MuseScore Page Settings Changes

XML File Format Changes

The spatium is now stored as an integer in points @72dpi x 10,000.

The rest of the page styles numeric values are also now stored in those same units.

The width and height widgets have limits that prevent 32bit signed integer overflow.

These are the new `Sid` values:

`pageSize`	`pageFullWidth`	`marginOddLeft`	`marginEvenTop`
`pageUnits`	`pageFullHeight`	`marginOddRight`	`marginEvenBottom`
`pageOrientation`		`marginOddTop`	
		`marginOddBottom`	

`pageSize` is stored as an integer version of the QPageSize::PageSizeId.

`pageUnits` and `pageOrientation` are stored as strings, for readability.

Internationalization and Default Values

MuseScore now uses QLocale and QLocale::MeasurementSystems to set a user's default units to millimeters or inches. It does this in the base style, which can be overriden by the user.

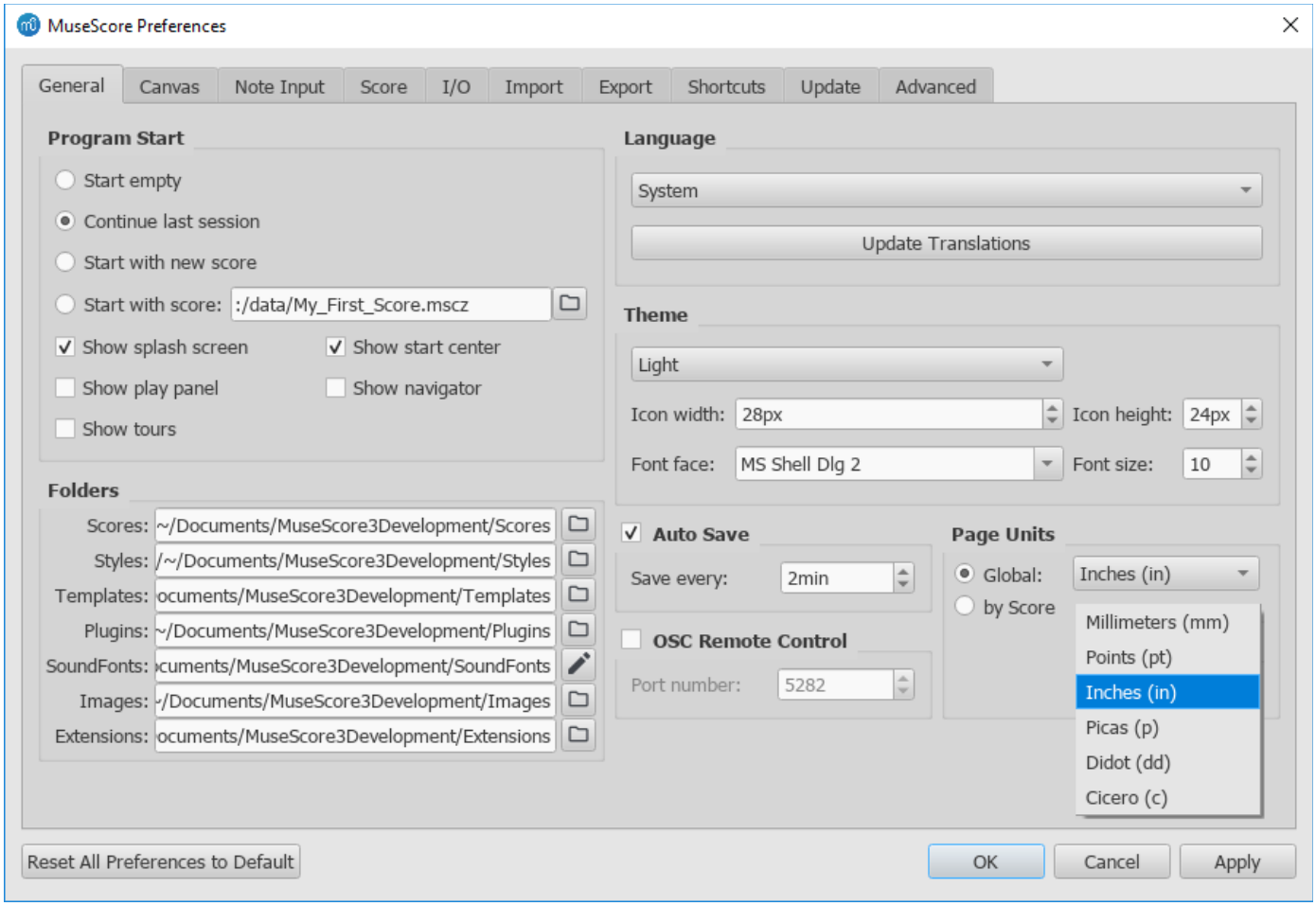
Additionally, if you use imperial units, MuseScore now defaults the preset page size to US Letter vs A4. The default margins remain at 10mm, 10mm, 10mm, 20mm. They are converted to other units as needed. Margins and page width/height are all rounded to two decimal places in pagesettings.ui and in QPageSize and QPageLayout.

