Shared: Validations

Setup Guide

Last Revised: August 10, 2020

Applies to these SAP Concur solutions:

- ☒ Expense
  - ☒ Professional/Premium edition
  - ☐ Standard edition
- ☐ Travel
  - ☐ Professional/Premium edition
  - ☐ Standard edition
- ☒ Invoice
  - ☒ Professional/Premium edition
  - ☐ Standard edition
- ☒ Request
  - ☒ Professional/Premium edition
  - ☐ Standard edition

Table of Contents

Revision History

Date	Notes/Comments/Changes
January 21, 2022	Updated the copyright year; no other changes; cover date not updated
April 15, 2021	Updated the copyright year; no other changes; cover date not updated
August 10, 2020	Added a procedure for assigning a validation to a field. Other minor updates and fixes.
April 27, 2020	Renamed the Authorization Request check box to Request on the guide's title page; cover date not updated
January 15, 2020	Updated the copyright; no other changes; cover date not updated
December 13, 2019	Added important note that field validations are not supported for the Expense Amount field.
February 11, 2019	Updated the copyright; no other changes; cover date not updated
April 16, 2018	Changed the check boxes on the front cover; no other changes; cover date not updated
January 29 2018	Updated the copyright; no other changes; cover date not updated
December 14 2016	Changed copyright and cover; no other content changes.
October 27 2016	Updated the guide content to new corporate style; no content changes.
May 6 2015	Updated the screen shots to the enhanced UI; no other content changes
October 10 2014	Added information about the two user interfaces; no other content changes
January 23 2014	Cover and copyright changes; no other content changes
February 27 2013	Made the following changes • Name change from "Travel Request" to "Request" • Removed references to using the classic user interface
December 28 2012	Made rebranding and/or copyright changes; no content changes
February 2012	Changed copyright; no content change
December 31 2010	Updated the copyright and made rebranding changes; no content changes
November 2010	Added information about the current user interface
December 2009	Changed to stand-alone setup guide; no content change

Validations

NOTE: Multiple SAP Concur product versions and UI themes are available, so this content might contain images or procedures that do not precisely match your implementation. For example, when SAP Fiori UI themes are implemented, home page navigation is consolidated under the SAP Concur Home menu.

Section 1: Permissions

A company administrator may or may not have the correct permissions to use this feature. The administrator may have limited permissions, for example, they can affect only certain groups and/or use only certain options (view but not create or edit).

If a company administrator needs to use this feature and does not have the proper permissions, they should contact the company's Concur administrator.

In addition, the administrator should be aware that some of the tasks described in this guide can be completed only by Concur. In this case, the client must initiate a service request with Concur Client Support.

Section 2: Overview

The Validations feature allows the administrator to add custom field-level validations to the system. The client can use validations to validate the data an employee inputs into the system while completing fields (such as phone number).

NOTE: There are several standard validations that are loaded with Concur. System field validations are not visible to anyone, as they primarily validate data types. Use this section to create validations that work in conjunction with these standard validations.

NOTE: **IMPORTANT!** Field validations are NOT supported for the Expense Amount field. Expenses are often created by the system and cannot respond to data entry validations, and the system may not work correctly if validations are set for this field.

