3 The Glyph Substitution Table (GSUB)

The Glyph Substitution table (GSUB) contains information for substituting glyphs to render the scripts and language systems supported in a font. Many language systems require glyph substitutes. For example, in the Arabic script, the glyph shape that depicts a particular character varies according to its position in a word or text string (see Figure 3a). In other language systems, glyph substitutes are aesthetic options for the user, such as the use of ligature glyphs in the English language (see Figure 3b).

Figure 3a. Isolated, initial, medial, and final forms of the Arabic character HAH

Figure 3b. Two Latin glyphs and their associated ligature

Overview

Many fonts use limited character encoding standards that map glyphs to characters one-to-one, assigning a glyph to each character code value in a font. Multiple character codes cannot be mapped to a single glyph, as needed for ligature glyphs, and multiple glyphs cannot be mapped to a single character code, as needed to decompose a ligature into its component glyphs.

To supply glyph substitutes, font developers must assign different character codes to the glyphs, or they must create additional fonts or character sets. To access these glyphs, users must bear the burden of switching between character codes, character sets, or fonts.

Substituting Glyphs with TrueType Open

The GSUB table in TrueType Open fully supports glyph substitution. To access glyph substitutes, GSUB maps from the glyph index or indices defined in a cmap table to the glyph index or indices of the glyph substitutes. For example, if a font has three alternative forms of an ampersand glyph, the cmap table associates the ampersand's character code with only one of these glyphs. In GSUB, the indices of the other ampersand glyphs are then referenced by this one index.

The text-processing client uses the GSUB data to manage glyph substitution actions. GSUB identifies the glyphs that are input to and output from each glyph substitution action, specifies how and where the client uses glyph substitutes, and regulates the order of glyph substitution operations. Any number of substitutions can be defined for each script or language system represented in a font.

The GSUB table supports five types of glyph substitutions that are widely used in international typography:

- A *single substitution* replaces a single glyph with another single glyph. This is used to render positional glyph variants in Arabic and vertical text in the Far East (see Figure 3c).

Figure 3c. Alternative forms of parentheses used when positioning Kanji vertically

- A *multiple substitution* replaces a single glyph with more than one glyph. This is used to specify actions such as ligature decomposition (see Figure 3d).

Figure 3d. Decomposing a Latin ligature glyph into its individual glyph components

- An *alternate substitution* identifies functionally equivalent but different looking forms of a glyph. These glyphs are often referred to as *aesthetic alternatives*. For example, a font might have five different glyphs for the ampersand symbol, but one would have a default glyph index in the cmap table. The client could use the default glyph or substitute any of the four alternatives (see Figure 3e).

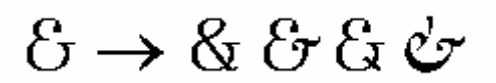

Figure 3e. Alternative ampersand glyphs in a font

- A *ligature substitution* replaces several glyph indices with a single glyph index, as when an Arabic ligature glyph replaces a string of separate glyphs (see Figure 3f) .

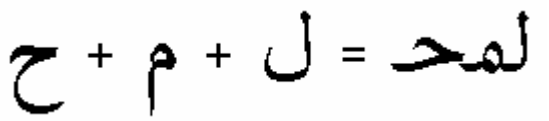

Figure 3f. Three Arabic glyphs and their associated ligature glyph

- **Contextual substitution**, the most powerful type, describes glyph substitutions in context–that is, a substitution of one or more glyphs within a certain pattern of glyphs. Each substitution describes one or more input glyph sequences and one or more substitutions to be performed on that sequence. Contextual substitutions can be applied to specific glyph sequences, glyph classes, or sets of glyphs.

Table Organization

The GSUB table begins with a header that defines offsets to a ScriptList, a FeatureList, and a LookupList (see Figure 3g):

- The ScriptList identifies all the scripts and language systems in the font that use glyph substitutes.
- The FeatureList defines all the glyph substitution features required to render these scripts and language systems.
- The LookupList contains all the lookup data needed to implement each glyph substitution feature.

For a detailed discussion of ScriptLists, FeatureLists, and LookupLists, see the chapter Common Table Formats.

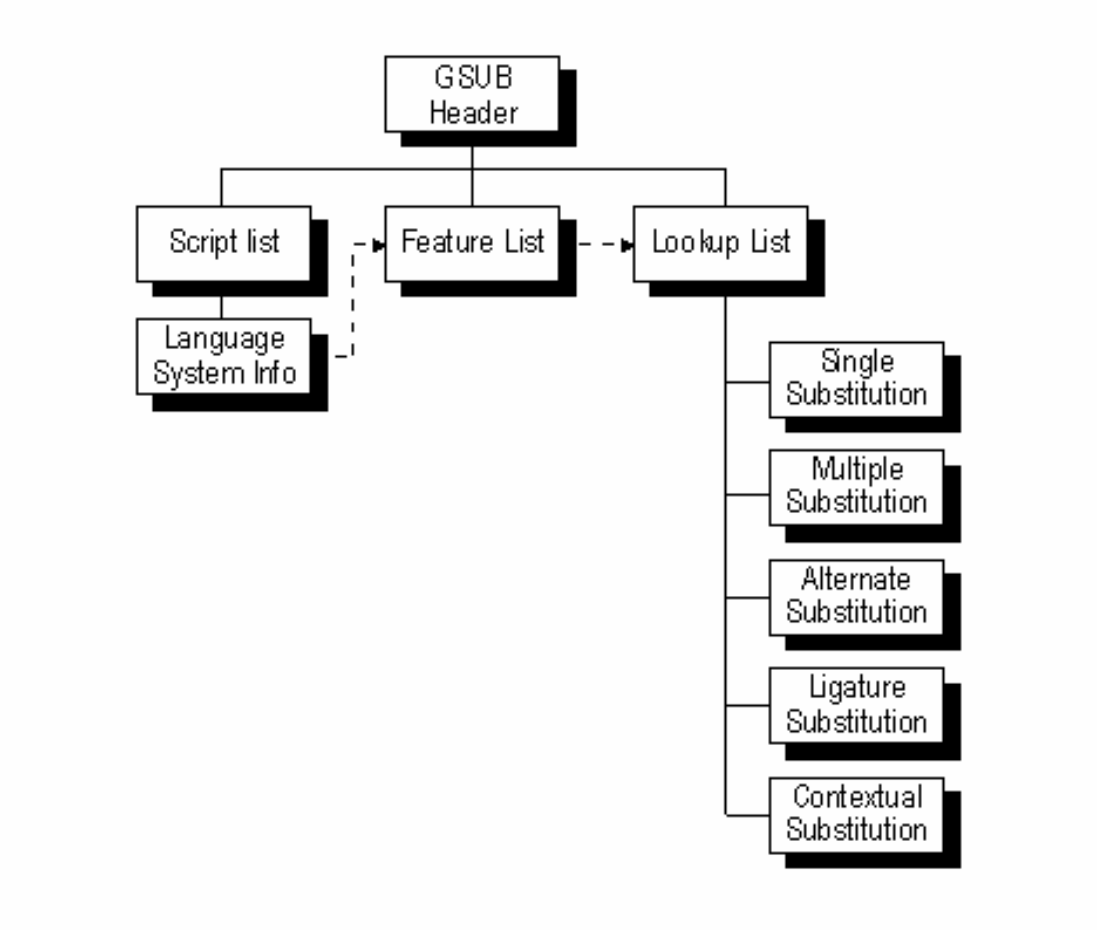

Figure 3g. High-level organization of GSUB table

This organization helps text-processing clients to easily locate the features and lookups that apply to a particular script or language system. To access GSUB information, clients should use the following procedure:

1. Locate the current script in the GSUB ScriptList table.
2. If the language system is known, search the script for the correct LangSys table; otherwise, use the script's default language system (DefaultLangSys table).
3. The LangSys table provides index numbers into the GSUB FeatureList table to access a required feature and a number of additional features.
4. Inspect the FeatureTag of each feature, and select the features to apply to an input glyph string. Each feature provides an array of index numbers into the GSUB LookupList table.
5. Assemble all lookups from the set of chosen features, and apply the lookups in the order given in the LookupList table.

