq	ﬡ = alem
ﬢ = betd	ﬣ = gimd	ﬤ = dald	ﬤ = hem
ﬥ = vavd	ﬦ = zayd	ﬧ = tetd	ﬨ = yodd
﬩ = fink	שׁ = kafd	שׂ = lamd	שּׁ = memd
שּׂ = nund	אַ = samd	אָ = finp	אּ = ped
בּ = tsad	גּ = qofd	דּ = resd	הּ = tavd
וּ = vavh	זּ = betr	﬷ = kafr	טּ = per

Table 2. ARABIC

ء, ء = hama	آ, ى, آ, آ, ى, ى = alem	أ, إ, أ, أ, إ, إ = aleh	ؤ, ؤ, ؤ = wawh
ئ, ئ, ئ, ئ, ئ = yehh	ا, ا, ا = alef	ب, ب, ب, ب, ب = beh	ة, ة, ة, ة = tehm
ت, ت, ت, ت, ت = teh	ث, ث, ث, ث, ث = theh	ج, ج, ج, ج, ج = jeem	ح, ح, ح, ح, ح = hah
خ, خ, خ, خ, خ = khah	د, د, د = dal	ذ, ذ, ذ = thal	ر, ر, ر = reh
ز, ز, ز = zain	س, س, س, س, س = seen	ش, ش, ش, ش, ش = shen	ص, ص, ص, ص, ص = sad
ض, ض, ض, ض, ض = dad	ط, ط, ط, ط, ط = tah	ظ, ظ, ظ, ظ, ظ = zah	ع, ع, ع, ع, ع = ain
غ, غ, غ, غ, غ = ghan	ف, ف, ف, ف, ف, ف = feh	ق, ق, ق, ق, ق = qaf	ك, ك, ك, ک, ک = kaf
ل, ل, ل, ل, ل = lam	م, م, م, م, م = meem	ن, ن, ن, ن, ن = noon	ه, ه, ه, ه, ه = heh
و, و, و = waw	ي, ي, ي, ي, ي = yeh	٠, ٠ = 0	١, ١ = 1
٢, ٢ = 2	٣, ٣ = 3	٤, ۴ = 4	٥, ۵ = 5
٦, ۶ = 6	٧, ٧ = 7	٨, ٨ = 8	٩, ٩ = 9
□, أ, إ, ٱ, ٱ = alew	ٔ, ٰا, ٶ, ى = high	ۇ, ۇ = uh	ٹ, ٹ, ٹ, ٹ, ٹ, ٹ, ٹ, ٹ, ٹ = tteh
ٻ, ٻ, ٻ, ٻ, ٻ = beeh	ٽ = tehr	ٿ = teht	پ, پ, پ, پ, پ = peh
ٿ, ٿ, ٿ, ٿ, ٿ = tehh	ڀ, ڀ, ڀ, ڀ, ڀ = behh	ځ = hahh	ڂ, ڂ = haht
ڃ, ڃ, ڃ, ڃ, ڃ = nyeh	ڄ, ڄ, ڄ, ڄ, ڄ = dyeh	چ, چ, چ, چ, چ, چ, چ, چ, چ = tchh	ڈ, ڈ, ڈ, ڈ, ڈ = ddal