Section 3: Work with Validations

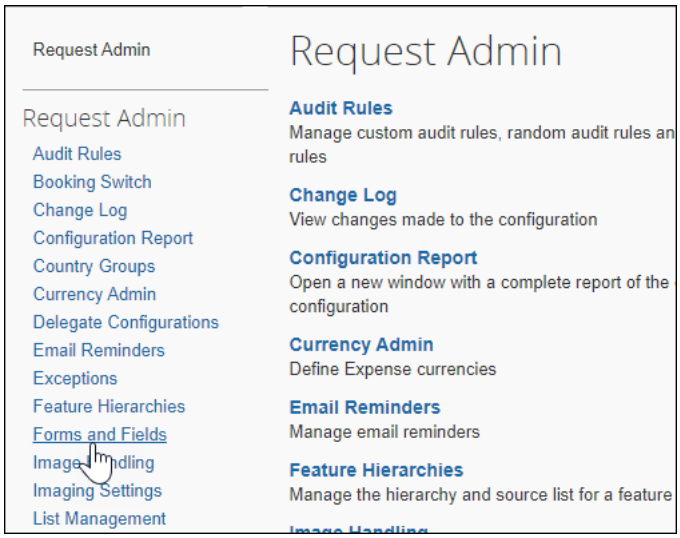
Access the Validations Page

- ***To access the Validations page:***

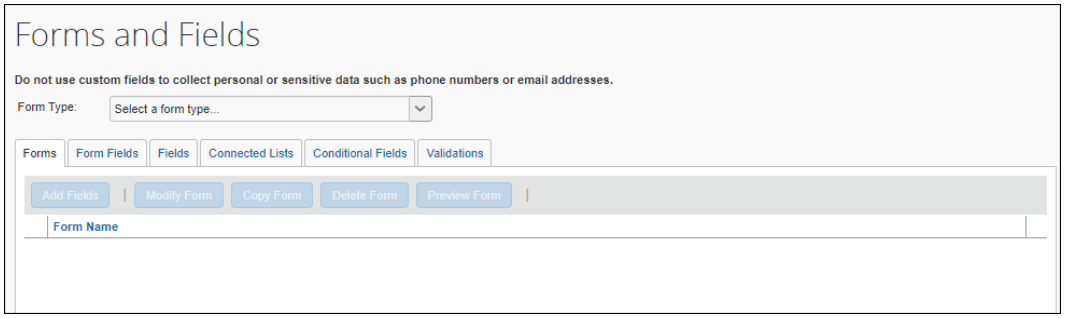
1. Click **Administration** and then click **Request**, **Expense**, or **Invoice**.

NOTE: The items that appear in the list after you click **Administration** depend on which SAP Concur products your company uses and the permissions assigned to you.

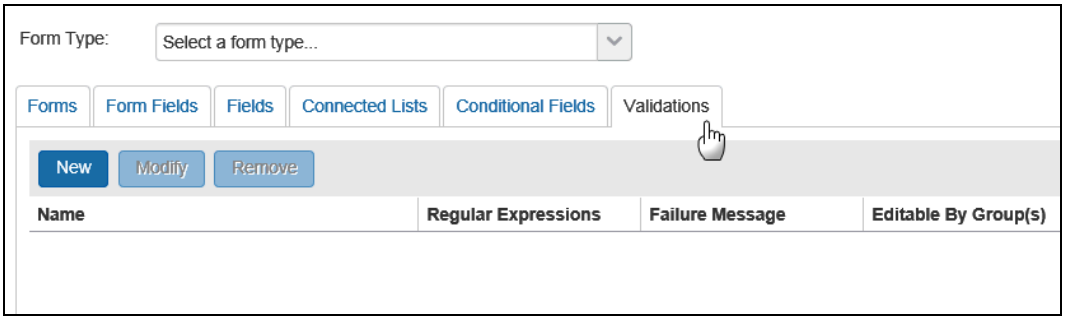
2. Click **Forms and Fields**.



The **Forms and Fields** page appears.



3. Click the **Validations** tab. The **Validations** page appears.



For more information about the **Forms and Fields** page, see the Concur Invoice: Forms and Fields Setup Guide, the Concur Expense: Forms and Fields Setup Guide, or the Concur Request: Forms and Fields Setup Guide.

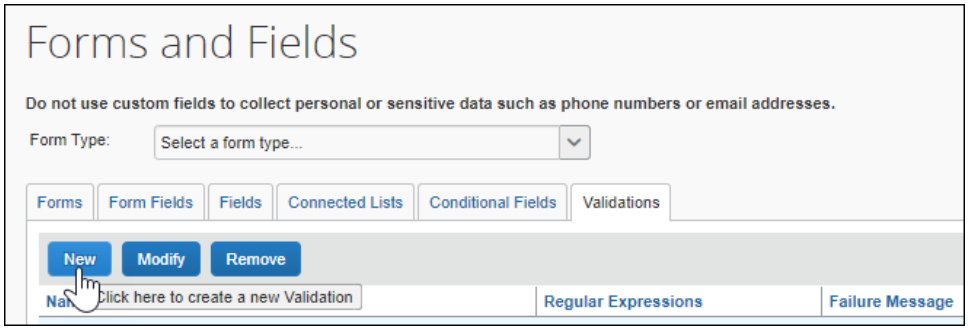
Create a New Validation

There is a very specific syntax required to create a new validation.