mscore/prefsdialog.ui

There are two standard desktop publishing methods for storing the page units:

- A global user preference (MS Office, Apache Open Office)
- As a document property, stored in each file (Scribus, some Adobe apps)

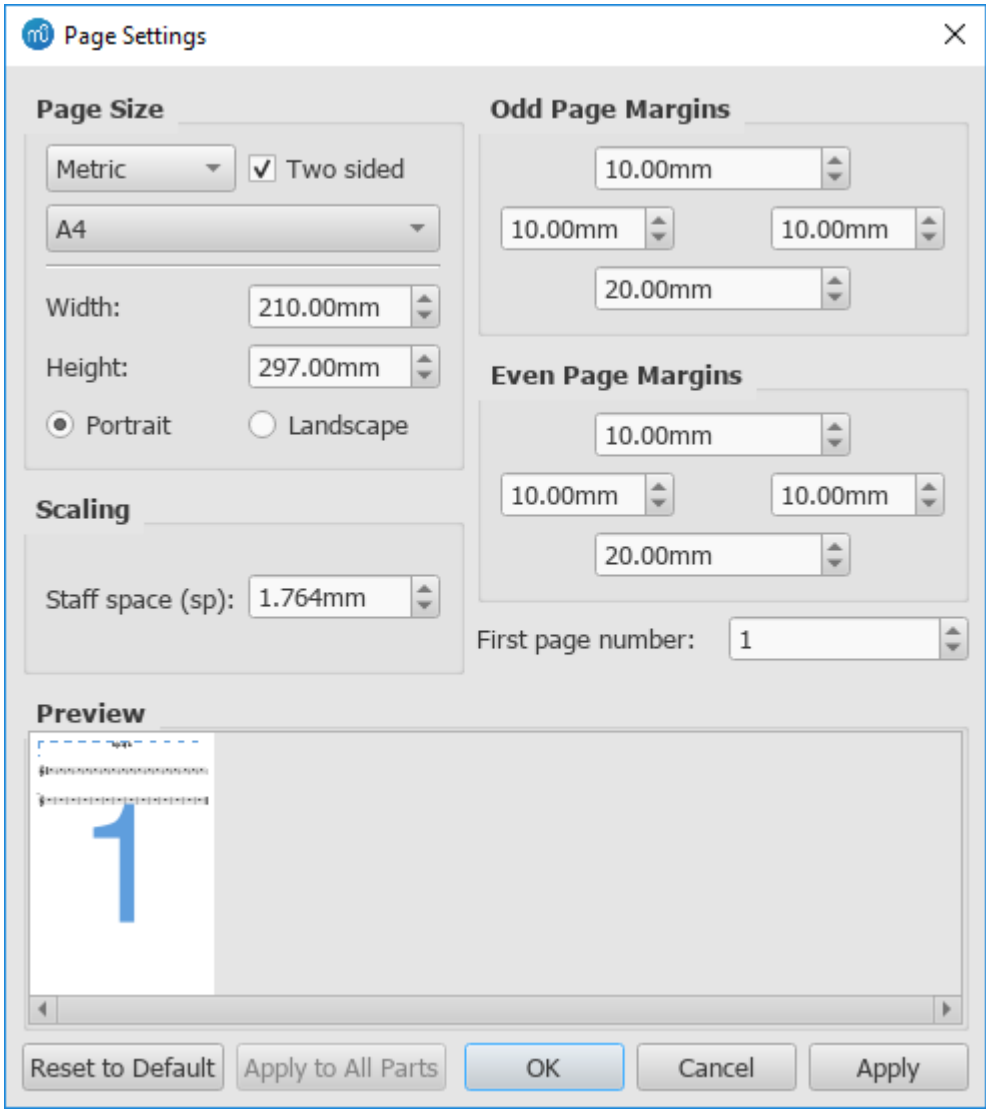
MuseScore now does a combination of those two. The Preferences dialog General tab now has a pair of radio buttons for Page Units: Global and by Score, and a drop-down list of units based on the QPageSize::Unit enumeration.