ﮋ = dalr	ﮈ ,ﮉ = dald	ﮈ ,ﮉ ,ﮈ = dahl	ﮉ ,ﮈ ,ﮉ = dul
ﮊ = dalt	ﮋ = dalf	ﮋ ,ﮊ ,ﮋ = rreh	ﮍ ,ﮋ = rehv
ﮌ = rehr	ﮍ ,ﮌ = rehd	ﮋ = reht	ﮊ ,ﮋ ,ﮊ = jeh
ﮍ = rehf	ﮋﮊ = seed	ﭙ ,ﭘ = seet	ﺿ ,ﺻ = sadt
ﻇ = taht	ﻏ = aint	ﻓ ,ﻔ = fehd	ﯗ ,ﯘ ,ﯙ ,ﯚ = veh
ﭖ = feht	ﻗ ,ﻘ ,ﻖ ,ﻕ = pehh	ﻗ = qafd	ﻕ = qaft
ﮎ ,ﮏ ,ﮑ ,ﮐ = kehh	ﮏﮐ = swak	ﮒ = kafr	ﮏ = kafd
ﯓ ,ﯔ ,ﯕ ,ﯖ = ng	ﮐ = kaft	ﮓ ,ﮔ ,ﮕ ,ﮖ = gaf	ﮔ = gafr
ﯖ ,ﯕ ,ﯔ ,ﯓ = ngoh	ﮓ ,ﮔ = gaft	ﮓ ,ﮔ ,ﮕ ,ﮖ = gueh	ﻟﻵ = lamv
ﻟ = lamd	ﻟﭧ ,ﻟ = lamt	ﮞ = nood	ﮟ ,ﮞ ,ﮟ = noog
ﮞ ,ﮟ ,ﮞ ,ﮟ = rnon	ﻧ = noor	ﻧ = noot	ﻫ ,ﻫ ,ﻫ ,ﻫ ,ﻫ = hehd
ﭺ = tchd	ﮧ ,ﮦ ,ﮦ = hehy	ﮩ ,ﮨ ,ﮩ ,ﻫ ,ﻫ = hehg	ﯙ = wawr
ﯚ ,ﯙ ,ﯙ = kiro	ﯗ ,ﯘ ,ﯗ = oe	ﯘ ,ﯗ ,ﯘ = u	ﯛ ,ﯚ ,ﯛ = yu
ﯛ ,ﯚ ,ﯛ = kiry	ﯘ = wawt	ﯗ ,ﯘ ,ﯗ = ve	ﯼ ,ﯽ ,ﯼ ,ﯾ = fary
ﯽ ,ﯼ = yeht	ﯼ = yehv	ﯙ = wawd	ﯾ ,ﯿ ,ﯾ ,ﯽ ,ﭙ = e
ﮮ ,ﮯ ,ﮮ ,ﮯ ,ﮮ = yehb	ﻩ = ae	ﭻ = shed	ﺽ = dadd
ﻍ = ghad	ﺳ = sina	ﺱ = sinp	ﯨ ,ﯩ = uigk

Table 3. ARMENIAN

Table 4. BENGALI
Table 5. BOPOMOFO

Table 6. CHEROKEE

Table 7. CJK_COMPATIBILITY

Table 8. CJK_COMPATIBILITY_IDEOGRAPHS

Table 9. CJK_RADICALS_SUPPLEMENT

□ = rept	□ = clif	□ = seco	乚, □ = sect
亻 = pern	□ = box	八 = tabe	□ = knio
刂 = knit	□ = divn	已 = seal	丷, 長 = one
丷, 長 = two	兀 = lamo	允, 尢 = lamt	□ = lamf
巳 = snae	幺 = thrd	旦 = snoo	彐 = snot
忄 = heao	小 = heat	扌 = hand	攵 = rap
无 = choe	□ = sun	□ = moon	歹 = deah
母 = motr	民 = civn	氵 = wato	氷 = watt
灬 = fire	爫 = pawo	□ = pawt	犭 = halt
□ = cow	犭 = dog	玉 = jade	疋 = bolo
罒 = eye	示 = spio	礻 = spit	艸 = bamo
□, □ = silk	罔 = neto	罒, 网 = nett	□ = netf

□ = mesh	□ = shep	羊 = ram	□ = ewe
岁 = old	□ = bruo	□ = brut	月 = meat
臼 = morr	艹 = grao	++, 艹 = grat	虎 = tigr
礻 = clos	覀 = weso	西 = west	□ = see
角, □ = horn	□ = speh	□ = shel	足 = foot
□ = cart	辶 = walk	辶 = walo	□ = walt
阝 = city	□ = gold	□ = long	□ = gate
□ = mouo	阝 = mout	雨 = rain	青 = blue
□ = tanl	□ = leaf	□ = wind	□ = fly
食 = eato	飠, 食 = eatt	□ = eat	□ = head
□ = hore	□ = bone	鬼 = ghot	□ = fish
□ = bird	□ = salt	麦 = whet	黄 = yelw
□ = frog	斉, □ = even	歯, □ = tooh	竜, □ = dran

