Indigo Form Mail

www.digitalindigo.com

Welcome to Digital Indigo's Indigo Form Mail Product (IFM). IFM is designed to be a general-purpose form processor. Therefore the core of IFM is to take a web-base form and put its data into an email.

History…
IFM is inspired by a program called Form Mail; probably the most popular form processor on the planet. However Form Mail was and is plagued with security issues, and lacks features that are necessary in today's evolving Internet. IFM is a plug-in replacement for Form Mail. In other words you can take scripts which used the formmail.pl or formmail.cgi program in the past, and direct them to the iformmail.php script. Most features from Form Mail are supported in IFM, and under most circumstances the only thing that should need changed in a Form Mail script is the location it is posted to (i.e. pointing it to iformmail.php rather than formmail.cgi).

Security…
IFM is designed with security in mind. Our goal is to reduce the abuse by spammers while providing maximum functionality in a form processor. We also have many extended features and add new functionality as necessary.

Let's be open…
IFM is provided as Open Source technology and comes without support, except for individuals who paid for extended modules or established a support account with Digital Indigo. Support contracts as well as priority development are available upon request. Please contact us for additional information about either of these services. Entities wishing to donate to the development of IFM can do so by mailing checks or, in the tradition of Open Source, postcards to the PO Box listed at http://www.digitalindigo.com/contact/.

Platform Independence
IFM was written in PHP and special care was take to ensure that it runs equally well on UNIX, GNU/Linux, Mac OS X, and Windows system. This allows the user to easily move to different web servers and operating systems while maintaining a working form system.

Development
IFM is developed on the GNU/Linux platform and then test on other platforms for compatibility. This manual is created using Open Office on Apple's Macintosh OS X operating system.

To Do...
We do have a to do list for IFM. Among some of the major priorities are the following: Better documentation in this manual, more Form Mail features, added code examples, DB/SQL/Flat file support, plug-in capabilities.

We listen!
We will gladly accept recommendations from anyone, and we will do our best to provide functionality as requested. If you have any thoughts or modifications for IFM, please contact us at `support@digitalindigo.com` or visit us on the web at `http://www.digitalindigo.com`.

Some Features Of IFM…

Multi-part MIME e-mails. This allows newer HTML-based e-mail clients to see forms more clearly. But we maintain backward-compatibility with older text-only clients.

Redirection settings. Allows IFM to send to smart forms. Redirection can occur because of an error, or successful completion of a form.

Base-64 Decoding. Users can wrap binary data or complex data into a Base-64 encoded variable. Simply specify which variables are Base-64 on your form, and IFM will decode them and display appropriately.

HTML Submission. Users can submit HTML in variables and have that HTML show up in their HTML-capable e-mail clients.

Templates. User can specify both text and HTML-based templates.

File attachment. Enables clients to attach files via a web form. (We don't recommend this feature. But it is available!)

S/MIME encryption. Allows secure transport from form to e-mail. Also keeps prying eyes (even those with access to the mail server) from seeing confidential information from forms.

Special Variables
All Special Variables must be in lowercase, with the exception of those listed in the *Non-Case Sensitive* Variables section. Special variables are form fields (usually hidden fields) that control various functionality of IFM. A list of special variables and their function follows.

Non-Case Sensitive Variables
Certain variables are accepted, regardless of case, by IFM. These variables are: recipient, subject, realname, and email. Therefore you can have either of the following lines to denote the recipient of the form:

```
<input type="hidden" name="subject" value="Form Results">
```