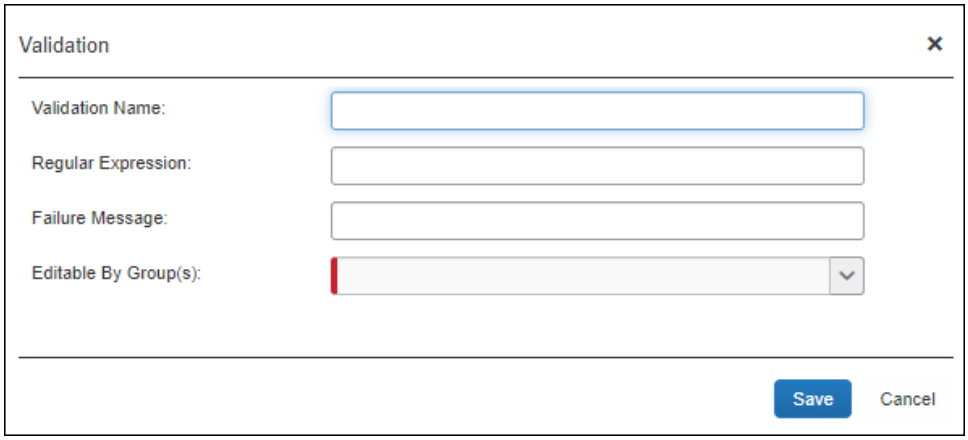
For more information about the required syntax, refer to the *Writing Regular Expressions* section of this guide.

To add a new validation:

1. On the **Validations** page, click **New**.



The **Validation** window appears.



2. Enter the appropriate information into the fields.

Field	Description
Validation Name	Enter a name for the validation.

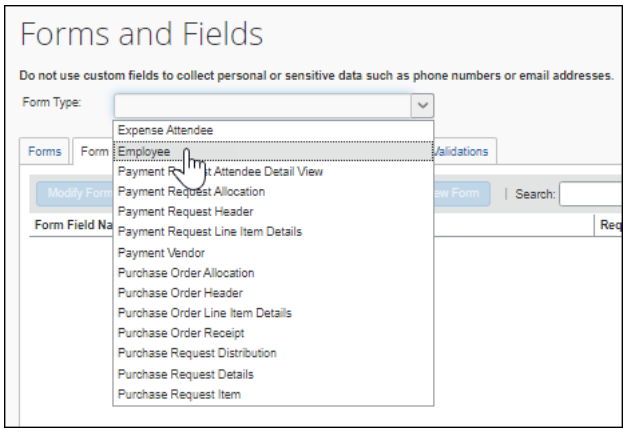
Field	Description
Regular Expression	Type a valid regular expression. **NOTE:** Regular expressions validate the text input by defining the pattern of characters the field requires. For example, if the field requires a phone number with the pattern of three numbers, followed by a dash, three numbers, dash, and four numbers (xxx-xxx-xxxx). The regular expression that you would enter into the **Regular Expression** field would be, `^[0-9]{3}-[0-9]{3}-[0-9]{4}$`. For more information about the required syntax for creating a valid regular expression, refer to the *Writing Regular Expressions* section of this guide.
Failure Message	Type the message you want the user to see if the data they enter into the field being validated doesn't match the validation criteria.
Editable By Group(s)	Select the group rights that an administrator must have in order to modify the data. • If you select *Global*, then the validation can be edited or deleted by any employee assigned to administer the Global group. All other administrators assigned at lower levels in the hierarchy can copy this validation, but not edit or delete it. Depending on your rights, you may not be able to select Global. You are only allowed to select your own group, or groups beneath you. • If you select one or more groups in this field, such as *United States* and *Europe*, then the administrator must have rights for both United States **and** Europe. Through inheritance, the administrator will also have rights to edit anything beneath United States and Europe. If the administrator only has rights for United States, then the information can only be viewed and copied. If the administrator has rights for one or more group(s) above United States and Europe, such as Global, then they also has rights to both of these groups.

3. Click **Save**.

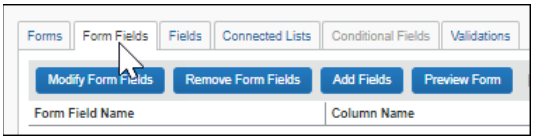
Assign a Validation to a Field

To assign a validation to a field

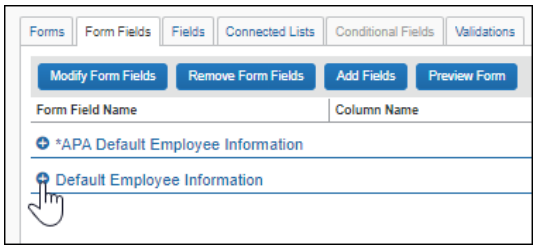
1. On the **Forms and Fields** page, select the form that contains the field you want to validate from the **Form Type** list.