Table 10. CJK_SYMBOLS_AND_PUNCTUATION

〇 = 0	丨 = 1	〻 = 2	〻 = 3
乂 = 4	𠂉 = 5	丄 = 6	亠 = 7
亖 = 8	夕 = 9	□ = 10	□ = 20

Table 11. CYRILLIC

а, А = a	б, Б = be	в, В = ve	г, Г = ghe
д, Д = de	е, Е = ie	ж, Ж = zhe	з, З = ze
и, И = i	й, Й = shoi	к, К = ka	л, Л = el
м, М = em	н, Н = en	о, О = o	п, П = pe
р, Р = er	с, С = es	т, Т = te	у, У = u
ф, Ф = ef	х, Х = ha	ц, Ц = tse	ч, Ч = che
ш, Ш = sha	щ, Щ = shca	ъ, Ъ = hars	ы, Ы = yeru
ь, Ь = sofs	э, Э = e	ю, Ю = yu	я, Я = ya
ѐ, Ѐ = ieg	ё, Ё = io	ђ, Ђ = dje	ѓ, Ѓ = gje
є, Є = ukri	ѕ, Ѕ = dze	і, І = byeu	ї, Ї = yi
ј, Ј = je	љ, Љ = lje	њ, Њ = nje	ћ, Ћ = tshe
ќ, Ќ = kje	ѝ, Ѝ = ig	ў, Ў = shou	џ, Џ = dzhe
ѡ, Ѡ = omea	ѣ, Ѣ = yat	ѥ, Ѥ = iote	ѧ, Ѧ = lity
ѩ, Ѩ = iotl	ѫ, Ѫ = bigy	ѭ, Ѭ = iotb	ѯ, Ѯ = ksi
ѱ, Ѱ = psi	ѳ, Ѳ = fita	ѵ, Ѵ = izha	ѷ, Ѷ = izhg
ѹ, Ѹ = uk	ѻ, Ѻ = rouo	ѽ, Ѽ = omet	ѿ, Ѿ = ot
ҁ, Ҁ = kopa	ҍ, Ҍ = sems	ҏ, Ҏ = ert	ґ, Ґ = gheu

ғ, Ғ = ghes	ɓ, Ɓ = ghem	ж, Ж, ӝ, Ӝ = zhed	ʒ, Ʒ, ӟ, Ӟ = zed
қ, Қ = kad	к, К = kav	ҟ, Ҟ = kas	к, К = bask
ң, Ң = end	ҧ, Ҧ = pem	ҩ, Ҩ = abkh	ҫ, Ҫ = esd
т, Т = ted	ү, Ү, ұ, Ұ = stru	х, Х = had	ч, Ч, ӵ, Ӵ = ched
ч, Ч = chev	ҳ, Ҳ = shha	ҽ, Ҽ, ҿ, Ҿ = abkc	ӂ, Ӂ = zheb
ҡ, Ҡ = kah	ҥ, Ҥ = enh	ҷ, Ҷ = khac	ӑ, Ӑ = ab
ӓ, Ӓ = ad	ӗ, Ӗ = ieb	ә, Ә = scha	ӛ, Ө = schd
ʒ, Ʒ = abkd	ӣ, Ӣ = im	й, Й = id	ӧ, Ӧ = od
ө, Ө, ӫ, Ӫ = baro	ӭ, Ӭ = ed	ӯ, Ӯ = um	ӱ, Ӱ = ud

Table 12. DEVANAGARI

Table 13. DINGBATS

❡ = ornt	❶, ①, ➊ = 1	❷, ②, ➋ = 2	❸, ③, ➌ = 3
❹, ④, ➍ = 4	❺, ⑤, ➎ = 5	❻, ⑥, ➏ = 6	❼, ⑦, ➐ = 7