Lookup data is defined in one or more subtables that define the specific conditions, type, and results of a substitution action used to implement a feature. All subtables in a lookup must be of the same LookupType, as listed in the LookupType Enumeration table:

LookupType Enumeration table for glyph substitution

Value	Type	Description
1	Single	Replace one glyph with one glyph
2	Multiple	Replace one glyph with more than one glyph
3	Alternate	Replace one glyph with one of many glyphs
4	Ligature	Replace multiple glyphs with one glyph
5	Context	Replace one or more glyphs in context
6+	Reserved	For future use

Each LookupType subtable has one or more formats. The “best” format depends on the type of substitution and the resulting storage efficiency. When glyph information is best presented in more than one format, a single lookup may define more than one subtable, as long as all the subtables are for the same LookupType. For example, within a given lookup, a glyph index array format may best represent one set of target glyphs, whereas a glyph index range format may be better for another set.

A series of substitution operations on the same glyph or string requires multiple lookups, one for each separate action. Each lookup is given a different array number in the LookupList table and is applied in the LookupList order.

During text processing, a client applies a lookup to each glyph in the string, starting with the first glyph, before moving to the next lookup. To apply a lookup, the client searches the subtables in the order they are defined in the lookup. If the first subtable does not contain data relevant to the glyph, the client searches the next subtable, and so on. If the client finds glyph data, it makes the substitution.

A lookup is finished for a glyph when the client makes a substitution or when the client finds no relevant data in any of the subtables defined in the lookup. The client then advances to the next glyph in the glyph string and begins the lookup process again.

The rest of this chapter describes the GSUB header and the subtables defined for each GSUB LookupType. Examples at the end of this chapter illustrate each of the five LookupTypes, including the three formats available for contextual substitutions.

GSUB Header

The GSUB table begins with a header that contains a version number for the table (Version) and offsets to a three tables: ScriptList, FeatureList, and LookupList. For descriptions of each of these tables, see the chapter, Common Table Formats. Example 1 at the end of this chapter shows a GSUB Header table definition.

GSUB Header

Type	Name	Description
fixed32	Version	Version of the GSUB table —initially set to 0x00010000

Offset	⇨ ScriptList	Offset to ScriptList table —from beginning of GSUB table
Offset	⇨ FeatureList	Offset to FeatureList table —from beginning of GSUB table
Offset	⇨ LookupList	Offset to LookupList table —from beginning of GSUB table

LookupType 1:
Single Substitution Subtable

Single substitution (SingleSubst) subtables tell a client to replace a single glyph with another glyph. The subtables can be either of two formats. Both formats require two distinct sets of glyph indices: one that defines input glyphs (specified in the Coverage table), and one that defines the output glyphs. Format 1 requires less space than Format 2, but it is less flexible.

Single Substitution Format 1

Format 1 calculates the indices of the output glyphs, which are not explicitly defined in the subtable. To calculate an output glyph index, Format 1 adds a constant delta value to the input glyph index. For the substitutions to occur properly, the glyph indices in the input and output ranges must be in the same order. This format does not use the Coverage Index that is returned from the Coverage table.

The SingleSubstFormat1 subtable begins with a format identifier (SubstFormat) of 1. An offset references a Coverage table that specifies the indices of the input glyphs. DeltaGlyphID is the constant value added to each input glyph index to calculate the index of the corresponding output glyph.

Example 2 at the end of this chapter uses Format 1 to replace standard numerals with lining numerals.

SingleSubstFormat1 subtable: Calculated output glyph indices

Type	Name	Description
uint16	SubstFormat	Format identifier —format = 1
Offset	⇨ Coverage	Offset to Coverage table —from beginning of Substitution table
int16	DeltaGlyphID	Add to original GlyphID to get substitute GlyphID

Single Substitution Format 2

Format 2 is more flexible than Format 1, but requires more space. It provides an array of output glyph indices (Substitute) explicitly matched to the input glyph indices specified in the Coverage table.

The SingleSubstFormat2 subtable specifies a format identifier (SubstFormat), an offset to a Coverage table that defines the input glyph indices, a count of output glyph indices in

the Substitute array (GlyphCount), and a list of the output glyph indices in the Substitute array (Substitute).
The Substitute array must contain the same number of glyph indices as the Coverage table. To locate the corresponding output glyph index in the Substitute array, this format uses the Coverage Index returned from the Coverage table.
Example 3 at the end of this chapter uses Format 2 to substitute vertically oriented glyphs for horizontally oriented glyphs.

SingleSubstFormat2 subtable: Specified output glyph indices

Type	Name	Description
uint16	SubstFormat	Format identifier —format = 2
Offset	⇨Coverage	Offset to Coverage table —from beginning of Substitution table
uint16	GlyphCount	Number of GlyphIDs in the Substitute array
GlyphID	Substitute[GlyphCount]	Array of substitute GlyphIDs —ordered by Coverage Index

LookupType 2:
Multiple Substitution Subtable

A Multiple Substitution (MultipleSubst) subtable replaces a single glyph with more than one glyph, as when multiple glyphs replace a single ligature. The subtable has a single format: MultipleSubstFormat1. The subtable specifies a format identifier (SubstFormat), an offset to a Coverage table that defines the input glyph indices, a count of offsets in the Sequence array (SequenceCount), and an array of offsets to Sequence tables that define the output glyph indices (Sequence). The Sequence table offsets are ordered by the Coverage Index of the input glyphs.
For each input glyph listed in the Coverage table, a Sequence table defines the output glyphs. Each Sequence table contains a count of the glyphs in the output glyph sequence (GlyphCount) and an array of output glyph indices (Substitute).

> **Note:** The order of the output glyph indices depends on the writing direction of the text. For text written left to right, the left-most glyph will be first glyph in the sequence. Conversely, for text written right to left, the right-most glyph will be first.

If the glyph should be deleted, the GlyphCount is set to zero, and no Substitute array is allocated.
Example 4 at the end of this chapter shows how to replace a single ligature with three glyphs.

MultipleSubstFormat1 subtable: Multiple output glyphs

Type	Name	Description
uint16	SubstFormat	Format identifier —format = 1

Offset	⇨ Coverage	Offset to Coverage table —from beginning of Substitution table
uint16	SequenceCount	Number of Sequence table offsets in the Sequence array
Offset	⇨ Sequence [SequenceCount]	Array of offsets to Sequence tables —from beginning of Substitution table —ordered by Coverage Index

Sequence table

Type	Name	Description
uint16	GlyphCount	Number of GlyphIDs in the Substitute array —to indicate glyph deletion, set to zero (0)
GlyphID	Substitute[GlyphCount]	String of GlyphIDs to substitute —ordered according to LookupFlag direction bit

LookupType 3:
Alternate Substitution Subtable

An Alternate Substitution (AlternateSubst) subtable identifies any number of aesthetic alternatives from which a user can choose a glyph variant to replace the input glyph. For example, if a font contains four variants of the ampersand symbol, the cmap table will specify the index of one of the four glyphs as the default glyph index, and an AlternateSubst subtable will list the indices of the other three glyphs as alternatives. A text-processing client would then have the option of replacing the default glyph with any of the three alternatives.

The subtable has one format: AlternateSubstFormat1. The subtable contains a format identifier (SubstFormat), an offset to a Coverage table containing the indices of glyphs with alternative forms (Coverage), a count of offsets to AlternateSet tables (AlternateSetCount), and an array of offsets to AlternateSet tables (AlternateSet).