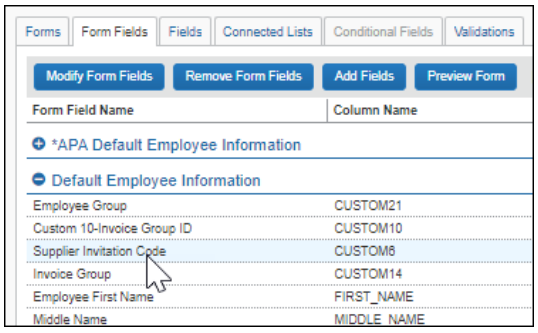
2. Click on the **Form Fields** tab.



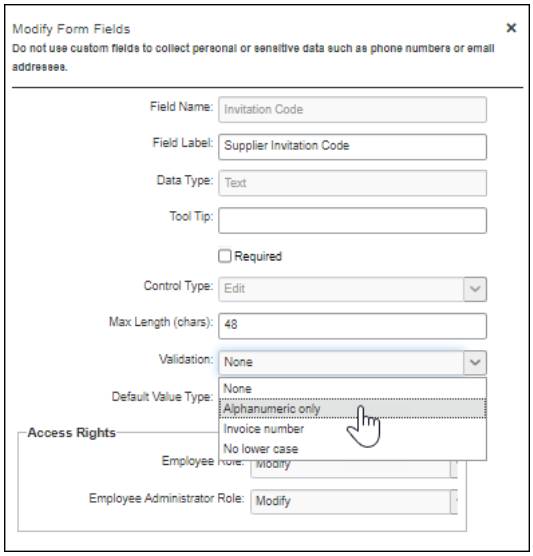
3. On the **Form Fields** page, click the plus sign (+) next to the form that contains the field to which you want to assign a validation.



4. Double-click on the field to which you want to assign a validation.



5. On the **Modify Form Fields** page, select the validation you want to assign to the field from the **Validation** list.



6. Click **Save**.

Edit an Existing Validation

To edit an existing validation:

1. Select the validation that you want to edit.
2. Click **Modify**. The **Validation** window appears.
3. Edit the information as necessary.
4. Click **Save**. The information is updated and the **Validations** page appears.

Delete an Existing Validation

Validations that are assigned to a field cannot be deleted.

To delete an existing validation:

1. Select the validation that you want to delete.
2. Click **Remove**. A confirmation dialog box appears.
3. Click **OK**. The selected validations are deleted.

Section 4: Writing Regular Expressions

A regular expression pattern is composed of simple characters, such as `/abc/`, or a combination of simple and special characters, such as `/ab*c/` or `/Chapter (\d+)\.\d*/`. The last example includes parentheses, which the system

uses as a memory device. The system remembers the match made with this part of the pattern, for later use.

Use Simple Patterns

You construct simple patterns out of characters for which you want to find a direct match. For example, the pattern `/abc/` matches character combinations in strings only when exactly the characters 'abc' occur together and in that order. Such a match would succeed in the strings "Hi, do you know your abc's?" and "The latest airplane designs evolved from slabcraft." In both cases the match is with the substring 'abc'. There is no match in the string "Grab crab" because it does not contain the substring 'abc'.

Use Special Characters

When the search for a match requires something more than a direct match, such as finding one or more b's, or finding whitespace, the pattern includes special characters. For example, the pattern `/ab*c/` matches any character combination in which a single 'a' is followed by zero or more 'b's (`*` means 0 or more occurrences of the preceding item) and then immediately followed by 'c'. In the string "cbbabbbbcdebc," the pattern matches the substring 'abbbbc'.

Table 1 provides a complete list and description of the special characters that you can use in regular expressions.