Table 14. ENCLOSED_ALPHANUMERICS

①, ⑴, 🄁 = 1	②, ⑵, 🄂 = 2	③, ⑶, 🄃 = 3	④, ⑷, 🄄 = 4
⑤, ⑸, 🄅 = 5	⑥, ⑹, 🄆 = 6	⑦, ⑺, 🄇 = 7	⑧, ⑻, 🄈 = 8
⑨, ⑼, 🄉 = 9	⑩, ⑽ = 10	⑪, ⑾ = 11	⑫, ⑿ = 12
⑬, ⒀ = 13	⑭, ⒁ = 14	⑮, ⒂ = 15	⑯, ⒃ = 16

⑰, ⑴⑺ = 17	⑱, ⑴⑻ = 18	⑲, ⑴⑼ = 19	⑳, ⑵⑽ = 20
⒜, ⓐ, Ⓐ = a	⒝, ⓑ, Ⓑ = b	⒞, ⓒ, Ⓒ = c	⒟, ⓓ, Ⓓ = d
⒠, ⓔ, Ⓔ = e	⒡, ⓕ, Ⓕ = f	⒢, ⓖ, Ⓖ = g	⒣, ⓗ, Ⓗ = h
⒤, ⓘ, Ⓘ = i	⒥, ⓙ, Ⓙ = j	⒦, ⓚ, Ⓚ = k	⒧, ⓛ, Ⓛ = l
⒨, ⓜ, Ⓜ = m	⒩, ⓝ, Ⓝ = n	⒪, ⓞ, Ⓞ = o	⒫, ⓟ, Ⓟ = p
⒬, ⓠ, Ⓠ = q	⒭, ⓡ, Ⓡ = r	⒮, ⓢ, Ⓢ = s	⒯, ⓣ, Ⓣ = t
⒰, ⓤ, Ⓤ = u	⒱, ⓥ, Ⓥ = v	⒲, ⓦ, Ⓦ = w	⒳, ⓧ, Ⓧ = x

Table 15. ENCLOSED_CJK_LETTERS_AND_MONTHS

Table 16. ETHIOPIC

Table 17. GEORGIAN

Table 18. GREEK

ΐ, ϊ, Ϊ, ι, ῖ, ῗ, ῒ, ΐ, Ῐ, Ῑ, Ὶ, ΐ, ῗ = iotd	ά, Ά = alpt	έ, Έ = epst	ή, Ή = etat
ί, Ί = iott	ΰ, ϋ, Ÿ, ῦ, ῦ, ῧ, Ὺ, ῠ, Ὺ, ῡ, Ὺ, ῧ, ῦ = upsd	α, Α = alpa	β, Β = beta
γ, Γ = gama	δ, Δ = dela	ε, Ε = epsn	ζ, Ζ = zeta
η, Η = eta	θ, Θ = thea	ι, Ι = iota	κ, Κ = kapa
λ, Λ = lama	μ, Μ = mu	ν, Ν = nu	ξ, Ξ = xi
ο, Ο = omin	π, Π = pi	ρ, Ρ = rho	ς, Σ = fins