For each glyph, an AlternateSet subtable contains a count of the alternative glyphs (GlyphCount) and an array of their glyph indices (Alternate). Because all the glyphs are functionally equivalent, they can be in any order in the array.

Example 5 at the end of this chapter shows how to replace the default ampersand glyph with alternative glyphs.

AlternateSubstFormat1 subtable: Alternative output glyphs

Type	Name	Description
uint16	SubstFormat	Format identifier —format = 1
Offset	⇨ Coverage	Offset to Coverage table —from beginning of Substitution table
uint16	AlternateSetCount	Number of AlternateSet tables
Offset	⇨ AlternateSet [AlternateSetCount]	Array of offsets to AlternateSet tables —from beginning of Substitution table —ordered by Coverage Index

AlternateSet table

Type	Name	Description
uint16	GlyphCount	Number of GlyphIDs in the Alternate array
GlyphID	Alternate[GlyphCount]	Array of alternate GlyphIDs —in arbitrary order

LookupType 4:
Ligature Substitution Subtable

A Ligature Substitution (LigatureSubst) subtable identifies ligature substitutions where a single glyph replaces multiple glyphs. One LigatureSubst subtable can specify any number of ligature substitutions.

The subtable uses a single format: LigatureSubstFormat1. It contains a format identifier (SubstFormat), a Coverage table offset (Coverage), a count of the ligature sets defined in this table (LigSetCount), and an array of offsets to LigatureSet tables (LigatureSet). The Coverage table specifies only the index of the first glyph component of each ligature set.

LigatureSubstFormat1 subtable: All ligature substitutions in a script

Type	Name	Description
uint16	SubstFormat	Format identifier —format = 1
Offset	⇨ Coverage	Offset to Coverage table —from beginning of Substitution table
uint16	LigSetCount	Number of LigatureSet tables
Offset	⇨ LigatureSet[LigSetCount]	Array of offsets to LigatureSet tables —from beginning of Substitution table —ordered by Coverage Index

A LigatureSet table, one for each covered glyph, specifies all the ligature strings that begin with the covered glyph. For example, if the Coverage table lists the glyph index for a lowercase “f,” then a LigatureSet table will define the “ffl,” “fl,” “ffi,” “fi,” and “ff” ligatures. If the Coverage table also lists the glyph index for a lowercase “e,” then a different LigatureSet table will define the “etc” ligature.

A LigatureSet table consists of a count of the ligatures that begin with the covered glyph (LigatureCount) and an array of offsets to Ligature tables, which define the glyphs in each ligature (Ligature). The order in the Ligature offset array defines the preference for using the ligatures. For example, if the “ffl” ligature is preferable to the “ff” ligature, then the Ligature array would list the offset to the “ffl” Ligature table before the offset to the “ff” Ligature table.

LigatureSet table: All ligatures beginning with the same glyph

Type	Name	Description
uint16	LigatureCount	Number of Ligature tables
Offset	⇨ Ligature[LigatureCount]	Array of offsets to Ligature tables —from beginning of LigatureSet table —ordered by preference

For each ligature in the set, a Ligature table specifies the GlyphID of the output ligature glyph (LigGlyph); a count of the total number of component glyphs in the ligature, including the first component (CompCount); and an array of GlyphIDs for the components (Component). The array starts with the second component glyph (array index = 1) in the ligature because the first component glyph is specified in the Coverage table.

> **Note:** The Component array lists GlyphIDs according to the LookupFlag bit that specifies the writing direction of the text. For text written right to left, the right-most glyph will be first. Conversely, for text written left to right, the left-most glyph will be first.

Example 6 at the end of this chapter shows how to replace a string of glyphs with a single ligature.

Ligature table: Glyph components for one ligature

Type	Name	Description
GlyphID	LigGlyph	GlyphID of ligature to substitute
uint16	CompCount	Number of components in the ligature
GlyphID	Component[CompCount - 1]	Array of component GlyphIDs —start with the second component —ordered by LookupFlag direction bit

LookupType 5:
Contextual Substitution Subtable

A Contextual Substitution (ContextSubst) subtable defines the most powerful type of glyph substitution lookup: it describes glyph substitutions in context that replace one or more glyphs within a certain pattern of glyphs.

ContextSubst subtables can be any of three formats that define a context in terms of a specific sequence of glyphs, glyph classes, or glyph sets. Each format can describe one or more input glyph sequences and one or more substitutions for each sequence.

All ContextSubst subtables specify the substitution data in a SubstLookupRecord. A description of that record follows the descriptions of the three formats available for ContextSubst subtables.

Context Substitution Format 1

Format 1 defines the context for a glyph substitution as a particular sequence of glyphs. For example, a context could be <xyz>, <holiday>, <!?*#@>, or any other glyph sequence.

Within a context sequence, Format 1 identifies particular glyph positions (not glyph indices) as the targets for specific substitutions. When a text-processing client locates a context in a string of text, it finds the lookup data for a targeted position and makes a substitution by applying the lookup data at that location.

For example, to replace the glyph string <abc> with its reverse glyph string <cba>, the input context is defined as the glyph sequence, <abc>. On locating “abc” in the text, the client searches the ContextSubstFormat1 subtable for lookup data that applies to the first glyph in the sequence. When it finds the lookup data, the client applies the lookup and replaces the glyph in the first position (in this case, the “a”) with a “c,” producing the output glyph string <cac>.

After completing the lookup for the first glyph position, the client searches the subtable for data applicable to the second glyph in the context sequence. Because the “b” does not need to be replaced, no data exists for this position.

Finally, the client searches the subtable for data to apply to the third glyph in the sequence. The subtable specifies a substitution, so the client applies the lookup and replaces the “c” glyph with an “a.” The contextual substitution ends, and the final output is the glyph string <cba>.

To specify a context, a Coverage table lists the first glyph in the sequence, and a SubRule table identifies the remaining glyphs. To describe the <abc> context used in the previous example, the Coverage table lists the glyph index of the first component of the sequence—the “a” glyph. A SubRule table defines indices for the “b” and “c” glyphs.

A single ContextSubstFormat1 subtable may define more than one context glyph sequence. If different context sequences begin with the same glyph, then the Coverage table should list the glyph only once because all glyphs in the table must be unique. For example, if three contexts each start with an “s” and two start with a “t,” then the Coverage table will list one “s” and one “t.”

For each context, a SubRule table lists all the glyphs that follow the first glyph. The table also contains an array of SubstLookupRecords that specify the substitution lookup data for each glyph position (including the first glyph position) in the context.

All of the SubRule tables defining contexts that begin with the same first glyph are grouped together and defined in a SubRuleSet table. For example, the SubRule tables that define the three contexts that begin with an “s” are grouped in one SubRuleSet table, and the SubRule tables that define the two contexts that begin with a “t” are grouped in a second SubRuleSet table. Each glyph listed in the Coverage table must have a SubRuleSet table defining all the SubRule tables that apply to a covered glyph.

To locate a context glyph sequence, the text-processing client searches the Coverage table each time it encounters a new text glyph. If the glyph is covered, the client reads the corresponding SubRuleSet table and examines each SubRule table in the set to determine whether the rest of the context matches the subsequent glyphs in the text. If the context and text string match, the client finds the target glyph positions, applies the lookups for those positions, and completes the substitutions.

A ContextSubstFormat1 subtable contains a format identifier (SubstFormat), an offset to a Coverage table (Coverage), a count of defined SubRuleSets (SubRuleSetCount), and an array of offsets to the SubRuleSet tables (SubRuleSet). As mentioned, one SubRuleSet table must be defined for each glyph listed in the Coverage table.