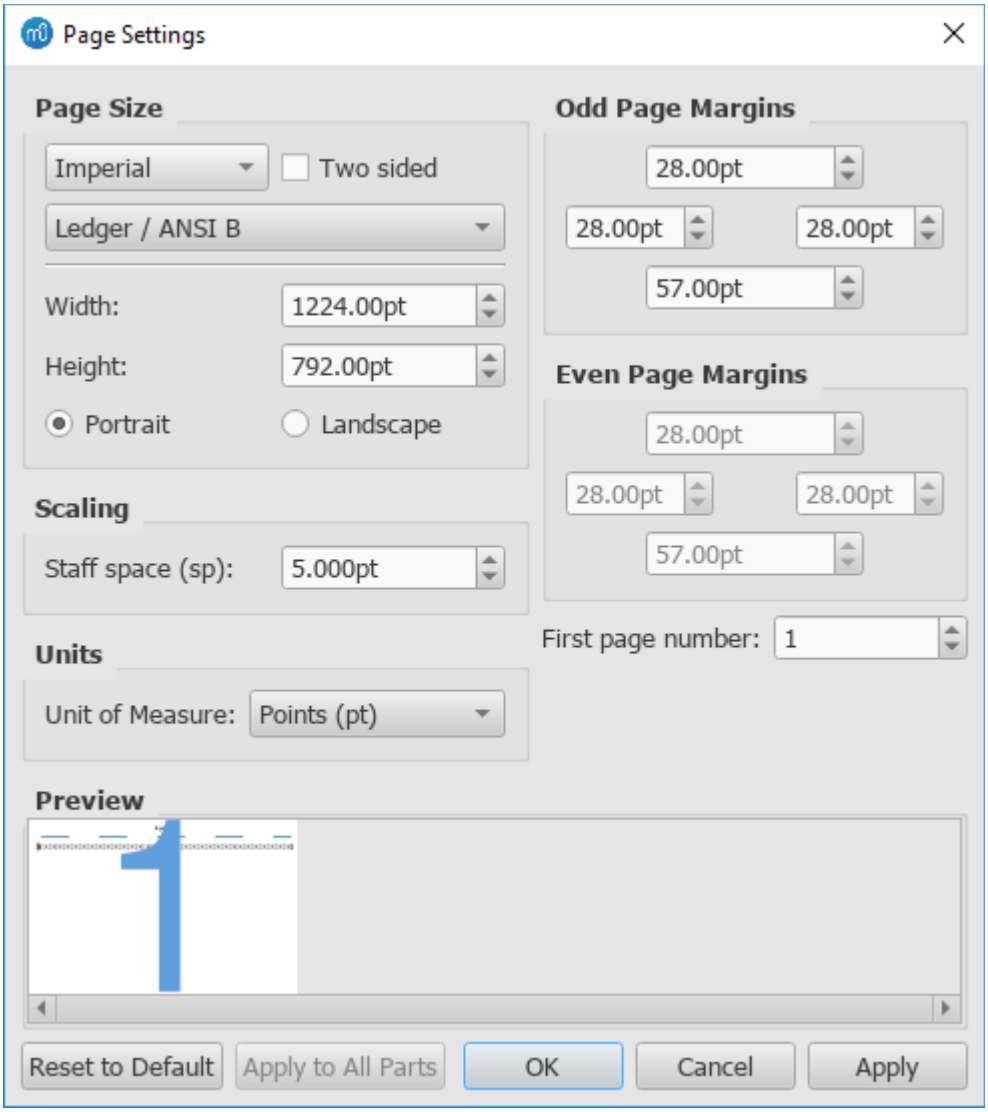
If Set units by score is checked, the combobox specifies the default units for new scores.

mscore/pagesettings.ui

1) If the user selects the Global option, then the units drop-down is hidden. The dialog will use the appropriate text suffix when displaying numbers in those global units, but there will be no radio buttons or drop-downs for units.