σ, Σ = siga	τ, Τ = tau	υ, Υ = upsn	φ, Φ = phi
χ, Χ = chi	ψ, Ψ = psi	ω, Ω = omea	ό, Ό = omit
ύ, ΄Υ = upst	ώ, ΄Ω = omet	ς, Ϛ = stia	ϝ, Ϝ = diga
ϗ, Ϟ = kopa	ϡ, Ϡ = sami	ϣ, Ϣ = shei	ϥ, Ϥ = fei
ϧ, Ϧ = khei	ϩ, Ϩ = hori	ϫ, Ϫ = gana	ϭ, Ϭ = shia
ϯ, Ϯ = dei	j = yot	ἀ, Ἀ, ἁ, Ἁ, ἂ, Ἂ, ἃ, Ἃ, ἄ, Ἄ, ἅ, Ἅ, ἆ, Ἆ, ἇ, Ἇ = alpp	ἀ, Ἀ, ἁ, Ἁ, ἂ, Ἂ, ἃ, Ἃ, ἄ, Ἄ, ἅ, Ἅ, ἆ, Ἆ, ἇ, Ἇ = alpd
ἐ, Ἐ, ἑ, Ἑ, ἒ, Ἒ = epsp	ἐ, Ἐ, ἑ, Ἑ, ἒ, Ἒ = epsd	ἠ, Ἠ, ἡ, Ἡ, ἢ, Ἢ, ἣ, Ἣ, ἤ, Ἤ, ἥ, Ἥ, ἦ, Ἦ, ἧ, Ἧ = etap	ἠ, Ἠ, ἡ, Ἡ, ἢ, Ἢ, ἣ, Ἣ, ἤ, Ἤ, ἥ, Ἥ, ἦ, Ἦ, ἧ, Ἧ = etad
ἰ, Ἰ, ἱ, Ἱ, ἲ, Ἲ, ἳ, Ἳ, ἴ = iotp	ὀ, Ὀ, ὁ, Ὁ, ὂ, Ὂ = omip	ὀ, Ὀ, ὁ, Ὁ, ὂ, Ὂ = omid	ὐ, ὑ, ὒ, ὓ, ὔ = upsp
ὠ, Ὠ, ὡ, Ὡ, ὢ, Ὢ, ὣ, Ὣ, ὤ, Ὤ, ὥ, Ὥ, ὦ, Ὦ, ὧ, Ὧ = omep	ὠ, Ὠ, ὡ, Ὡ, ὢ, Ὢ, ὣ, Ὣ, ὤ, Ὤ, ὥ, Ὥ, ὦ, Ὦ, ὧ, Ὧ = omed	ᾰ, Ᾰ, ᾱ, Ᾱ, ᾲ = alpv	ᾰ, Ᾰ, ᾱ = alpo
ὲ, Ὲ = epsv	έ, Έ = epso	ὴ, Ὴ, ῂ = etav	ή, Ή, ῄ = etao
ὶ, Ὶ, ῐ, Ῐ = iotv	ί, Ί = ioto	ὸ, Ὸ = omiv	ό, Ό = omio
ὺ, Ὺ, ῠ, Ῠ = upsv	ύ, Ύ = upso	ὼ, Ὼ, ῲ = omev	ώ, Ώ, ῴ = omeo
ᾱ, Ᾱ = alpm	ᾳ, ᾼ = alpy	ῃ, ῌ = etay	ῑ, Ῑ = iotm
ῡ, Ῡ = upsm	ῥ = rhop	ῤ, Ῥ = rhod	ῳ, ῼ = omey

Table 19. GUJARATI

Table 20. GURMUKHI

Table 21. HALFWIDTH_AND_FULLWIDTH_FORMS

Table 22. HANGUL_COMPATIBILITY_JAMO

Table 23. HEBREW

א = alef	ב = bet	ג = giml	ד = dalt
ה = he	ו = vav	ז = zayn	ח = het
ט = tet	י = yod	ך = fink	כ = kaf
ל = lamd	ם = finm	מ = mem	ן = finn
נ = nun	o = samh	ע = ayin	ף = finp
פ = pe	ץ = fint	צ = tsai	ק = qof

Table 24. HIRAGANA

ぁ, あ = a	ぃ, い = i	ぅ, う = u	ぇ, え = e
ぉ, お = o	か = ka	が = ga	き = ki
ぎ = gi	く = ku	ぐ = gu	け = ke
げ = ge	こ = ko	ご = go	さ = sa
ざ = za	し = si	じ = zi	す = su
ず = zu	せ = se	ぜ = ze	そ = so
ぞ = zo	た = ta	だ = da	ち = ti