In the SubRuleSet array, the SubRuleSet table offsets are ordered in the Coverage Index order. The first SubRuleSet in the array applies to the first GlyphID listed in the

Coverage table, the second SubRuleSet in the array applies to the second GlyphID listed in the Coverage table, and so on.

ContextSubstFormat1 subtable: Simple context glyph substitution

Type	Name	Description
uint16	SubstFormat	Format identifier —format = 1
Offset	⇨ Coverage	Offset to Coverage table —from beginning of Substitution table
uint16	SubRuleSetCount	Number of SubRuleSet tables —must equal GlyphCount in Coverage table
Offset	⇨ SubRuleSet [SubRuleSetCount]	Array of offsets to SubRuleSet tables —from beginning of Substitution table —ordered by Coverage Index

A SubRuleSet table consists of an array of offsets to SubRule tables (SubRule), ordered by preference, and a count of the SubRule tables defined in the set (SubRuleCount). The order in the SubRule array can be critical. Consider two contexts, <abc> and <abcd>. If <abc> is first in the SubRule array, all instances of <abc> in the text—including all instances of <abcd>—will be changed. If <abcd> comes first in the array, however, only <abcd> sequences will be changed, without affecting any instances of <abc>.

SubRuleSet table: All contexts beginning with the same glyph

Type	Name	Description
uint16	SubRuleCount	Number of SubRule tables
Offset	⇨ SubRule[SubRuleCount]	Array of offsets to SubRule tables —from beginning of SubRuleSet table —ordered by preference

A SubRule table consists of a count of the glyphs to be matched in the input context sequence (GlyphCount), including the first glyph in the sequence, and an array of glyph indices that describe the context (Input). The Coverage table specifies the index of the first glyph in the context, and the Input array begins with the second glyph (array index = 1) in the context sequence.

> **Note:** The Input array lists the indices in the order the corresponding glyphs appear in the text. For text written from right to left, the right-most glyph will be first; conversely, for text written from left to right, the left-most glyph will be first.

A SubRule table also contains a count of the substitutions to be performed on the input glyph sequence (SubstCount) and an array of SubstitutionLookupRecords (SubstLookupRecord). Each record specifies a position in the input glyph sequence and a LookupListIndex to the substitution lookup that is applied at that position. The array should list records in design order, or the order the lookups should be applied to the entire glyph sequence.

SubRule table: One simple context definition

Type	Name	Description

uint16	GlyphCount	Total number of glyphs in input glyph sequence —includes the first glyph
uint16	SubstCount	Number of SubstLookupRecords
GlyphID	Input[GlyphCount - 1]	Array of input GlyphIDs —start with second glyph
struct	SubstLookupRecord [SubstCount]	Array of SubstLookupRecords —in design order

Example 7 at the end of the chapter shows how to use the ContextSubstFormat1 subtable to replace three dashes with a sequence preferred for the French language system.

Context Substitution Format 2

Format 2, a more flexible format than Format 1, describes class-based context substitution. For this format, a specific integer, called a class value, must be assigned to each glyph component in all context glyph sequences. Contexts are then defined as sequences of glyph class values. More than one context may be defined at a time.

For example, suppose that a swash capital glyph should replace each uppercase letter glyph that is preceded by a space glyph and followed by a lowercase letter glyph (a glyph sequence of space - uppercase - lowercase). The set of uppercase glyphs would constitute one glyph class (Class 1), the set of lowercase glyphs would constitute a second class (Class 2), and the space glyph would constitute a third class (Class 3). The input context might be specified with a context rule (called a SubClassRule) that describes “the set of glyph strings that form a sequence of three glyph classes, one glyph from Class 3, followed by one glyph from Class 1, followed by one glyph from Class 2.”

Each ContextSubstFormat2 subtable contains an offset to a class definition table (ClassDef), which defines the glyph class values of all input contexts. Generally, a unique ClassDef table will be declared in each instance of the ContextSubstFormat2 table that is included in a font, even though several Format 2 tables could share ClassDef tables. Class assignments are fixed (the same for each position in the context), and classes are exclusive (a glyph cannot be in more than one class at a time). The output glyphs that replace the glyphs in the context sequences do not need class values because they are specified elsewhere by GlyphID.

The ContextSubstFormat2 subtable also contains a format identifier (SubstFormat) and defines an offset to a Coverage table (Coverage). For this format, the Coverage table lists indices for the complete set of unique glyphs (not glyph classes) that may appear as the first glyph of any class-based context. In other words, the Coverage table contains the list of glyph indices for all the glyphs in all classes that may be first in any of the context class sequences. For example, if the contexts begin with a Class 1 or Class 2 glyph, then the Coverage table will list the indices of all Class 1 and Class 2 glyphs. This Coverage listing is redundant because the ClassDef table also identifies input glyphs, but it accelerates the lookup process.

A ContextSubstFormat2 subtable also defines an array of offsets to the SubClassSet tables (SubClassSet) and a count of the SubClassSet tables (SubClassSetCnt). The array contains one offset for each class (including Class 0) in the ClassDef table. In the array,

the class value defines an offset's index position, and the SubClassSet offsets are ordered by ascending class value (from 0 to SubClassSetCnt - 1).
For example, the first SubClassSet listed in the array contains all contexts beginning with Class 0 glyphs, the second SubClassSet contains all contexts beginning with Class 1 glyphs, and so on. If no contexts begin with a particular class (that is, if a SubClassSet contains no SubClassRule tables), then the offset to that particular SubClassSet in the SubClassSet array will be set to NULL.

ContextSubstFormat2 subtable: Class-based context glyph substitution

Type	Name	Description
uint16	SubstFormat	Format identifier —format = 2
Offset	⇨ Coverage	Offset to Coverage table —from beginning of Substitution table
Offset	⇨ ClassDef	Offset to glyph ClassDef table —from beginning of Substitution table
uint16	SubClassSetCnt	Number of SubClassSet tables
Offset	SubClassSet [SubClassSetCnt]	Array of offsets to SubClassSet tables —from beginning of Substitution table —ordered by class —may be NULL

Each context is defined in a SubClassRule table, and all SubClassRules that specify contexts beginning with the same class value are grouped in a SubClassSet table. Consequently, the SubClassSet containing a context identifies a context's first class component.
Each SubClassSet table consists of a count of the SubClassRule tables defined in the SubClassSet (SubClassRuleCnt) and an array of offsets to SubClassRule tables (SubClassRule). The SubClassRule tables are ordered by preference in the SubClassRule array of the SubClassSet.

SubClassSet table: All contexts beginning with the same class

Type	Name	Description
uint16	SubClassRuleCnt	Number of SubClassRule tables
Offset	⇨ SubClassRule [SubClassRuleCnt]	Array of offsets to SubClassRule tables —from beginning of SubClassSet —ordered by preference

For each context, a SubClassRule table contains a count of the glyph classes in the context sequence (GlyphCount), including the first class. A Class array lists the classes, beginning with the second class (array index = 1), that follow the first class in the context.

> **Note:** Text order depends on the writing direction of the text. For text written from right to left, the right-most class will be first. Conversely, for text written from left to right, the left-most class will be first.

The values specified in the Class array are the values defined in the ClassDef table. For example, a context consisting of the sequence “Class 2, Class 7, Class 5, Class 0” will produce a Class array of 7,5,0. The first class in the sequence, Class 2, is identified in the ContextSubstFormat2 table by the SubClassSet array index of the corresponding SubClassSet.

A SubClassRule also contains a count of the substitutions to be performed on the context (SubstCount) and an array of SubstLookupRecords (SubstLookupRecord) that supply the substitution data. For each position in the context that requires a substitution, a SubstLookupRecord specifies a LookupList index and a position in the input glyph sequence where the lookup is applied. The SubstLookupRecord array lists SubstLookupRecords in design order—that is, the order in which lookups should be applied to the entire glyph sequence.