On the other hand, if the user selects the by Score option, the user will select the units in the page settings dialog instead of the preferences dialog. The current pair of inches vs millimeters radio buttons is replaced by a drop-down list identical to the one in the preferences dialog, described previously.



2) There is a new Reset to Default button that resets to MScore::defaultStyle() for the page settings styles only.

3) Qt's list of preset paper sizes is 119 items long. That is user-unfriendly. I have implemented a solution that can be refined easily over time.

First, I eliminated all the envelope sizes. Then I grouped the remaining 82 paper sizes into 3 types (top-left drop-down list in the dialog):

- Metric = 28 sizes: ISO A, ISO B, ISO Extra
- Imperial = 28 sizes: Letter/Legal/etc, Extra, Imperial, Ansi, Arch (Architectural)
- Other = 25 sizes ISO Jis B (Japanese), FanFold, PRC, Postcards, miscellaneous

Plus the Custom page size, which is always available to the user as the first item in each list. Those three paper size types are in a new drop-down above the paper sizes list. The default selection of paper size type is done by locale: Metric and Imperial (never Other). When you open the page settings dialog, it uses the score's Sid::pageSize to

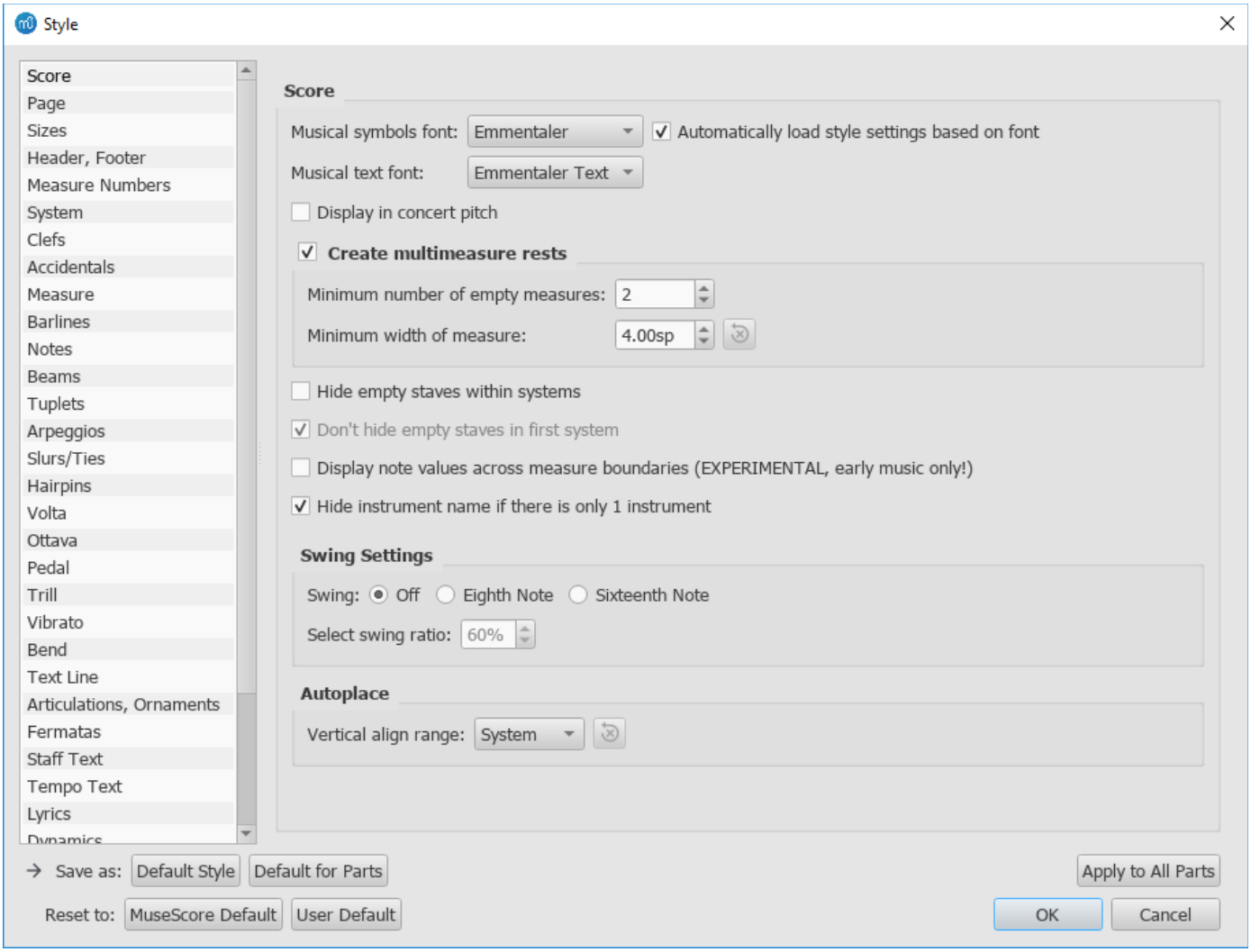
select the correct paper size type and page size. If the score's page size is Custom, the type is set by locale.