ぢ = di	っ, つ = tu	づ = du	て = te
で = de	と = to	ど = do	な = na
に = ni	ぬ = nu	ね = ne	の = no
は = ha	ば = ba	ぱ = pa	ひ = hi
び = bi	ぴ = pi	ふ = hu	ぶ = bu
ぷ = pu	へ = he	べ = be	ぺ = pe
ほ = ho	ぼ = bo	ぽ = po	ま = ma
み = mi	む = mu	め = me	も = mo
ゃ, や = ya	ゅ, ゆ = yu	ょ, よ = yo	ら = ra
り = ri	る = ru	れ = re	ろ = ro
ゎ, わ = wa	ゐ = wi	ゑ = we	を = wo

Table 25. IPA_EXTENSIONS

Table 26. KANGXI_RADICALS

Table 27. KANNADA

Table 28. KATAKANA

ァ, ア = a	ィ, イ = i	ゥ, ウ = u	ェ, エ = e
ォ, オ = o	カ, ヵ = ka	ガ = ga	キ = ki

ギ = gi	ク = ku	グ = gu	ケ, ヶ = ke
ゲ = ge	コ = ko	ゴ = go	サ = sa
ザ = za	シ = si	ジ = zi	ス = su
ズ = zu	セ = se	ゼ = ze	ソ = so
ゾ = zo	タ = ta	ダ = da	チ = ti
ヂ = di	ッ, ツ = tu	ヅ = du	テ = te
デ = de	ト = to	ド = do	ナ = na
ニ = ni	ヌ = nu	ネ = ne	ノ = no
ハ = ha	バ = ba	パ = pa	ヒ = hi
ビ = bi	ピ = pi	フ = hu	ブ = bu
プ = pu	ヘ = he	ベ = be	ペ = pe
ホ = ho	ボ = bo	ポ = po	マ = ma
ミ = mi	ム = mu	メ = me	モ = mo
ャ, ヤ = ya	ュ, ユ = yu	ョ, ヨ = yo	ラ = ra
リ = ri	ル = ru	レ = re	ロ = ro

ㇷ, ワ = wa	ㇲ = wi	ㇱ = we	ヲ = wo
ン = n	ヴ = vu	ヷ = va	ㇳ = vi

Table 29. KAYAH_LI

Table 30. KHMER

Table 31. LAO

Table 32. LATIN

² = 2	³ = 3	¹ = 1	ß = shas
à, À, ȁ, Ȁ = ag	á, Á = aa	â, Â, ă, Ă, ấ, Ấ, ầ, Ầ, ẩ, Ẩ, ẫ, Ẫ, ậ, Ậ = ac	ã, Ã = at
ä, Ä, ā̈, Ā̈, ā̈, Ā̈, ȧ̈, Ȧ̈, ạ̈, Ạ̈ = ad	å, Å, ắ, Ắ, ą̊, Ą̊, ả̊ = ar	æ, Æ = ae	ç, Ç, ĉ, Ĉ, č, Č, ḉ, Ḉ = cc
è, È, ȅ, Ȅ = eg	é, É = ea	ê, Ê, ĕ, Ĕ, ę̂, Ę̂, ę̌, Ę̌, ğ̂, Ğ̂, ế, Ế, ề, Ề, ể, Ể, ễ, Ễ, ệ, Ệ = ec	ë, Ë, ė̈, Ė̈, ẹ̈, Ẹ̈ = ed
ì, Ì, ȉ, Ȉ = ig	í, Í = ia	î, Î, ǐ, Ǐ = ic	ï, Ï, ḯ, Ḯ, į̈, Į̈ = id
ð, Ð = eth	ñ, Ñ = nt	ò, Ò, ȍ, Ȍ = og	ó, Ó, ő, Ő = oa
ô, Ô, ǒ, Ǒ, ố, Ố, ồ, Ồ, ổ, Ổ, ỗ, Ỗ, ộ, Ộ = oc	õ, Õ, ō̃, Ō̃, ő̃, Ő̃, ö̃, Ö̃ = ot	ö, Ö, ō̈, Ō̈, ȯ̈, Ȯ̈, ō̈, Ō̈, ọ̈, Ọ̈ = od	ø, Ø, ǿ, Ǿ = os
ù, Ù, ȕ, Ȕ= ug	ú, Ú, ű, Ű = ua	û, Û, ŭ, Ŭ, u̯, U̯ = uc	ü, Ü, ṻ, Ṻ, ű̈, Ű̈, ŭ̈, Ŭ̈, ù̈, Ù̈, ü̯, Ü̯, ü̯, Ü̯ = ud
ý, Ý = ya	þ, Þ = thon	ÿ, Ÿ, ẏ, Ẏ, y̯, Y̯ = yd	ā, Ā = am