or

```
<input type="hidden" name="Subject" value="Form Results">
```

enable_underscore	[0,1]	Normally IFM will display your variable names, replacing any underscores with a space. Thus a variable on the form called "Last_Name" will show up in your e-mail as "Last Name". Setting enable_underscore to "1" will result in the variable be displayed as "Last_Name" in your email.
required	[CSV]	When filling out a form, you may often wish to make certain fields required. For instance, you may want to have fields called "first_name" and "last_name" on your form. To require the user to fill out these fields, you would have a line similar to the following: <input type="hidden" name="required" value="first_name,last_name">
sort	[rev,alpha]	When IFM displays the variable from your form in e-mail, it displays them in the order they occurred on your form. However, you have the option of sorting them alphabetically or reverse-alphabetically using the sort option.
redirect	[uri/url]	After a user has successfully completed a form and pressed the submit button, IFM will display a simple page of the data they entered. As an alternative you can skip this page and direct the user to a page that you created using the redirect option.
redirect_values	[0,1]	To create customized pages, you may wish to include information entered on the form in the resulting HTML after a user submits the form. You can redirect the values from the form to your page specified in the redirect option. By enabling the redirect_values option, your script will be called with form fields appended as a GET. If a user puts "Mike" in the first_name field of the form, and you set your redirect value to "success.php", IFM will redirect to the URI success.php?first_name=Mike. Values will also be redirected on error. This permits the creation of a "smart form" that will fill in entered values, and display fields that have errors. Security! Since data (even data submitted by a post) is sent back via GET, data that could be confidential may be left in your server's logs. Be sure to take this into account when using this feature.

no_redirect	[CSV]	Sometimes forms contain private data, such as a credit card number. In this case you DO NOT want to redirect these values when using the redirect_values option. Since redirect_values passes the data using a GET (the form variables are appended to the URL), values can be logged by the web server and potentially displayed where they shouldn't be. To prevent this we might use a line like the following: <input type="hidden" name="hidden_redirect" value="cc_number">
redirect_errors	[0,1]	Some developers will want to make smart forms. For instance, if a user forgets to fill in their first name, and it's required, an error will be generated. By using redirect_errors, this error will be directed back to the form. For instance, if the field name is called first_name, and it is left blank, the initial form will be called and data will be passed on the URL as follows: someform.php?X_failed_first_name=1 Hint: You will most likely want to use the redirect_values option with this feature.
error_redirect	[uri/url]	By default, when using the redirect_errors option, IFM will redirect the errors back to the referrer; the initial form page. However, using error_redirect, you can specify a different URI or URL to go to if a form error occurs.
print_blank_fields	[0,1]	By default IFM will only list variables in your e-mail that were filled out. For instance, if you have a form field called Work_Phone and it is left empty, it will not be displayed in your e-mail. Enabling print_blank_fields changes this. Thus in your e-mail you will have a line that looks like the following: Work Phone:
hidden_fields	[CSV]	Sometimes you may have a field on a form that you do not wish to show in the e-mail. By placing that variable in the hidden_fields option, you will not see its placement in your e-mail.
hide_html_var_names	[0,1]	IFM allows your form to have HTML-based fields, as long as the field name ends in _html. Some times you may not want to show the fieldname in which the HTML was entered. To turn of the fieldname in e-mail set this option to 1.
uu_decode	[CSV]	IFM allows you to submit fields that are Base 64 encoded. This may be popular during forms that span multiple web pages and contain large text areas. The text areas can be Base 64 encoded on successive pages as data is passed down the line. You can list all Base 64 encoded fields in the uu_decode option, and IFM will decode these fields appropriately.

ifm_smime_enable	[1]	IFM can encrypt outgoing e-mails according to the S/MIME format. This can be popular with forms that may collect e-commerce data or private personal information. To enable S/MIME encryption, set this value to "1". Be sure to have include a valid certificate in the ifm_smime_cert field, or your e-mail will be blank!
ifm_smime_cert		This is an ASCII version of your public key from your Digital ID / Digital Certificate. To read an exported PFX file use the following command. Use a text editor to extrat the certificate from the chain.txt file. openssl pkcs12 -in testcert.pfx –out chain.txt
redirect_values_on_success_as_post	[0,1]	If form is successfully filled out, create a form with a button only that says "Continue". This allows values to be redirected as a POST.
redirect_values_on_error_as_post	[0,1]	If form is not successfully filled out, create a form with a list of errors, as well as a "Return To Form" button. This allows values to be redirected to the original script a POST.
html_redirect	[url/uri]	Creates an HTML page that displays "To continue, click here."
form_relative_redirect	[0,1]	If enabled, redirects occur relative to the location of the form, rather than the location of the IFM script. This feature is moot if the form and script are located in the same directory.
ifm_send_form	[URL]	Fetches a page via POST (of all incoming variables from form) and sends the HTML in the body of the e-mail. This allows smart forms to e mailed directly to the recipient.