Table 1: *Special Characters that can be used in Regular Expressions*

Special Characters	Definition
\	Means one of the following: 1. If the system usually treats this character literally, this indicates that the next character is special and should not be interpreted literally. For example, /b/ matches the character 'b'. When you place a backslash in front of the b, /\b/, the system understands the character to be special, and that it is now an instruction to match a word boundary. 2. If the system usually treats the character specially, this indicates that the next character is not special, and that the system should therefore interpret it literally. For example, the asterisk (*) is a special character that means 0 or more occurrences of the preceding item should be matched. So, /a*/ means that 0 or more a's should be matched. Conversely, to match the * literally, precede it with a backslash, /a*/. This will now only match 'a*'.
^	Matches the beginning of input. If the multi-line flag is set to True, it also matches immediately after a line break character. For example, /^A/ does not match the 'A' in "an A", but it does match the first 'A' in "An A"
$	Matches the end of input. If the multi-line flag is set to True, it also matches immediately before a line break character. For example, /t$/ does not match the 't' in "catch", but it does match the last 't' in "cat".

Special Characters	Definition
*	Matches the preceding character 0 or more times. For example, /fo*/ matches 'fool' in "A fool's errand", and 'f' in "The foibles of man", but nothing in "Time goes by".
+	Matches the preceding character 1 or more times. This is equivalent to {1,}. For example, /a+/ matches the 'a' in "Candy" and all of the a's in "Caaandy".
?	Matches the preceding character 0 or 1 time. For example, /e?le?/ matches the 'el' in "angel" and the 'le' in "angle". If this is used immediately after any of the quantifiers, (*, +, ?, or {}) it will make the quantifier match the minimum number of times, as opposed to the default, which matches the maximum number of times. This is also used in lookahead assertions, described in the x(?=y) and x(?!y) sections of this table.
. (Decimal Point)	Matches any single character, except newline. For example, /.n/ matches 'an' and 'on' in "No, I don't see an apple on that tree", but not in "No".
(x)	Matches 'x' and remembers the match. These are called capturing parentheses. For example, /(foo)/ matches and remembers 'foo' in "football". The matched substring can be recalled from the resulting array's elements [1], ..., [n].
(?:x)	Matches 'x', but does not remember the match. These are called non-capturing parentheses. The matched substring cannot be recalled from the resulting array's elements [1], ..., [n].
x(?=y)	Matches 'x' only if 'x' is followed by 'y'. For example, /Back(?=Spring)/ matches 'Back', but only if it is followed by 'Spring'. /Back(?=Spring|Yard)/matches 'Back' only if it is followed by 'Spring' or 'Yard'. However, neither 'Spring' nor 'Yard' is part of the match results.
x(?!y)	Matches 'x' only if 'x' is not followed by 'y'. For example, /\Back+(?!\Spring)/ matches 'Back', only if it is not followed by 'Spring'. Another example, using a regular expression is, /\d+(?!\)./.exec("3.141"); whereby, 141 is matched, but not 3.141.
X|Y	Matches either 'x' or 'y'. For example, /one|two/ matches 'one' in "One day" and 'two' in "Two days".
{n}	Where *n* is a positive integer. Matches exactly *n* occurrences of the preceding character. For example, /a{2} does not match the 'a' in "candy", but it matches all of the a's in "caandy", and the first two a's in "caaaandy".
{n,}	Where *n* is a positive integer. Matches at least *n* occurrences of the preceding character. For example, /a{2} does not match the 'a' in "candy", but it matches all of the a's in "caandy" and "caaaandy".
{n,m}	Where *n* and *m* are positive integers. Matches at least *n* occurrences, and at most *m* occurrences, of the preceding character. For example, /a{1,3}/ matches nothing in "cndy", the 'a' in "candy," the first two a's in "caandy," and the first three a's in "caaaaaaandy" Notice that when matching "caaaaaaandy", the match is "aaa", even though the original string contained more a's.