The list of paper size types, and each type's list of sizes is easily modified. See the last page of this document for tables listing the current contents of each type.

mscore/editstyle.ui

editstyle.ui has four new buttons:

1. Reset to MuseScore Default resets to `MScore::baseStyle()`.
2. Reset to User Default resets to `MScore::defaultStyle()`, the user's default style. It replaces the Format menu's "Reset Style" option.
3. Save as Default Style saves the current settings as "default.mss" in the default Styles folder.
4. Save as Default for Parts saves the current settings as "part.mss".



Units and Units Text

Qt has infrastructure for page units. It even includes locale-specific, translated versions of standard page size names. But it does not include any kind of text for units.

MuseScore uses two strings for each unit: name and suffix/abbreviation. Both are used in the two new drop-down units lists. The suffix is used in all the numeric fields in the page settings dialog, which change dynamically, unlike other styles.

Currently MuseScore supports two units: millimeters and inches. The text for those units is hardcoded in pagesettings.cpp, maybe elsewhere too, all based on a boolean. The new implementation uses an integer index into the drop-down lists, which also aligns with the QPageSize::Unit enumeration.

See the top of style.h for new types and globals. The new Ms::units[] global variable is an array of type PageUnit, which includes a string name + suffix, a convertion-to-points factor, and two "singleStep" values for the QDoubleSpinBox widgets in pagesettings.ui.

MStyle::pageSize(), MStyle::pageOdd(), and MStyle::pageEven()

Changing the page settings currently involves three layers:

1) The widgets in pagesettings.ui

2) The preview score's style object, specifically these ten styles:

```
Sid::pageWidth
Sid::pageHeight
Sid::pagePrintableWidth
Sid::pageEvenLeftMargin
Sid::pageOddLeftMargin
Sid::pageEvenTopMargin
Sid::pageEvenBottomMargin
Sid::pageOddTopMargin
Sid::pageOddBottomMargin
Sid::pageTwosided
```

3) The current score's style, which is updated from the preview score when you click Apply or OK.

The page number offset is not cached as a style, but as the `_pageNumberOffset` member variable of class `Score`. QPageLayout does not support this property.

The other property not supported by QPageLayout is `Sid::pageTwosided`.

Because of even/odd pages, MuseScore requires two instances of QPageLayout, to cache all the margin values. Those two QPageLayouts share an instance of QPageSize.

Initially it looked like a good idea to use QPageLayout/QPageSize to replace these style entries. But that has adverse consequences in several places where everything is currently handled by styles, especially reading and writing XML.

Unfortunately, QPageLayout/QPageSize round values - points are rounded to whole numbers. So I created a new subclass: `class MPageLayout : public QPageLayout`. MPageLayout handles all the unit conversions with full precision and provides convenient access directly to numeric values, without intervening layers of QRectF or QMarginsF. MPageLayout's public methods (see style.h):

return points @720,000dpi	return points @360dpi
`MPageLayout::widthPoints()`	`MPageLayout::width()`
`MPageLayout::heightPoints()`	`MPageLayout::height()`
`paintWidthPoints not needed`	`MPageLayout::paintWidth()`
`MPageLayout::leftMarginPoints()`	`MPageLayout::leftMargin()`
`MPageLayout::rightMarginPoints()`	`MPageLayout::rightMargin()`
`MPageLayout::topMarginPoints()`	`MPageLayout::topMargin()`
`MPageLayout::bottomMarginPoints()`	`MPageLayout::bottomMargin()`

