# VALIDATION.CFC

## About the Project

Validation.cfc is a ColdFusion Component that performs server-side form validation. It currently validates for required fields, integers, numeric values, e-mail addresses, URLs, IP addresses, social security numbers, United States and Canadian postal codes, US-style Dates, European-style dates, US telephone numbers, US currency, credit card numbers, and passwords. It also has the capability of managing file uploads and can perform mime-type validation.

## License



If you use this component in a live site, I would love to know where! Please end me a quick note at rheldt@ryanheldt.com with the name of the site the URL.

## Credits

This project contains a bunch of regular expressions. Unfortunately, those really aren't my thing, so most were gathered from various places across the internet, including a good number from the Regular Expression Library (regexlib.com). Credit card validation is based on isCreditCard() by Nick de Voil (cflib.org/udf.cfm?id=49).

## Requirements

I know that Validation.cfc runs under ColdFusion MX 7 and ColdFusion 8. It should run on MX 6.1, but that hasn't been confirmed. It also runs on BlueDragon 7 and Railo 3.0. It hasn't been tested on the Smith Project or other versions of the above engines.

## Installation

All you need to is put `Validation.cfc` with your other components. Done.

## Getting Started

Hopefully Validation.cfc is one of the easiest forms of validation you're ever worked with – not only in terms of usability, but also in maintainability. The component currently performs the following types of validation:

- Integers;
- Numbers;
- E-mail Address;
- URLs;
- IP Addresses;
- Social Security Numbers (nnn-nn-nnnn);
- United States and Canada Postal Codes (nnnnn or nnnnn-nnnn or ana-nan);
- US-Style Dates (mm/dd/yyyy);
- European-Style Dates (dd/mm/yyyy);
- Time (hh:mm:ss tt);
- US-Style Telephone Numbers (nnn-nnn-nnnn);
- US Currency (Allows optional "$", optional "-" or "()" but not both, optional; cents, and optional commas separating thousands);
- Credit Cards (Using Luhn Algorithm, also attempts to determine card type);
- Passwords (case-sensitive comparison to make sure two fields match); and
- File mine-types. Also has provides ability to manage your file uploads.

There are two steps in adding validation to your projects: adding one or more hidden fields to you forms to specify what kinds of validation you wish to perform and after the form is submitted invoking the CFC and acting on results.

## Hidden Form Fields

For each type of validation you wish to perform, create a hidden input field containing a semi-colon delimited list of “FIELDNAME|ERROR MESSAGE”. For example, say you have a simple form in which you have a name and e-mail address you want required and you want to make sure a valid e-mail address has been entered. Your hidden fields would appear as:

```
<input name="validate_require" type="hidden" value="Name|'Name' is a
required field.;Email|'E-mail Address' is a required field." />

<input name="validate_email" type="hidden" value="Email|'E-mail
Address' must be a valid e-mail address." />
```

That’s really all there is to it. Of course, there are 16 different forms of validation you can perform:

- validate_require
- validate_integer
- validate_numeric
- validate_email
- validate_url
- validate_ip
- validate_ssn
- validate_postal
- validate_dateus
- validate_dateeu
- validate_time
- validate_phoneus
- validate_currencyus
- validate_creditcard
- validate_password
- validate_file

Hopefully the above field names should be self explanatory. All of them work in the same manner as the example above, with exception of password and file validation.

With password validation, since we’re comparing two fields, we’ll need to add one additional parameter to specify the secondary field we’re matching with the first.

```
<input name="validate_password" type="hidden"
value="Password|Password2|Both passwords must match." />
```

With file validation, you'll need to specify which types of files are accepted. This is done by specifying a comma-delimited list of typical file extensions. Although we're using extensions to specify file types, Validation.cfc actually looks at the mime-type when performing validation. If you wish to allow all file types, enter "*" although I would not recommend this for a whole host of security reasons.

```
<input name="validate_file" type="hidden" value="Document|doc,rtf,
,pdf|'Document' must be either a Word, Rich Text, or Adobe PDF." />
```

## Invoking the Component

Once you have all your hidden fields properly set, all you need to do is call up the validation component and pass in the struct containing your form fields using the `setFields()` method (typically the form scope or the attributes scope if you're a fellow Fuseboxer, and run `validate()`.

```
<cfscript>
    objValidation = createObject("component","com.Validation").init();
    objValidation.setFields(form);
    objValidation.validate();
</cfscript>
```

## Acting on the Results

After performing the validation, there are two methods you can use to determine what to do next.

> `getErrorCount()` returns an integer containing the number of total validation errors. If 0 is returned, then there were no validation errors.
>
> `getMessages()` returns a struct of error messages from the fields that failed validation. Typically, you would call this if `getErrorCount()` retuned something other than zero.