Sending Files
Allowing people to post files to your e-mail from a web interface is NOT recommended. However, under certain circumstances, some people wish to use this feature, such as intranets and forms that trusted users use. *We explicitly suggest you DO NOT use this feature!*

The most important part of a form capable of sending files is the <form> tag itself. The tag **must** contain the text `enctype="multipart/form-data"`. Failure to do this will result in a form that processes all variables, but does not accept files. Below is a complete form capable of sending files. Be sure to note the construction of the <form> tag!

```
<form enctype="multipart/form-data" action="/sys-php/iformmail.php" method="post">
  <input type="hidden" name="MAX_FILE_SIZE" value="1000000">
  <input type="hidden" name="sort" value="">
  <input type="hidden" name="required" value="first,last">
  <input type="hidden" name="recipient" value="someone@test.com">
  Name <input type="text" name="realname"><br>
  Email <input type="text" name="email"><br>
  First <input type="text" name="first name"><br>
  Last <input type="text" name="last"><br>
  Middle <input type="text" name="middle"><br>
  Send this file: <input name="userfile" type="file">
  <input type="submit" value="Send File">
</form>
```

Security! It's important to realize that the MAX_FILE_SIZE variable is not necessary, and can easily be tampered. PHP does provide internal restrictions on the sizes of submitted files. (In the php.ini file under the variable upload_max_filesize). The point is, do not rely on the MAX_FILE_SIZE variable to limit submissions to IFM.

Troubleshooting! If IFM does not accept files, it is most likely a result of not having `enctype="multipart/form-data"` in the <form> tag. Another possibility is that `file_uploads` is turned off in the php.ini file.

If a user includes a large file for submission, they may experience a long wait after clicking the Submit button on your form. This is a result of how HTTP works. The file must be uploaded to the iformmail.php before iformmail.php can return any data. Advanced web designers may wish to design a form that uses animated technology to provide user feedback should the user submit a large file.

UBE Guard

Indigo Form Mail has various means by which to prevent itself from being used to deliver Unsolicited Bulk E-mail (often called spam). By default IFM has this feature enabled. This means if a specific IP address accesses the IFM script more than seven time in a given hour, the user will not be allowed to use the script.

You can control the UBE Guard only by directly editing the iformmail.php script. The UBE guard is controlled by the following variables in the script:

```
$uce_db_enable = 1;
$uce_db_file = "/tmp/.iformmail_uce_log.db";
$uce_db_secs = 3600;
$uce_db_threshold = 7;
```

`uce_db_enable` – This variable turns on the UBE guard. To enable set to "1" to disable, set to "0". We strongly recommend keeping this feature on. However, you may want to turn it off at certain times, for instance when testing out a new form, in which case you may be submitting it to the iformmail.php script several times in a short time period.

`uce_db_file` – This is the actual database file used by IFM. This file contains the IP address of clients who used IFM, as well as the times they accessed script. By default it is set to /tmp/.iformmail_uce_log.db. This may cause problems for shared hosting setups or users on MS Operating Systems.

`uce_db_secs` – This is the time in seconds that IFM will store a clients IP address and script access time.

`uce_db_threshold` – This limits the amount of times a client can access the IFM script in the period of time specified by uce_db_secs. If the client exceeds this their access to IFM will be temporarily disabled.

Security! In most setups the DB file will be owned by user "apache". This means anyone on the same server (for example a shared hosting server) can write a script that could potentially corrupt the database. To prevent this, you may wish to put the file in a more secure location, such as your home directory. **Do NOT put the database file in your web tree!!!**

Troubleshooting! Using default settings on a shared server environment, IFM will log all accesses across all websites to a central DB file. This may be used to find general abusers. It may also prevent legitimate users from accessing the form submission if they are submitting to various sites on the same shared server. If this is the case, we recommend giving each site its own iformmail.php file with a unique uce_db_file location for each site.

Viewing the UBE Guard Database
You can also view an HTML version of your current iFormMail access database. To do so, you must enable access by editing the iformmail.php script. Find the line that says:
`$allow_client_db_dump = 0; //DO NOT ENABLE/SECURITY REASONS. TEST PURPOSES ONLY!`
Change the "0" to a "1" and save the file.