Three new variables/methods added to class MStyle:

`QPageSize* _pageSize`	`MStyle::pageSize()`	`MStyle::setPageSize()`
`MPageLayout* _pageOdd`	`MStyle::pageOdd()`	`MStyle::setPageOdd()`
`MPageLayout* _pageEven`	`MStyle::pageEven()`	`MStyle::setPageEven()`

pageOddRightMargin = pagePrintableWidth + pageEvenLeftMargin

pagePrintableWidth and pageEvenLeftMargin are made obsolete by these changes. In their place there is one new style: `Sid::pageOddRightMargin`. Two for one is a good deal, and it directly imitates the way the page settings dialog box operates. It now includes all four odd margins and the top/bottom even margins. Even left/right margins are calculated from the odd margins based on Sid::pageTwosided.

For the equivalent of pagePrintableWidth, use MStyle::pageOdd().paintWidth() instead. It's already converted to 360dpi.

pageSize, pageUnits, and pageOrientation

These are three new styles. They could be eliminated as styles if more code was written to convert QPageLayout to/from XML, but it's probably not worth it.

`Sid::pageSize` caches/stores the QPageSize::PageSizeId as an integer. If the score's page size is not Custom, then the width and height could be be optimized out of file storage.

`Sid::pageUnits` caches/stores the QPageSize::Unit for this score as a string. It could be optimized out if the user has selected Set units globally in their preferences, and they are using the base value.

`Sid::pageOrientation` works with Sid::PageSize to replace pageWidth and pageHeight for non-custom page sizes. It is necessary for any size where the portrait orientation has a width greater than height, for example QPageSize::Ledger.

Page Orientation

MuseScore currently assumes that if width > height, the page is in landscape orientation, else it's in portrait. This is not a valid assumption, as seen in both 2.3 and 3.0 via problems with Ledger vs Tabloid size. Ledger portrait == Tabloid landscape. Ledger is a preset page size where width > height.

What does this matter? It seems that QPrinter (and hardware printers) care about orientation. This is even more important for custom page sizes that are trying to imitate a real paper size that might be wider than tall.

So, I have de-linked width and height from orientation in this way: Changes to width and height widgets in pagesettings.ui do not change the orientation unless the width and height exactly match a preset page size in the other orientation. Changing width and height tries to match to a preset page size in the current list (by page type). It does not match across type (Metric, Imperial, Other).

Metric (all three tables in numerical/list order, top-to-bottom, left-to-right)

ISO A	ISO B	ISO Extra
QPageSize::A4	QPageSize::B5	QPageSize::A3Extra
QPageSize::A0	QPageSize::B0	QPageSize::A4Extra
QPageSize::A1	QPageSize::B1	QPageSize::A4Plus
QPageSize::A2	QPageSize::B2	QPageSize::A4Small
QPageSize::A3	QPageSize::B3	QPageSize::A5Extra
QPageSize::A5	QPageSize::B4	QPageSize::B5Extra
QPageSize::A6	QPageSize::B6	
QPageSize::A7	QPageSize::B7	
QPageSize::A8	QPageSize::B8	
QPageSize::A9	QPageSize::B9	
QPageSize::A10	QPageSize::B10	

Imperial (QPageSize:: is implied)

Ansi	Ansi Extra	Architectural	Imperial
Letter	LegalExtra	ArchA	Imperial7x9
Legal	LetterExtra	ArchB	Imperial8x10
Executive	LetterPlus	ArchC	Imperial9x11
Folio	LetterSmall	ArchD	Imperial9x12
Ledger	TabloidExtra	ArchE	Imperial10x11
Tabloid			Imperial10x13
AnsiC			Imperial10x14
AnsiD			Imperial12x11
AnsiE			Imperial15x11

Other (QPageSize:: is implied)

Japanese	Miscellaneous	PRC	FanFold
JisB0	ExecutiveStandard	Prc16K	FanFoldUS
JisB1	Note	Prc32K	FanFoldGerman
JisB2	Quarto	Prc32KBig	FanFoldGermanLegal
JisB3	Statement		
JisB4	SuperA		
JisB5	SuperB		
JisB6	Postcard		
JisB7	DoublePostcard		
JisB8			
JisB9			
JisB10			