**Finding or Verifying Credit Card Numbers**

With a few simple regular expressions, you can easily verify whether your customer entered a valid credit card number on your order form. You can even determine the type of credit card being used. Each card issuer has its own range of card numbers, identified by the first 4 digits.

You can use a slightly different regular expression to find credit card numbers, or number sequences that might be credit card numbers, within larger documents. This can be very useful to prove in a security audit that you're not improperly exposing your clients' financial details.

We'll start with the order form.

**Stripping Spaces and Dashes**

The first step is to remove all non-digits from the card number entered by the customer. Physical credit cards have spaces within the card number to group the digits, making it easier for humans to read or type in. So your order form should accept card numbers with spaces or dashes in them.

To remove all non-digits from the card number, simply use the "replace all" function in your scripting language to search for the regex [^0-9]+ and replace it with nothing. If you only want to replace spaces and dashes, you could use [ -]+. If this regex looks odd, remember that in a [character class](http://www.regular-expressions.info/charclass.html), the hyphen is a literal when it occurs right before the closing bracket (or right after the opening bracket or negating caret).

If you're wondering what the [plus](http://www.regular-expressions.info/repeat.html" \t "_top) is for: that's for performance. If the input has consecutive non-digits, e.g. 1===2, then the regex will match the three equals signs at once, and delete them in one replacement. Without the plus, three replacements would be required. In this case, the savings are only a few microseconds. But it's a good habit to keep regex efficiency in the back of your mind. Though the savings are minimal here, so is the effort of typing the extra plus.

**Validating Credit Card Numbers on Your Order Form**

Validating credit card numbers is the ideal job for regular expressions. They're just a sequence of 13 to 16 digits, with a few specific digits at the start that identify the card issuer. You can use the specific regular expressions below to alert customers when they try to use a kind of card you don't accept, or to route orders using different cards to different processors. All these regexes were taken from [RegexBuddy's library](http://www.regexbuddy.com/library.html).

* Visa: ^4[0-9]{12}(?:[0-9]{3})?$ All Visa card numbers start with a 4. New cards have 16 digits. Old cards have 13.
* MasterCard: ^5[1-5][0-9]{14}$ All MasterCard numbers start with the numbers 51 through 55. All have 16 digits.
* American Express: ^3[47][0-9]{13}$ American Express card numbers start with 34 or 37 and have 15 digits.
* Diners Club: ^3(?:0[0-5]|[68][0-9])[0-9]{11}$ Diners Club card numbers begin with 300 through 305, 36 or 38. All have 14 digits. There are Diners Club cards that begin with 5 and have 16 digits. These are a joint venture between Diners Club and MasterCard, and should be processed like a MasterCard.
* Discover: ^6(?:011|5[0-9]{2})[0-9]{12}$ Discover card numbers begin with 6011 or 65. All have 16 digits.
* JCB: ^(?:2131|1800|35\d{3})\d{11}$ JCB cards beginning with 2131 or 1800 have 15 digits. JCB cards beginning with 35 have 16 digits.

If you just want to check whether the card number looks valid, without determining the brand, you can combine the above six regexes using [alternation](http://www.regular-expressions.info/alternation.html" \t "_top). A [non-capturing group](http://www.regular-expressions.info/brackets.html" \t "_top) puts the [anchors](http://www.regular-expressions.info/anchors.html" \t "_top) outside the alternation. [Free-spacing](http://www.regular-expressions.info/freespacing.html" \t "_top) allows for comments and for the regex to fit the width of this page.

^(?:4[0-9]{12}(?:[0-9]{3})?          # Visa  
 |  5[1-5][0-9]{14}                  # MasterCard  
 |  3[47][0-9]{13}                   # American Express  
 |  3(?:0[0-5]|[68][0-9])[0-9]{11}   # Diners Club  
 |  6(?:011|5[0-9]{2})[0-9]{12}      # Discover  
 |  (?:2131|1800|35\d{3})\d{11}      # JCB  
)$

These regular expressions will easily catch numbers that are invalid because the customer entered too many or too few digits. They won't catch numbers with incorrect digits. For that, you need to follow the [Luhn algorithm](http://en.wikipedia.org/wiki/Luhn_algorithm), which cannot be done with a regex. And of course, even if the number is mathematically valid, that doesn't mean a card with this number was issued or if there's money in the account. The benefit or the regular expression is that you can put it in a bit of JavaScript to instantly check for obvious errors, instead of making the customer wait 30 seconds for your credit card processor to fail the order. And if your card processor charges for failed transactions, you'll really want to implement both the regex and the Luhn validation.

**Finding Credit Card Numbers in Documents**

With two simple modifications, you could use any of the above regexes to find card numbers in larger documents. Simply replace the [caret and dollar](http://www.regular-expressions.info/anchors.html) with a [word boundary](http://www.regular-expressions.info/wordboundaries.html), e.g.: \b4[0-9]{12}(?:[0-9]{3})?\b.

If you're planning to search a large document server, a simpler regular expression will speed up the search. Unless your company uses 16-digit numbers for other purposes, you'll have few false positives. The regex \b\d{13,16}\b will find any sequence of 13 to 16 digits.

When searching a hard disk full of files, you can't strip out spaces and dashes first like you can when validating a single card number. To find card numbers with spaces or dashes in them, use \b(?:\d[ -]\*?){13,16}\b. This regex allows any amount of spaces and dashes anywhere in the number. This is really the only way. Visa and MasterCard put digits in sets of 4, while Amex and Discover use groups of 4, 5 and 6 digits. People typing in the numbers may have different ideas yet.