You can direct your application to act on this information however you like. But, since people (including myself) like examples, please refer to `example.cfm` for a working demonstration.

## Validating Credit Cards

Validation.cfc uses the Luhn algorithm (also known as the mod 10 algorithm) to determine if a credit card number is proper. It cannot determine if the cardholder's account is valid. You will need a merchant account to do that. It can, however, help you perform another level of validation by telling you the card type, as determined by the number.

After performing the validation, the `getCardType()` function will return the card type, either "Amex" for American Express, "Diners" for Diner's Club, "Discover" for

Discover Card, “MasterCard” for MasterCard, or “Visa” for Visa. If Validation.cfc is unable to determine the card type, `getCardType()` will return “Unknown.” All you need to pass to the function is the name of the field(s) containing the credit card number. If the field passed isn’t a credit card field, `getCardType()` will return an empty string.

```
<cfscript>
    if(objValidation = getCardType("CardNumber")) neq form.CardType {
        // Card type entered isn't type specified by the user
    }
</cfscript>
```

## File Upload Management

Validation.cfc has the capability to manage your file uploads for you, in addition to validating mime-types. This means that it will handle all of the `<cffile>` stuff, it will create a unique filename (based on the original filename), and it will clean up after itself in the event of a validation failure.

Before you can upload any files, you need to tell Validation.cfc what directory you want them stored in. You can upload more than one file at a time and even put them in different directories. Here’s how:

```
<cfscript>
    stcDirectories = structNew();
    stcDirectories.Document = "#expandPath("./")#";
    stcDirectories.Document2 = "#expandPath("./")#";
    objValidation = createObject("component","com.Validation").init();
    objValidation.setFields(form);
    objValidation.setDirectories(stcDirectories);
    objValidation.validate();
</cfscript>
```

In this case, we have two file fields named `Document` and `Document2`. We’ve created a struct called `stcDirectories` in which we have an entry for each of the two respective file fields. This struct is then passed along to Validation.cfc using the `setDirectories()` method.

In the event of a successful validation, you can use the `getFile()` method to retrieve the names of the uploaded files. Filenames are returned without directory information. Validation.cfc assumes that your application will keep track of directories. If there are any validation errors, all uploaded files will be deleted. This helps keep your application free of any orphaned files.

Please refer to `upload.cfm` for a working demonstration.

Note: please check out the private `loadMimeTypes()` method of Validation.cfc for all of the supported mime-types (it’s the very bottom). You can even add your own by simply adding to the existing array.

## Security and Best Practices

Although setting your validation though hidden fields makes your forms easy to maintain, it might not be the most secure. Would-be hackers and other mischievous folk could easily tamper with the fields, leading to problems.

One way to resolve this issue would be to set the validation fields in your code before you perform the validation. For example:

```
<cfset form.validate_require = "Name|'Name' is a required
field.;Email|'E-mail Address' is a required field." />

<cfset form.validate_email = "Email|'E-mail Address' must be a valid e-
mail address." />
```

This doesn't change the way the way Validation.cfc works, only how it gets its information. Because the settings remain in the code, it's never returned to the browser and cannot be manipulated. This makes it a more secure alternative.

Also, it probably goes without saying that whenever you're dealing with credit card numbers, social security numbers, or any other form or personally identifiable information, always, always use an SSL and if you have to store the information encrypt and tightly control who has access to it. If you're interested in learning more, I suggest checking out the PCI Security Standards (pcisecuritystandards.org).

## Release Notes

| Date | Version | Notes |
|---|---|---|
| 10/8/2008 | 1.00 | Substantially added to the list of built-in MIME types. |
| 10/7/2007 | 0.25 | Added credit card validation (beta). Added getCardType() method. Fixed non-implemented getMessage() method. |
| 9/30/2007 | 0.20 | Complete rewrite. Included support for files, SSNs, and time. Added methods for setting and retrieving information. |
| 6/8/2007 | 0.12 | Initial public release. |
| 5/27/2007 | 0.11 | Added init() constructor. |
| 11/23/2006 | 0.10 | Creation of initial cfc. |

## Feature Requests and Enhancements

I will gladly accept feature requests and enhancement ideas for Validation.cfc. Personally, I still want to add enhanced checkbox validation (like check at least 3 items or no more than 2 items, etc) and some form of user-defined validation using regular expressions. Such features are planned for the 1.0 release. On that note, if you find yourself adding a feature of your own, please pass it along so that everyone in the community can benefit. I shall gladly give credit where credit is due!

I sincerely hope you find Validation.cfc useful. Happy coding and God bless!

Ryan J. Heldt
rheldt@ryanheldt.com