Special Characters	Definition
[xyz]	A character set. Matches any one of the enclosed characters. You can specify a range of characters by using a hyphen. For example, [abcd] is the same as [a-d]. They match the 'b' in "brisket" and the 'c' in "ache".
[^xyz]	A negated or complemented character set. It matches anything that is not enclosed in the brackets. You can specify a range of characters by using a hyphen. For example, [^abc] is the same as [^a-c]. They initially match the 'r' in "brisket" and 'h' in "chop."
[\b]	Matches a backspace. **NOTE:** Not to be confused with \b.
\b	Matches a word boundary, such as a space or a newline character. For example, /\bn\w/ matches the 'no' in "noonday";/\wy\b/ matches the 'ly' in "possibly yesterday." **NOTE:** Not to be confused with [\b].
\B	Matches a non-word boundary. For example, /\w\Bn/ matches 'on' in "noonday", and /y\B\w/ matches 'ye' in "possibly yesterday."
\c*X*	Where *X* is a control character. Matches a control character in a string. For example, /\cM/ matches control-M in a string.
\d	Matches a digit character; equivalent to [0-9]. For example, /\d/ or /[0-9]/ matches '2' in "B2 is the suite number."
\D	Matches any non-digit character; equivalent to [^0-9]. For example, /\D/ or /[^0-9]/ matches 'B' in "B2 is the suite number."
\f	Matches a form feed.
\n	Matches a line feed.
\r	Matches a carriage return.
\s	Matches a single, white-space character, including SPACE BAR, TAB, form feed, line feed. Equivalent to [\f\n\r\t\v\u00A0\u2028\u2029]. For example, /\s\w*/ matches ' bar' in "foot bath."
\S	Matches a single character, other than white-space. Equivalent to [^\f\n\r\t\v\u00A0\u2028\u2029]. For example, /\S\w*/ matches 'foo' in "football."
\t	Matches a TAB.
\v	Matches a vertical TAB.
\w	Matches any alphanumeric character including the underscore (_). Equivalent to [A-Za-z0-9_]. For example, /\w/ matches 'a' in "apple," '5' in "$5.28," and '3' in "3D."
\W	Matches any non-word character. Equivalent to [^A-Za-z0-9_]. For example, /\W/ or /[^$A-Za-z0-9_]/ matches '%' in "50%."

Special Characters	Definition
\n	Where *n* is a positive integer. A back reference to the last substring matching the *n* parenthetical in the regular expression (counting left parentheses). For example, /apple(,)\sorange\1/ matches 'apple, orange,' in "apple, orange, cherry, peach".
\0	Matches a NUL character. Do not follow this with another digit.
\xhh	Matches the character with the code hh (two hexadecimal digits).
\uhhhh	Matches the character with the code hhhh (four hexadecimal digits).

Use Parentheses

Parentheses around any part of the regular expression pattern cause the system to remember that part of the matched substring. Once the system remembers the substring, the system can recall it for other use.

For example, the pattern `/Chapter (\d+)\.\d*/` illustrates additional escaped and special characters and indicates that part of the pattern should be remembered. It matches precisely the characters 'Chapter ' followed by one or more numeric characters (`\d` means any numeric character and `+` means 1 or more times), followed by a decimal point (which in itself is a special character; preceding the decimal point with \ means the pattern must look for the literal character '.'), followed by any numeric character 0 or more times (`\d` means numeric character, `*` means 0 or more times). In addition, parentheses are used to remember the first matched numeric characters. This pattern is found in "Open Chapter 4.3, paragraph 6" and '4' is remembered. The pattern is not found in "Chapter 3 and 4", because that string does not have a period after the '3'.

To match a substring without causing the matched part to be remembered, in the parentheses preface the pattern with ?:. For example, (?:\d+) matches one or numeric characters but does not remember the matched characters.

Examples of Regular Expressions

The following sections provide examples of how to write regular expressions.

Require a Field to be Uppercase

If you require a field to be all uppercase, alpha-numeric characters, including the German uppercase letters with umlauts (Ä,Ë,Ö,Ü), or an empty string, it would be expressed as:

^[A-Z\u00c4\u00cb\u00d6\u00dc\d]*$

The ^ requires a match at the beginning of the string and the $ requires a match at the end of the string. These are significant and prevent matching in the case where the string is empty. You specify the Unicode characters using the \u00xx escape

format. The \d specifies the digits; you can also express it as 0-9 in the character class.

Use Special Characters to Verify a Phone Number

If you require your users to enter a phone number, you can verify that it is formatted correctly by using this regular expression:

/\(?\d{3}\)?([-\/\.])\d{3}\1\d{4}/

The regular expression looks for zero or one open parenthesis `\(?`, followed by three digits `\d{3}`, followed by zero or one close parenthesis `\)?`, followed by one dash, forward slash, or decimal point and when found, remember the character `([-\/\.])`, followed by three digits `\d{3}`, followed by the remembered match of a dash, forward slash, or decimal point `\1`, followed by four digits `\d{4}`.