ă, Ă, ắ, Ắ, ằ, Ằ, ẳ, Ẳ, ẵ, Ẵ, ặ, Ặ = ab	ą, Ą = ao	ć, Ć = ca	ċ, Ċ = cd
ď, Ď, ḑ, Ḑ, ḍ, Ḍ = dc	đ, Đ = ds	ē, Ē, ề, Ề, é, Ế = em	ĕ, Ĕ = eb
ę, Ę = eo	ĝ, Ĝ, ġ, Ġ, ğ, Ğ = gc	ģ, Ģ = gb	ġ, Ġ = gd
ĥ, Ĥ, ȟ, Ȟ, ḫ, Ḫ = hc	ħ, Ħ = hs	ĩ, Ĩ, į, Į = it	ī, Ī = im
ĭ, Ĭ = ib	į, Į = io	ı, I = i	ĵ, Ĵ, ǰ = jc
ķ, Ķ, ǩ, Ǩ = kc	ĸ = kra	Í, Ĺ = la	ļ, Ļ, ľ, Ľ, ḷ, Ḷ = lc
l·, L· = lm	ł, Ł = ls	ń, Ń = na	ņ, Ņ, ň, Ň, ṉ, Ṉ = nc
ŉ = np	ŋ, Ŋ = eng	ō, Ō, ȍ, Ȍ, ő, Ő = om	ŏ, Ŏ = ob
ŕ, Ŕ = ra	ŗ, Ŗ, ř, Ř = rc	ś, Ś, ṡ, Ṡ = sa	ŝ, Ŝ, ş, Ş, š, Š, ṣ, Ṣ, ṡ, Ṡ = sc
ţ, Ţ, ť, Ť, ṭ, Ṭ, ṯ, Ṯ = tc	ŧ, Ŧ = ts	ũ, Ũ, ų, Ų, ű, Ű = ut	ū, Ū, ü, Ü = um
ŭ, Ŭ = ub	ů, Ů = ur	ų, Ų = uo	ŵ, Ŵ = wc
ŷ, Ŷ = yc	ź, Ź = za	ż, Ż, ẓ, Ẓ = zd	ž, Ž, Dž, DŽ, ẑ, Ẑ = zc
ſ, S = s	ƀ, Ƀ = bs	ƃ, Ƃ = bt	ƅ, Ƅ = tons
ƈ, Ƈ = ch	ƌ, Ƌ = dt	ɋ = turd	ƒ, Ƒ = fh
ƕ, Ƕ = hv	ƙ, Ƙ = kh	ɬ, Ɬ = lb	ƛ = lams
ɳ, Ɲ = nr	ơ, Ơ, ỏ, Ỏ, ớ, Ớ, ờ, Ờ, ở, Ở, ỡ, Ỡ, ợ, Ợ = oh	ɔɩ, Oɩ, ô, Ô = oi	ƥ, Ƥ = ph