Next access your iformmail.php script directly. Be sure to put `?show_db_dump=1` after the filename.

For example, if your iformmail.php script is located at
`http://www.somehost.com/sys-php/iformmail.php`
you would access the URL
`http://www.somehost.com/sys-php/iformmail.php?show_db_dump=1`

Security! It is recommended to use this option only during debugging and/or troubleshooting. Use this option responsibly. By enabling this feature, you also allow outsider users to peruse your database of recent clients. This can give someone an idea of who's accessing your forms, where they're coming from, or how many people use the script in a given period of time. Also never put your uce_db_file in your web tree.

Multiple Fields with the Same Name
In certain cases a user may wish to have multiple fields with the same variable name. This is typically used for checkboxes, such as the case below:

```
Please contact me via the following methods: <br>
<input type="checkbox" name="Contact_By[]" value="Phone"> Phone<br>
<input type="checkbox" name="Contact_By[]" value="E-mail"> E-mail<br>
<input type="checkbox" name="Contact_By[]" value="US Mail"> US Mail<br>
```

Brackets must also be used in the variable names assigned to multiple select boxes, too:

```
Please select your toppings. You may SHIFT-Click to select multiple toppings: <br>
<select name="Toppings[]" size="3" multiple>
  <option value="Ketchup">Ketchup</option>
  <option value="Mustard">Mustard</option>
  <option value="Mayo">Mayo</option>
</select>
```

When assigning the same name to multiple fields, it is important to add [] after the field name. The brackets indicate to iFormMail that it should take all fields and combine them together with a comma separated list. The brackets will not be printed in the recipients e-mail.

In the example above, if the user would check the boxes before Phone and US Mail, the resulting e-mail would have a line as follows:

```
Contact By: Phone, US Mail
```

Troubleshooting! IFM needs indication of multiple-select options. This indication is done by placing the characters "[]" (without quotes) after the variable name. The [] characters will not show up in the variable name when IFM displays it, however it will indicate to IFM that you may have multiple options checked with the same name.

S/MIME Email Envelope Encryption

In some cases IFM may be used to collect private information about a person. This may be something in e-commerce, such as a credit card number. Or it might be private records containing a persons Social Security number or other confidential information. In such cases it's important to keep data out of the hands of those who aren't permitted to access it. This is where S/MIME is used.

S/MIME is a setup that comes installed on various e-mail programs. It takes an e-mail and encrypts (scrambles) all the data except the sender, recipient, date, and subject. The encryption is done using public key encryption. That means a public key is used to scramble the data. However the data cannot be descrambled by that key. Only a *private key* is capable of descrambling the data.

IFM allows you to put the public key inside the form you create. This key is then used to scramble (lock) the form information once it is submitted. Even though people on the Internet can see the public key, it is useless to them since it is not capable of "unlocking" the encrypted e-mail. The e-mail then rests on a mail server until pickup. Even then no one can see what's in the e-mail (not even system administrators) unless they have the unlock key.

The unlock key is called a Digital ID, and you can get one from a Certificate Authority. Once this is loaded into an e-mail client, the e-mail client has the capability to "unlock" the scrambled e-mail. This becomes as easy as opening the e-mail.

There are various ways to achive email encryption. IFM uses S/MIME because of its popular support in e-mail clients. The examples given below deal with the process of using MS Outlook to access S/MIME encrypted e-mails. It is assumed that the reader has a decent knowledge of security, public key encryption, Outlook, and Linux, as well as access to the aforementioned.

Getting Your Public Key

To use S/MIME, you must first aquire a Digital ID. This can be done by going into Outlook and choosing: Tools -> Options -> Security (Tab) -> Get A Digital ID… (Button)

After clicking the Get A Digital ID… button your web browser will launch and take you a web page listing various companies that are authorities for Digital IDs (also called Digital Certificates). Choose a company and follow the directions to get, and install your Digital ID.

Extracting Your Public Key Using Internet Explorer

After you have your Digital ID installed, you will need to extract your Public Key for use in IFM. For security measures it is IMPORTANT to only export your Public Key. DO NOT EXPORT YOUR PRIVATE KEY!!!

