Typing diacritics in Junicode 2

Junicode has better support for diacritics than any other free medieval font. You can place a diacritic (or "combining mark") on just about any character where it makes any sense at all to place it. Just type the base character followed by the diacritic.

But typing a diacritic can be a problem. (I'm not talking about the common letter+diacritic combinations like **á**, for which your system no doubt provides special keyboard sequences, but about exotic combinations like **þ̓** or **m̋**. Stuff that comes up with distressing regularity when you're transcribing medieval manuscripts, whose scribes weren't constrained by the limitations of mechanical typesetting.) Diacritics—the exotic ones, at least—have no assigned keys on a keyboard. Instead you need to know their *encodings*: the four-digit hexadecimal (base-16) number assigned to them in the Unicode standard.

If what I've just written looks like gibberish, don't worry: Junicode 2 offers you a way around the pain of having to learn dozens of codes like "1DE1." Specifically, it provides a collection of entities for typing in diacritics: these are enabled by Stylistic Set 10 (**ss10**), which should be applied to the entire document (in a word processor document do this via a style—"Normal" in MS Word or "Default Paragraph Style" in LibreOffice). In some cases, these entities are conveniences, while in other cases, using them will help you avoid a serious technical difficulty (as explained below).

The entities are all preceded by an ampersand (**&**) and followed by a semicolon (**;**), and they consist of the underscore character followed by one or more letters. In the simplest case, the sequence **&_a;** produces a combining **a**, which can go above any letter, e.g. **m̊**. Some diacritics are like small capitals, and you type those in the same way, but with "sc" after the letter. For example, the sequence **t&_dsc;** gives you **t̤**—a **t** with a small cap **ᴅ** above. Some are ligatures, and you type both elements of those, e.g. **m&_an**; = **m̄**. For some ligatures, the second element is a small cap, so **Z&_ansc;** = **Z̄**.

There are also entities for diacritics that are not alphabetic characters. For example, **&_zz;** gives you a zigzag, that maddeningly ambiguous abbreviation for **er**, **is**, **es** and dozens of other things: **æ̑**.

There's not much more to it, except for a few important caveats. First, these codes affect the appearance of your document but not the underlying text. When you type the sequence **p&_oa;**, it looks like **p** with an open **a** on your screen (**p̋**), but underneath, the text is still

p&_oa;. So if you send your text file to a friend who doesn't have Junicode 2 installed, that friend will see the entities, not the diacritics; if you copy your text onto the clipboard and paste it into an application that either can't access Junicode 2 or can't apply OpenType features, you'll also see the entities.

And so you may prefer typing those incomprehensible Unicode values to typing entities, at least for the Unicode-standard diacritics. Indeed, if you are sending your text to a publisher, you really ought to do so, or perhaps perform a last-minute search-and-replace to replace the entities with their Unicode equivalents (turn off **ss10** to make them visible).

The second caveat is that, while most of the diacritics in Junicode 2 belong to the Unicode standard, and any competent application will render them correctly, positioning them precisely over their base characters, many diacritics *do not* belong to Unicode, and some applications will not position them correctly when entered via their code points: they may come out looking like **m̊**, or worse yet **Ấ**. (You can tell these non-Unicode diacritics by the codes that start with "E" or "F," e.g. "F03B"—their entries are in green in the table below.)

☞ But (this is important) if you use the entities for these non-standard diacritics, they will always be positioned perfectly over their base characters. (If you really need to know why, brace yourself for a very technical lecture lasting about an hour.)

A third caveat: in most text-processing applications, the features needed to make entities work are on by default, but *in Microsoft Word they are not*. You've got to turn them on. To do so, open the "Font" dialog, click over to the "Advanced" tab, and enable **Kerning**, **Standard Ligatures**, and **Contextual Alternates**. Come to think of it, you should do this for every font you use (do it for the "Normal" style and check "Apply to Template" before clicking "OK"). You may be surprised what wonders a font is capable of performing when these three features are enabled.

Here are the entities. I should mention that there are many more diacritics in Junicode 2 than there are entities.

Encoding	Entity	Example	Encoding	Entity	Example
035B	&_zz;	◌ͻ	F03D	&_thorn;	◌ᵬ
1DD3	&_oa;	◌ᵕ	1DE3	&_rr;	◌ʳ

Encoding	Entity	Example	Encoding	Entity	Example
0305	&_ol;		1DE5	&_longs;	
0363	&_a;		1DD8	&_dins;	
F012	&_b;		1DD5	&_ao;	
0368	&_c;		1DD6	&_av;	
0369	&_d;		F135	&_eogo;	
0364	&_e;		F136	&_emac;	
F017	&_f;		F02F	&_idotl;	
1DDA	&_g;		F031	&_jdotl;	
036A	&_h;		F13E	&_oogo;	
0365	&_i;		F032	&_oslash;	
F030	&_j;		F13F	&_omac;	
1DD3	&_k;		1DD2	&_us;	
1DDD	&_l;		1DD1	&_ur;	
036B	&_m;		F013	&_bsc;	
1DE0	&_n;		F016	&_dsc;	
0366	&_o;		1DDB	&_gsc;	
F025	&_p;		F01C	&_ksc;	
F033	&_q;		1DDD	&_lsc;	
036C	&_r;		036B	&_msc;	
1DE4	&_s;		1DE0	&_nsc;	
036D	&_t;		036C	&_rsc;	

Encoding	Entity	Example	Encoding	Entity	Example
0367	&_u;		F02A	&_tsc;	
036E	&_v;		F036	&_an;	
F03C	&_w;		F038	&_ar;	
036F	&_x;		F03A	&_ansc;	
F02B	&_y;		F130	&_arsc;	
1DE6	&_z;		F03E	&_orr;	
1DD4	&_aelig;		F03F	&_oru;	
1DD9	&_eth;		F03B	&_tins;	

Finally, Junicode 2 has several OpenType features for producing alternative shapes of certain diacritics. These are "Character Variants," cv84–cv92 and Stylistic Set 20 (ss20):

cv84 transforms U+1DD1 or &_ur; () any of the MUFI diacritics encoded in the PUA. For details, see the "Feature Reference."

cv85 transforms U+1DD3 or &_oa; () into . Alternative: U+F1C1.

cv86 transforms U+1DD8 or &_dins; () into ().

cv87 transforms U+1DE3 or &_rr; () into . Alternative: U+F1C2.

cv88 transforms U+0308 () into ().

cv89 transforms U+0305 () into (). Alternative: U+F1C0.

cv90 transforms U+0303 () into . Alternative: U+F1CC.

cv91 produces variants of U+0335 (short horizontal stroke) that are either widened or offset left or right. For details, see "Feature Reference."

cv92 transforms U+032E (, breve below) into a version that spans three characters (). Alternative: U+F1FC.

ss20 alters the position (and, when necessary, the shape) of several diacritics when placed over letters with ascenders (e.g. h&_munc; + ss10 = h̄ + ss20 = h̄).

Unfortunately, MS Word provides no access to Character Variants, so you must either use an app that does (e.g. LibreOffice) or directly enter non-standard character codes listed as alternatives above, though these will not always be positioned correctly over base characters.