ƨ, Ƨ = tont	ɻ = reve	ʯ = tp	ſ, Ƭ = th
ư, Ư, ủ, Ủ, ứ, Ứ, ừ, Ừ, ử, Ử, ữ, Ữ, ự, Ự = uh	ʏ, Ƴ, ỷ, Ỷ = yh	ƶ, Ƶ = zs	ȝ, Ȝ = ezhr
ʒ = ezht	ƨ = twos	ƽ, Ƽ = tonf	ƽ = invg
ƿ, Ƿ = wynn	ǀ = denc	ǁ = latc	ǂ = alvc
! = retc	dž, DŽ = dzc	Lj, LJ, Nj, NJ = j	lj, LJ = lj
nj, NJ = nj	ə, Ǝ = ture	ǣ, Ǣ = aem	ɡ, Ɠ = gs
ǫ, Ǫ, ǭ, Ǭ = oo	ǯ, Ǯ = ezhc	Dz, DZ = z	dz, DZ = dz
ǵ, Ǵ = ga	ƞ, Ɲ = ng	ǽ, Ǽ = aea	â, Â = ai
ê, Ê = ei	î, Î = ii	ȓ, Ȓ = rg	r̂, R̂ = ri
û, Û = ui	ȝ, Ȝ = yogh	ȣ, Ȣ = ou	ʒ, Ʒ = zh
ȳ, Ȳ = ym	ƀ, Ḃ, ḃ, Ḅ = bd	b̲, B̲ = bl	đ, Ḋ, ḋ, Ḍ = dd
d̲, D̲ = dl	ę, E̲, ẽ, Ẽ = et	ḟ, Ḟ = fd	ḡ, Ḡ = gm
ḥ, Ḥ, ḥ, Ḥ, ḧ, Ḧ = hd	h̲, H̲ = hb	ḱ, Ḱ = ka	ḳ, Ḳ = kd
k̲, K̲ = kl	l̥, L̥, l̥̄, L̥̄ = ld	l̲, L̲ = ll	ḿ, Ḿ = ma
ṁ, Ṁ, ṃ, Ṃ = md	ṅ, Ṅ, ṇ, Ṇ = nd	n̲, N̲ = nl	ṕ, Ṕ = pa
ṗ, Ṗ = pd	ṛ, Ṛ, ṛ, Ṛ, ṝ, Ṝ = rd	r̲, R̲ = rl	ś, Ś, ṣ, Ṣ, ṩ, Ṩ, ẛ = sd

ṫ, Ṫ, ṯ, Ṯ, ẗ = td	ṯ, Ṯ = tl	ṽ, Ṽ = vt	ṿ, Ṿ = vd
ẇ, Ẇ = wg	ẃ, Ẃ = wa	ẅ, Ẅ, ẇ, Ẇ, ẉ, Ẉ = wd	ẋ, Ẋ, ẍ, Ẍ = xd
ẓ, Ẓ = zl	ẖ = hl	ẘ = wr	ẙ = yr
ả, Ả = ah	ẻ, Ẻ = eh	ỉ, Ỉ = ih	ỳ, Ỳ = yg

Table 33. LETTERLIKE_SYMBOLS

℘ = p	℩ = iota	℺ = q

Table 34. MALAYALAM

Table 35. MYANMAR

Table 36. NKO

Table 37. NUMBER_FORMS

ⅰ , Ⅰ = 1	ⅱ , Ⅱ = 2	ⅲ , Ⅲ = 3	ⅳ , Ⅳ = 4
ⅴ , Ⅴ = 5	ⅵ , Ⅵ = 6	ⅶ , Ⅶ = 7	ⅷ , Ⅷ = 8
ⅸ , Ⅸ = 9	ⅹ , Ⅹ = 10	ⅺ , Ⅺ = 11	ⅻ , Ⅻ = 12
ⅼ , Ⅼ = 50	ⅽ , Ⅽ = 100	ⅾ , Ⅾ = 500	ⅿ , Ⅿ = 1000

Table 38. OGHAM

Table 39. ORIYA

Table 40. RUNIC

Table 41. SINHALA

Table 42. SUPERSCRIPTS_AND_SUBSCRIPTS

⁰, ₀ = 0	⁴, ₄ = 4	⁵, ₅ = 5	⁶, ₆ = 6
⁷, ₇ = 7	⁸, ₈ = 8	⁹, ₉ = 9	ⁿ = n