SubClassRule table: Context definition for one class

Type	Name	Description
uint16	GlyphCount	Total number of classes specified for the context in the rule —includes the first class
uint16	SubstCount	Number of SubstLookupRecords
uint16	Class[GlyphCount - 1]	Array of classes —beginning with the second class —to be matched to the input glyph class sequence
struct	SubstLookupRecord [SubstCount]	Array of Substitution lookups —in design order

Example 8 at the end of this chapter uses Format 2 to substitute Arabic mark glyphs for base glyphs of different heights.

Context Substitution Format 3

Format 3, coverage-based context substitution, defines a context rule as a sequence of coverage tables. Each position in the sequence may define a different Coverage table for the set of glyphs that matches the context pattern. With Format 3, the glyph sets defined in the different Coverage tables may intersect, unlike Format 2 which specifies fixed class assignments (identical for each position in the context sequence) and exclusive classes (a glyph cannot be in more than one class at a time).

For example, consider an input context that contains a lowercase glyph (position 0), followed by an uppercase glyph (position 1), either a lowercase or numeral glyph (position 2), and then either a lowercase or uppercase vowel (position 3). This context requires four Coverage tables, one for each position:

- In position 0, the Coverage table lists the set of lowercase glyphs.
- In position 1, the Coverage table lists the set of uppercase glyphs.
- In position 2, the Coverage table lists the set of lowercase and numeral glyphs, a superset of the glyphs defined in the Coverage table for position 0.

- In position 3, the Coverage table lists the set of lowercase and uppercase vowels, a subset of the glyphs defined in the Coverage tables for both positions 0 and 1.

Unlike Formats 1 and 2, this format defines only one context rule at a time. It consists of a format identifier (SubstFormat), a count of the glyphs in the sequence to be matched (GlyphCount), and an array of Coverage offsets that describe the input context sequence (Coverage).

> **Note:** The order of the Coverage tables listed in the Coverage array must follow the writing direction. For text written from right to left, then the right-most glyph will be first. Conversely, for text written from left to right, the left-most glyph will be first.

The subtable also contains a count of the substitutions to be performed on the input Coverage sequence (SubstCount) and an array of SubstLookupRecords (SubstLookupRecord) in design order—that is, the order in which lookups should be applied to the entire glyph sequence. (SubstLookupRecords are described next.) Example 9 at the end of this chapter substitutes swash glyphs for two out of three glyphs in a sequence.

ContextSubstFormat3 subtable: Coverage-based context glyph substitution

Type	Name	Description
uint16	SubstFormat	Format identifier —format = 3
uint16	GlyphCount	Number of glyphs in the input glyph sequence
uint16	SubstCount	Number of SubstLookupRecords
Offset	⇨ Coverage[GlyphCount]	Array of offsets to Coverage table —from beginning of Substitution table —in glyph sequence order
struct	SubstLookupRecord [SubstCount]	Array of SubstLookupRecords —in design order

Substitution Lookup Record

All contextual substitution subtables specify the substitution data in a Substitution Lookup Record (SubstLookupRecord). Each record contains a SequenceIndex, which indicates the position where the substitution will occur in the glyph sequence. In addition, a LookupListIndex identifies the lookup to be applied at the glyph position specified by the SequenceIndex.

The contextual substitution subtables defined in Examples 7, 8, and 9 at the end of this chapter show SubstLookupRecords.

SubstLookupRecord

Type	Name	Description
uint16	SequenceIndex	Index into current glyph sequence —first glyph = 0
uint16	LookupListIndex	Lookup to apply to that position

—zero-based

The SequenceIndex in a SubstLookupRecord must take into consideration the order in which lookups are applied to the entire glyph sequence. Because multiple substitutions may occur per context, the SequenceIndex and LookupListIndex refer to the glyph sequence *after* the text-processing client has applied any previous lookups. In other words, the SequenceIndex identifies the location for the substitution at the time that the lookup is to be applied.

For example, consider an input glyph sequence of four glyphs. The first glyph does not have a substitute, but the middle two glyphs will be replaced with a ligature, and a single glyph will replace the fourth glyph:

- The first glyph is in position 0. No lookups will be applied at position 0, so no SubstLookupRecord is defined.
- The SubstLookupRecord defined for the ligature substitution specifies the SequenceIndex as position 1, which is the position of the first-glyph component in the ligature string. After the ligature replaces the glyphs in positions 1 and 2, however, the input glyph sequence consists of only three glyphs, not the original four.
- To replace the last glyph in the sequence, the SubstLookupRecord defines the SequenceIndex as position 2 instead of position 3. This position reflects the effect of the ligature substitution applied before this single substitution.

  Note: This example assumes that the LookupList specifies the ligature substitution lookup *before* the single substitution lookup.

GSUB Subtable Examples

The rest of this chapter describes and illustrates examples of all the GSUB subtables, including each of the three formats available for contextual substitutions. All the examples reflect unique parameters described below, but the samples provide a useful reference for building subtables specific to other situations.

All the examples have three columns showing hex data, source, and comments.

Example 1: GSUB Header Table

Example 1 shows a typical GSUB Header table definition.

Example 1

Hex Data	Source	Comments
	GSUBHeader	
	TheGSUBHeader	;GSUBHeader table definition
00010000	0x00010000	;version
000A	TheScriptList	;offset to ScriptList table

```
001E                                    TheFeatureList      ;offset to
                                        FeatureList table
002C                                    TheLookupList;offset to
                                        LookupList table
```

Example 2: SingleSubstFormat1 Subtable

Example 2 illustrates the SingleSubstFormat1 subtable, which uses ranges to replace single input glyphs with their corresponding output glyphs. The indices of the output glyphs are calculated by adding a constant delta value to the indices of the input glyphs. In this example, the Coverage table has a format identifier of 1 to indicate the range format, which is used because the input glyph indices are in consecutive order in the font. The Coverage table specifies one range that contains a StartGlyphID for the "0" (zero) glyph and an EndGlyphID for the "9" glyph.

Example 2

Hex Data	Source Comments
	SingleSubstFormat1
	LiningNumeralSubtable ;SingleSubst subtable definition
0001	1 ;SubstFormat, ranges
0006	LiningNumeralCoverage ;offset to Coverage table
	; for input glyphs
00C0	192 ;DeltaGlyphID = 192
	; add to each input glyph index
	; to produce output glyph
	; index
	CoverageFormat2
	LiningNumeralCoverage ;Coverage table definition
0002	2 ;CoverageFormat, ranges
	1 ;RangeCount
	;RangeRecord[0]
004E	78 ;Start GlyphID
	; for numeral zero glyph
0058	87 ;End GlyphID
	; for numeral nine glyph
0000	0 ;StartCoverageIndex
	; first CoverageIndex = 0

Example 3: SingleSubstFormat2 Subtable

Example 3 uses the SingleSubstFormat2 subtable for lists to substitute punctuation glyphs in Japanese text that is written vertically. Horizontally oriented parentheses and square brackets (the input glyphs) are replaced with vertically oriented parentheses and square brackets (the output glyphs).

The Coverage table, Format 1, identifies each input glyph index. The number of input glyph indices listed in the Coverage table matches the number of output glyph indices listed in the subtable. For correct substitution, the order of the glyph indices in the Coverage table (input glyphs) must match the order in the Substitute array (output glyphs).