Different programs allow you to save your Private Key. We will use Internet Explorer for our example. First you must install a security certificate on your system. Following issuer's directions to do this. To extract your key, navigate though IE's menus as follows.

1. Tools -> Internet Options -> Content (Tab) -> Certificates (Button)
2. Select the certificate you wish to use. Then click the *Export* button. Your next screen will be the Certificate Wizard.
3. On the Certificate Wizard screen click the *Next* button.
4. Choose the radio button in front of *No, do not export the private key*. It is ***EXTREMELY IMPORTANT*** to make sure this is selected. DO NOT highlight the *Yes* button. Now click the *Next* button.
5. Next highlight the radio button in front of *Base64 encoded X.509 (.CER).* Click the *Next* button.
6. Choose a location and a file name for your public key. For this example we will call it *mykey.cer*.
7. Click the *Finish* button.

Extracting Your Digital ID (Not necessary, advanced users ONLY!)

IMPORTANT!!! This section can (and should) be skipped by most users. Unless you know what you are doing with Digital IDs and have a specific reason to manipulate your Digital ID, you should NOT use this section. Refer to the section ***Extracting Your Public Key Using Internet Explorer*** *for instructions on how to get your Public Key for use with IFM. This section is provided only as a supplement for users who wish to perform advanced extraction from their Digital IDs.*

In this example we'll extract our entire Digital ID from Outlook. Keep in mind that this includes your Private Key! Always guard your private key, and files that contain it. Working with files that contain a Private Key (such as is covered below) is NOT recommended unless you know what you're are doing. Most users should never need to use this section.

For security reasons, your Digital ID file (in this case digitalid.pfx) should not be transmitted by insecure means (such as FTP). Access to it, such as typing the password for decoding into openssl should not be done using insecure means (such as telnet). The files should not be manipulated on a server with insecure memory (such as a shared hosting server). And finally the output file (in this example chain.txt) should NEVER be left for anyone to see. It should not be stored on any public system, or on media available to anyone but you. It should not be transferred over networks. And for that matter it probably shouldn't be unwrapped in the first place! Obviously not all security concerns have been covered. Keep in mind that the output file contains your raw Private Key. It should be treated and handled with thorough security considerations—or better yet it should not be handled at all!

After your ID is installed go back to Outlook and choose:
Tools -> Options -> Security (Tab) -> Import/Export Digital ID… (Button)

Next click the radio button aside of "Export your Digital ID to a file…"
Use the Select button to select your newly installed certificate.
Click on the Browse button to choose where to save your Digital ID. (for example digitalid.pfx)
Enter the password for your certificate.

Next you must transfer your Digital ID file to a Linux server or system with OpenSSL installed.

Run the following command to extract your Digital ID into a ASCII chain file:

```
openssl pkcs12 –in digitalid.pfx –out.chain.txt
```

Now use an editor to look through the file chain.txt. Copy the public key. (Typically the public key will be below a line that has Cert/Email=you@somewhere.com in it.) Your copy should look similar to the following *(Do NOT copy anything before -----BEGIN CERTIFICATE----- or anything after -----END CERTIFICATE----):*

```
-----BEGIN CERTIFICATE-----
fJsdJ42xowRjfM34 blah, blah, blah
-----END CERTIFICATE----
```

Congratulations! You now have your public key certificate. This will be the key that you plug into IFM so that it can encrypt your data. Remember after it is encrypted, the e-mail cannot be read by any client without the Digital ID installed. There is no way to read your e-mail if you only have the public key certificate.

IMPORTANT! Guard the chain.txt file and anything else that might have your private key accessible. Delete them off your Linux system when you have extracted your public key.

Setting Up Your Form To Encrypt Email

Once you have your public key, you're ready to encrypt e-mail using S/MIME. To do this you must tell IFM to enable encryption. This is done by placing a hidden tag in your form. The tag should look like the following:

```
<input type="hidden" name="ifm_smime_enable" value="1">
```

Note: If you wish to temporarily turn off encryption during the test phase of your form, simply edit it so the ifm_smime_enable is set to zero.