Example 3

Hex Data	Source Comments
	SingleSubstFormat2
	VerticalPunctuationSubtable ;SingleSubst subtable definition
0002	2 ;SubstFormat lists
000E	VerticalPunctuationCoverage ;offset to Coverage table
0004	4 ;GlyphCount ; equals GlyphCount in ; Coverage table
0131	VerticalOpenBracketGlyph ;Substitute[0] ; ordered by Coverage Index
0135	VerticalClosedBracketGlyph ;Substitute[1]
013E	VerticalOpenParenthesisGlyph ;Substitute[2]
0143	VerticalClosedParenthesisGlyph ;Substitute[3]
	CoverageFormat1
	VerticalPunctuationCoverage ;Coverage table definition
0001	1 ;CoverageFormat ; lists
0004	4 ;GlyphCount
003C	HorizontalOpenBracketGlyph ;GlyphArray[0] ; ordered by GlyphID
0040	HorizontalClosedBracketGlyph ;GlyphArray[1]
004B	HorizontalOpenParenthesisGlyph ;GlyphArray[2]
004F	HorizontalClosedParenthesisGlyph ;GlyphArray[3]

Example 4: MultipleSubstFormat1 Subtable

Example 4 uses a MultipleSubstFormat1 subtable to replace a single “ffi” ligature with three individual glyphs that form the string <ffi>. The subtable defines a format identifier of 1, an offset to a Coverage table that specifies the glyph index of the “ffi” ligature (the input glyph), an offset to a Sequence table that specifies the sequence of glyph indices for the <ffi> string in its substitute array (the output glyph sequence), and a count of Sequence table offsets.

Example 4

Hex Data	Source Comments
	MultipleSubstFormat1
	FfiDecompSubtable ;MultipleSubst subtable ; definition
0001	1 ;SubstFormat
0008	FfiDecompCoverage ;offset to Coverage table
0001	1 ;SequenceCount ; equals GlyphCount in ; Coverage table
000E	FfiDecompSequence ;offset to Sequence[0] table
	CoverageFormat1
	FfiDecompCoverage ;Coverage table definition
0001	1 ;CoverageFormat ; lists
0001	1 ;GlyphCount
00F1	ffiGlyphID ;ligature glyph
	Sequence
	FfiDecompSequence ;Sequence table definition
0003	3 ;GlyphCount
001A	fGlyphID ;first glyph in sequence order
001A	fGlyphID ;second glyph
001D	iGlyphID ;third glyph

Example 5: AlternateSubstFormat 1 Subtable

Example 5 uses the AlternateSubstFormat1 subtable to replace the default ampersand glyph (input glyph) with one of two alternative ampersand glyphs (output glyph).

In this case, the Coverage table specifies the index of a single glyph, the default ampersand, because it is the only glyph covered by this lookup. The AlternateSet table for this covered glyph identifies the alternative glyphs: AltAmpersand1GlyphID and AltAmpersand2GlyphID.

In Example 5, the index position of the AlternateSet table offset in the AlternateSet array is zero (0), which correlates with the index position (also zero) of the default ampersand glyph in the Coverage table.

Example 5

Hex Data	Source Comments
	AlternateSubstFormat1
	AltAmpersandSubtable ;AlternateSubstFormat1 subtable ;definition
0001	1 ;SubstFormat

```
0008                                    AltAmpersandCoverage;offset to
                                        Coverage table
0001                                    1      ;AlternateSetCnt
                                               ;  equals GlyphCount in
                                               ;  Coverage table
000E                                    AltAmpersandSet     ;offset to
                                        AlternateSet[0]
                                               ;  table

                                        CoverageFormat1
                                        AltAmpersandCoverage;Coverage
                                        table definition
0001                                    1      ;CoverageFormat
0001                                    1      ;GlyphCount
003A                                    DefaultAmpersandGlyphID
                                               ;GlyphArray[0]

                                        AlternateSet
                                        AltAmpersandSet     ;AlternateSet
                                        table definition
0002                                    2      ;GlyphCount
00C9                                    AltAmpersand1GlyphID;offset to
                                        Alternate[0]
                                               ;  in arbitrary order
00CA                                    AltAmpersand2GlyphID;offset to
                                        Alternate[1]
```

Example 6: LigatureSubstFormat1 Subtable

Example 6 shows a LigatureSubstFormat1 subtable that defines data to replace a string of glyphs with a single ligature glyph. Because a LigatureSubstFormat1 subtable can specify glyph substitutions for more than one ligature, this subtable defines three ligatures: “etc,” “ffi,” and “fi.”

The sample subtable contains a format identifier (4) and an offset to a Coverage table. The Coverage table, which lists an index for each first glyph in the ligatures, lists indices for the “e” and “f” glyphs. The Coverage table range format is used here because the “e” and “f” glyph indices are numbered consecutively.

In the LigatureSubst subtable, LigSetCount specifies two LigatureSet tables, one for each covered glyph, and the LigatureSet array stores offsets to them. In this array, the “e” LigatureSet precedes the “f” LigatureSet, matching the order of the corresponding first-glyph components in the Coverage table.

Each LigatureSet table identifies all ligatures that begin with a covered glyph. The sample LigatureSet table defined for the “e” glyph contains only one ligature, “etc.” A LigatureSet table defined for the “f” glyph contains two ligatures, “ffi” and “fi.”

The sample FLigaturesSet table has offsets to two Ligature tables, one for “ffi” and one for “fi.” The Ligature array lists the “ffi” Ligature table first to indicate that the “ffi” ligature is preferred to the “fi” ligature.

Example 6

Hex Data	Source Comments
	LigatureSubstFormat1
	LigaturesSubtable ;LigatureSubstFormat1 subtable
	; definition
0001	1 ;SubstFormat
000A	LigaturesCoverage ;offset to Coverage table
0002	2 ;LigSetCount
0014	ELigaturesSet ;offset to LigatureSet[0] table
	; in Coverage Index order
0020	FLigaturesSet ;offset to LigatureSet[1] table
	CoverageFormat2
	LigaturesCoverage ;Coverage table definition
0002	2 ;CoverageFormat, ranges
0001	1 ;RangeCount
	;RangeRecord[0]
0019	eGlyphID ;Start, first GlyphID
001A	fGlyphID ;End, last GlyphID in range
0000	0 ;StartCoverageIndex
	; coverage index of start
	; glyphID
	LigatureSet
	ELigaturesSet ;LigatureSet table definition
	; all ligatures that start
	; with e
0001	1 ;LigatureCount
0004	etcLigature ;offset to Ligature[0] table
	Ligature
	etcLigature ;Ligature table definition
015B	etcGlyphID ;LigGlyph
	;output GlyphID
0003	3 ;CompCount
	; number of components
0028	tGlyphID ;Component[1]
	; second component in ligature
0017	cGlyphID ;Component[2
	; third component in ligature
	LigatureSet

```
                                    FLigaturesSet;LigatureSet table
                                    definition
                                          ;  all ligatures start
                                    with f
0002                                2     ;LigatureCount
0006                                ffiLigature  ;offset to
                                    Ligature[0] table
                                          ;  listed first because
                                    ffi
                                          ;    ligature is preferred
                                    to
                                          ;    fi ligature
000E                                fiLigature   ;offset to
                                    Ligature[1] table

                                    Ligature
                                    ffiLigature  ;Ligature table
                                    definition
00F1                                ffiGlyphID   ;LigGlyph, output
                                    GlyphID
0003                                3     ;CompCount
001A                                fGlyphID     ;Component[1]
                                          ;  second component in
                                    ligature
001D                                iGlyphID     ;Component[2]
                                          ;  third component in
                                    ligature

                                    Ligature
                                    fiLigature   ;Ligature table
                                    definition
00F0                                fiGlyphID    ;LigGlyph, output
                                    GlyphID
0002                                2     ;CompCount
001D                                iGlyphID     ;Component[1]
                                          ;  second component in
                                    ligature
```

Example 7: ContextSubstFormat1 Subtable and SubstLookupRecord

Example 7 uses a ContextSubstFormat1 subtable for glyph sequences to replace a string of three glyphs with another string. For the French language system, the subtable defines a contextual substitution that replaces the input sequence, space-dash-space, with the output sequence, thin space-dash-thin space.