Next you will need to include a tag that contains your key. The tag will look something like this:

```
<input type="hidden" name="ifm_smime_cert" value="_your_certifcate_here_">
```

Next you will have to send you public key to IFM. This can be done in two different ways. The first is to include your certificate directly into your form. This is not very clean, but will suffice. The other option is to store the public certificate in a file and use PHP to include the certificate in the form.

Example 1

First we'll take a look at the setup of form using the first method described. This is a plain HTML file. Be sure to pay close attention to formatting. Notice how the certificate is pasted in-between the quotes of the value field of the ifm_smime_cert field.

```
<form action="iformmail.php" method="post" enctype="multipart/form-data">
<input type="hidden" name="recipient" value="test@somewhere.com">
<input type="hidden" name="ifm_smime_enable" value="1">
<input type="hidden" name="ifm_smime_cert" value="
-----BEGIN CERTIFICATE-----
MIIEXDCCA8WgAwIBAgIQd63XhupYv2b8K481atxIwzANBgkqhkiG9w0BAQQFADCB
zDEXMBUGA1UEChMOVmVyaVNpZ24sIEluYy4xHzAdBgNVBAsTFlZlcmlTaWduIFRy
cDQm2wr51r8orWKfV/F0JQ==
-----END CERTIFICATE-----
">
<br>
Your Email Address: <input type="text" name="email"><br>
Your Comments: <input type="text" name="Comment"><br>
<input type="submit">
</form>
```

Example 2

With the following example we'll take our public key certificate and paste it into a file called cert.txt. We will then rely on a PHP call to include the file automatically in the form. Remember, the following code has some PHP calls in it, therefore you'll need to end your file with a .php extension.

```
<form action="iformmail.php" method="post" enctype="multipart/form-data">
<input type="hidden" name="recipient" value="test@somewhere.com">
<input type="hidden" name="ifm_smime_enable" value="1">
<input type="hidden" name="ifm_smime_cert"
       value="<?echo implode("", file("cert.txt"));?>">
<br>
Your Email Address: <input type="text" name="email"><br>
Your Comments: <input type="text" name="Comment"><br>
<input type="submit">
</form>
```

Frequently Asked Questions

P: I'm trying to send a file, and the variables from my form come through fine. But the file is never attached to the e-mail.

S: You probably forgot to put the "enctype" in the <form> tag. Be sure your form tag looks something like the following:

P: When someone checks multiple checkboxes, I only receive one value in the e-mail.

S: Your probably forgot the [] tag at the end of your field name. The [] after a field name indicates that multiple values should be grouped under one field name.

P: I'm not receiving the results of a form in my e-mail box.

S: First check to see if you have a "recipient" tag in your form. If you don't, iFormMail will attempt to use the e-mail address that the web server has on file. This may not be your e-mail address, and the script may be sending the e-mail to the wrong address.

P. I receive the message "`ERROR: Could not open DBA.`" when accessing IFM.

S. This may be that you do not have access to the location specified inside of iformmail.php under the uce_db_file. It may also mean that the file is owned by some user other than the user the web server is running IFM as. Your best solution is to change the location of the uce_db_file in the iformmail.php script. Be sure the location you specify is writable by user "apache" or whatever user your web server runs the IFM script as.

P. I receive the message "`Sending to multiple recipients is not enabled.`" when accessing IFM.

S. In the "recipients" field of your form, you specified more than one e-mail address. By default, this is disabled. This prevents spammers from putting in hundreds of recipients and then lauching junk mail to them using IFM. To allow multiple recipients, you must edit the iformmail.php script and set `enable_mr` to "1".

P. I submit files to iformmail.php but it totally ignores them.

S1. This is most likely one of two problems. The first may be the setting in your /etc/php.ini file. Be sure there is the following line: `file_uploads = On`. On some distributions of PHP the default is file_uploads = Off. You need system administrator access to change this. If you are not the system administrator, contact the appropriate person to fix this. Be sure to restart Apache (or whatever webserver you're running) after making this change.

S2. The other problem may be in the design of your form. It is imperative that you specific an encoding type in the <form> tag of your form. For example the following line will **NOT** work:
```
<form action="/sys-php/iformmail.php" method="post">
```
However, the following line will allow you to successfully submit files:
```
<form enctype="multipart/form-data" action="/sys-php/iformmail.php" method="post">
```