The contextual substitution, called Dash Lookup in this example, contains one ContextSubstFormat1 subtable called the DashSubtable. The subtable specifies two contexts: a SpaceGlyph followed by a DashGlyph, and a DashGlyph followed by a SpaceGlyph. In each sequence, a single substitution replaces the SpaceGlyph with a ThinSpaceGlyph.

The Coverage table, labeled DashCoverage, lists two GlyphIDs for the first glyphs in the SpaceGlyph and DashGlyph sequences. One SubRuleSet table is defined for each covered glyph.

SpaceAndDashSubRuleSet lists all the contexts that begin with a SpaceGlyph. It contains an offset to one SubRule table (SpaceAndDashSubRule), which specifies two glyphs in

the context sequence, the second of which is a DashGlyph. The SubRule table contains an offset to a SubstLookupRecord that lists the position in the sequence where the glyph substitution should occur (position 0) and the index of the SpaceToThinSpaceLookup applied there to replace the SpaceGlyph with a ThinSpaceGlyph.
DashAndSpaceSubRuleSet lists all the contexts that begin with a DashGlyph. An offset to a SubRule table (DashAndSpaceSubRule) specifies two glyphs in the context sequence, and the second one is a SpaceGlyph. The SubRule table contains an offset to a SubstLookupRecord, which lists the position in the sequence where the glyph substitution should occur, and an index to the same lookup used in the SpaceAndDashSubRule. The lookup replaces the SpaceGlyph with a ThinSpaceGlyph.

Example 7

Hex Data	Source Comments
	ContextSubstFormat1
	DashSubtable ;ContextSubstFormat1 subtable
	; definition
	; for Lookup[0], DashLookup
0001	1 ;SubstFormat
000A	DashCoverage ;offset to Coverage table
0002	2 ;SubRuleSetCount
0012	SpaceAndDashSubRuleSet
	;offset to SubRuleSet[0]
	; ordered by Coverage Index
0020	DashAndSpaceSubRuleSet
	;offset to SubRuleSet[1]
	CoverageFormat1
	DashCoverage ;Coverage table definition
0001	1 ;CoverageFormat
	; lists
0002	2 ;GlyphCount
0028	SpaceGlyph ;GlyphArray[0]
	; in numeric order
005D	DashGlyph ;GlyphArray[1]
	;dash GlyphID
	SubRuleSet
	SpaceAndDashSubRuleSet
	;SubRuleSet[0] table definition
0001	1 ;SubRuleCount
0004	SpaceAndDashSubRule ;offset to SubRule[0]
	; ordered by preference
	SubRule
	SpaceAndDashSubRule ;SubRule[0] table definition
0002	2 ;GlyphCount

```
                                        ;number in input sequence
0001                                1   ;SubstCount
005D                                DashGlyph    ;Input[1]
                                        ;   starting with second
                                    glyph
                                        ;  SpaceGlyph in Coverage
                                    table
                                        ;   is first glyph
                                        ;SubstLookupRecord[0]
0000                                0   ;SequenceIndex
                                        ;  substitution at first
                                    glyph
                                        ;    position (0)
0001                                1   ;LookupListIndex
                                        ;  index for
                                        ;
                                    SpaceToThinSpaceLookup in
                                        ;    LookupList

                                    SubRuleSet
                                    DashAndSpaceSubRuleSet
                                        ;SubRuleSet[0] table
                                    definition
0001                                1   ;SubRuleCount
0004                                DashAndSpaceSubRule ;offset to
                                    SubRule[0]
                                        ;  ordered by preference

                                    SubRule
                                    DashAndSpaceSubRule ;SubRule[0]
                                    table definition
0002                                2   ;GlyphCount
                                        ;  number in the input
                                    glyph
                                        ;    sequence
0001                                1   ;SubstCount
0028                                SpaceGlyph   ;Input[1]
                                        ;  starting with second
                                    glyph
                                        ;SubstLookupRecord
                                    definition
0001                                1   ;SequenceIndex
                                        ;substitution at second
                                    glyph
                                        ;  position(1)
0001                                1   ;LookupListIndex
                                        ;  for
                                    SpaceToThinSpaceLookup
```

Example 8: ContextSubstFormat2 Subtable

Example 8 uses a ContextSubstFormat2 subtable with glyph classes to replace default mark glyphs with their alternative forms. Glyph alternatives are selected depending upon the height of the base glyph that they combine with—that is, the mark glyph used above a high base glyph differs from the mark glyph above a very high base glyph.

In the example, SetMarksHighSubtable contains a Class table that defines four glyph classes: medium-height glyphs (Class 0), all default mark glyphs (Class 1), high glyphs

(Class 2), and very high glyphs (Class 3). The subtable also contains a Coverage table that lists each base glyph that functions as a first component in a context, ordered by glyph index.

Two SubClassSets are defined, one for substituting high marks and one for very high marks. No SubClassSets are specified for Class 0 and Class 1 glyphs because no contexts begin with glyphs from these classes. The SubClassSet array lists SubClassSets in numerical order, so SubClassSet 2 precedes SubClassSet 3.

Within each SubClassSet, a SubClassRule is defined. In SetMarksHighSubClassSet2, the SubClassRule table specifies two glyphs in the context, the first glyph in Class 2 (a high glyph) and the second in Class 1 (a mark glyph). The SubstLookupRecord specifies applying SubstituteHighMarkLookup at the second position in the sequence—that is, a high mark glyph will replace the default mark glyph.

In SetMarksVeryHighSubClassSet3, the SubClassRule specifies two glyphs in the context, the first in Class 3 (a very high glyph) and the second in Class 1 (a mark glyph). The SubstLookupRecord specifies applying SubstituteVeryHighMarkLookup at the second position in the sequence—that is, a very high mark glyph will replace the default mark glyph.

Example 8

Hex Data	Source Comments
	ContextSubstFormat2
	SetMarksHighSubtable
	;ContextSubstFormat2 subtable
	; definition
0002	2 ;SubstFormat
0010	SetMarksHighCoverage;offset to Coverage table
001C	SetMarksHighClassDef;offset to Class Def table
0004	4 ;SubClassSetCnt
0000	NULL ;offset to SubClassSet[0] table
	; no contexts that begin with
	; Class 0 glyphs are defined
0000	NULL ;offset to SubClassSet[1] table
	; no contexts that begin with
	; Class 1 glyphs are defined
0032	SetMarksHighSubClassSet2
	;offset to SubClassSet[2] table
	; for contexts that begin with
	; Class 2 glyphs (high base
	; glyphs)

```
0040                    SetMarksVeryHighSubClassSet3
                              ;offset to SubClassSet[3]
                        table
                              ;  for contexts that begin
                        with
                              ;    Class 3 glyphs (very
                        high
                              ;    base glyphs)

                        CoverageFormat1
                        SetMarksHighCoverage;Coverage
                        table definition
0001                    1     ;CoverageFormat, lists
0004                    4     ;GlyphCount
0030                    tahGlyphID   ;GlyphArray[0], high
                        base glyph
0031                    dhahGlyphID  ;GlyphArray[1], high
                        base glyph
0040                    cafGlyphID   ;GlyphArray[2], very
                        high
                        ;  base glyph
0041                    gafGlyphID   ;GlyphArray[3], very
                        high
                        ;  base glyph

                        ClassDefFormat2
                        SetMarksHighClassDef;Class table
                        definition
0002                    2     ;Class Format, ranges
0003                    3     ;ClassRangeCount
                              ;ClassRange[0]
                              ;  ordered by StartGlyphID
                              ;for Class 2, high base
                        glyphs
0030                    tahGlyphID   ;Start
                              ;  first Glyph ID in range
0031                    dhahGlyphID  ;End
                              ;  last Glyph ID in range
0002                    2     ;Class
                              ;ClassRange[1]
                              ;  for Class 3, very high
                              ;    base glyphs
0040                    cafGlyphID   ;Start
                              ;  first Glyph ID in the
                        range
0041                    gafGlyphID   ;End
                              ;  last Glyph ID in the
                        range
0003                    3     ;Class
                              ;ClassRange[2]
                              ;  for Class 1, mark gyphs
00D2                    fathatanDefaultGlyphID    ;Start
                              ;  first Glyph ID in range
                              ;    default fathatan mark
00D3                    dammatanDefaultGlyphID    ;End
                              ;  last Glyph ID in the
                        range
                              ;    default dammatan mark
0001                    1     ;Class
```

```
                                    SubClassSet
                                    SetMarksHighSubClassSet2
                                          ;SubClassSet[2] table
definition
                                          ;  all contexts that begin
with
                                          ;Class 2 glyphs
0001                                1     ;SubClassRuleCnt
0004                                SetMarksHighSubClassRule2
                                          ;offset to SubClassRule[0]
table
                                          ;  ordered by preference

                                    SubClassRule
                                    SetMarksHighSubClassRule2
                                          ;SubClassRule[0] table
                                          ;  definition
                                          ;Class 2 glyph (high base)
                                          ;  glyph followed by a
Class 1
                                          ;  glyph (mark)
0002                                2     ;GlyphCount
0001                                1     ;SubstCount
0001                                1     ;offset to Class[1]
                                          ;  beginning with the
second
                                          ;    Class in the context
                                          ;    sequence (mark =
Class 1)
                                          ;begin SubstLookupRecord
array
                                          ;  in design order
                                          ;SubstLookupRecord[0]
0001                                1     ;SequenceIndex
                                          ;  apply substitution to
                                          ;    position 2, a mark
0001                                1     ;LookupListIndex

                                    SubClassSet
                                    SetMarksVeryHighSubClassSet3
                                          ;SubClassSet[3] table
definition
                                          ;  all contexts that begin
with
                                          ;    Class 3 glyphs
0001                                1     ;SubClassRuleCnt
0004                                SetMarksVeryHighSubClassRule3
                                          ;offset to SubClassRule[0]
table
                                          ;  ordered by preference

                                    SubClassRule
                                    SetMarksVeryHighSubClassRule3
                                          ;SubClassRule[0] table
;  definition
                                          ;Class 3 glyph (very high
base
                                          ;  glyph) followed by a
Class 1
                                          ;    glyph (mark)
```

```
0002                                    2       ;GlyphCount
0001                                    1       ;SubstCount
0001                                    1       ;offset to Class[1]
                                                ;  beginning with the
                                        second
                                                ;    Class in the context
                                                ;    sequence = marks,
                                        Class 1
                                                ;begin SubstLookupRecord
                                        array
                                                ;  in design order
                                                ;SubstLookupRecord[0]
0001                                    1       ;SequenceIndex
                                                ;apply substitution to
                                        position
                                                ;  2, second glyph class
                                        (mark)
0002                                    2       ;LookupListIndex
```

Example 9: ContextualSubstFormat3 Subtable

Example 9 uses the ContextSubstFormat3 subtable with Coverage tables to describe a context sequence of three lowercase glyphs in the pattern: any ascender or descender glyph in position 0 (zero), any x-height glyph in position 1, and any descender glyph in position 2. The overlapping sets of covered glyphs for positions 0 and 2 make Format 3 better for this context than the class-based Format 2.

In positions 0 and 2, swash versions of the glyphs replace the default glyphs. The contextual-substitution lookup is SwashLookup (LookupList index = 0), and its subtable is SwashSubtable. The SwashSubtable defines three Coverage tables: AscenderDescenderCoverage, XheightCoverage, and DescenderCoverage—one for each glyph position in the context sequence, respectively.

The SwashSubtable also defines two SubstLookupRecords: one that applies to position 0, and one for position 2. (No substitutions are applied to position 1.) The record for position 0 uses a single substitution lookup called AscDescSwashLookup to replace the current ascender or descender glyph with a swash ascender or descender glyph. The record for position 2 uses a single substitution lookup called DescSwashLookup to replace the current descender glyph with a swash descender glyph.

Example 9

Hex Data	Source Comments
	ContextSubstFormat3
	SwashSubtable;ContextSubstFormat3 subtable
	; definition
0003	3 ;SubstFormat
0003	3 ;GlyphCount
	; in input glyph sequence
0002	2 ;SubstCount
0030	AscenderDescenderCoverage
	;offset to Coverage[0] table

```
                                        ;  in context sequence
                                    order
004C                                XheightCoverage     ;offset to
                                    Coverage[1] table
006E                                DescenderCoverage   ;offset to
                                    Coverage[2] table
                                        ;SubstLookupRecord[0]
                                        ;  in glyph position order
0000                                0       ;SequenceIndex
0001                                1       ;LookupListIndex
                                        ;single substitution to
                                    output
                                        ;  ascender or descender
                                    swash
                                        ;SubstLookupRecord[1]
0002                                2       ;SequenceIndex
0002                                2       ;LookupListIndex
                                        ;  single substitution to
                                    output
                                        ;    descender swash

                                    CoverageFormat1
                                    AscenderDescenderCoverage
                                        ;Coverage table definition
0001                                1       ;CoverageFormat, lists
000C                                12      ;GlyphCount
0033                                bGlyphID      ;GlyphArray[0]
                                        ;  in GlyphID order
0035                                dGlyphID      ;GlyphArray[1]
0037                                fGlyphID      ;GlyphArray[2]
0038                                gGlyphID      ;GlyphArray[3]
0039                                hGlyphID      ;GlyphArray[4]
003B                                jGlyphID      ;GlyphArray[5]
003C                                kGlyphID      ;GlyphArray[6]
003D                                lGlyphID      ;GlyphArray[7]
0041                                pGlyphID      ;GlyphArray[8]
0042                                qGlyphID      ;GlyphArray[9]
0045                                tGlyphID      ;GlyphArray[10]
004A                                yGlyphID      ;GlyphArray[11]

                                    CoverageFormat1
                                    XheightCoverage     ;Coverage
                                    table definition
0001                                1       ;CoverageFormat, lists
000F                                15      ;GlyphCount
0032                                aGlyphID      ;GlyphArray[0]
                                        ;  in GlyphID order
0034                                cGlyphID      ;GlyphArray[1]
0036                                eGlyphID      ;GlyphArray[2]
003A                                iGlyphID      ;GlyphArray[3]
003E                                mGlyphID      ;GlyphArray[4]
003F                                nGlyphID      ;GlyphArray[5]
0040                                oGlyphID      ;GlyphArray[6]
0043                                rGlyphID      ;GlyphArray[7]
0044                                sGlyphID      ;GlyphArray[8]
0045                                tGlyphID      ;GlyphArray[9]
0046                                uGlyphID      ;GlyphArray[10]
0047                                vGlyphID      ;GlyphArray[11]
0048                                wGlyphID      ;GlyphArray[12]
```

```
0049                                    xGlyphID     ;GlyphArray[13]
004B                                    zGlyphID     ;GlyphArray[14]

                                        CoverageFormat1
                                        DescenderCoverage   ;Coverage
                                        table definition
0001                                    1      ;CoverageFormat, lists
0005                                    5      ;GlyphCount
0038                                    gGlyphID     ;GlyphArray[0]
                                               ;  in GlyphID order
003B                                    jGlyphID     ;GlyphArray[1]
0041                                    pGlyphID     ;GlyphArray[2]
0042                                    qGlyphID     ;GlyphArray[3]
004A                                    yGlyphID     ;GlyphArray[4]
```