Frequently Asked Questions

Table of Contents

Table of Contents

1. How does Interchange work?

1.1. Where are the pages?

Interchange pages are not kept in normal HTML space. Look in the catalog subdirectory pages. The pages are always filtered through the Interchange daemon before being delivered.

1.2. Where are the images?

Interchange is a CGI program, and if relative image paths are used, IMG tags like the following will occur:

<IMG SRC="/cgi−bin/simple/../whatever.jpg">

Interchange, by default, uses an `ImageDir` for a prefix. In the demo, image specs that have no absolute path information are prefixed with `/simple/images/`.

In an Interchange page, this tag:

```
<IMG SRC="ordernow.gif">
```

will become this:

```
<IMG SRC="/simple/images/ordernow.gif">
```

This tag:

```
<IMG SRC="items/00-0011.jpg">
```

will become this:

```
<IMG SRC="/simple/images/items/00-0011.jpg">
```

Absolute image paths are not affected. An image such as `/other/images/whatever.gif` will not be changed.

1. How does Interchange work?

2. INSTALLATION

2.1. Configuration Problems

Most Interchange configuration and setup problems are due to one of the following:

Wrong information given to makecat program.

This is by far the most common problem. To install a working demo, Interchange needs to know what the `DocumentRoot` is and how to run CGI programs. Details of this setup are server– and site–specific, which may require some research.
Re–run the configuration again, and pay close attention to the prompts given. There are examples given which apply to most systems.
If the web server is Apache or NCSA, Interchange will try and parse its `httpd.conf` file to help you along, but many ISPs don't allow users to read these and it may fail.

Too–low version of Perl.

If you have a Perl earlier than 5.005, Interchange will not work. Don't even try an earlier version.

Perl compiled with USE_THREADS.

Run `perl -V`. If you see –DUSE_THREADS in the compilation definition, you might run into problems with Interchange.
NOTE: You cannot run the Interchange software as `root`.

If you are setting Interchange up for the entire machine, and not just as a virtual host user, it is usual to create a special `interch` user to run the daemon and the link program. This means the directory listing for your cgi–bin should be something like:

```
-rwsr-xr-x   1 interchange users          6312 Dec 30 11:39 cgi-bin/simple
```

and for the socket file it should be:

```
srw-------   1 interchange users             0 Dec 30 11:41 etc/socket
```

Once you have set up the software, you can easily install catalogs as root **as long as your `umask` is set to 2 or 22**.

(The following assumes you have made the Interchange software owned and run by the special user `interchange` and that each user has a Interchange catalogs directory `/home/user/catalogs`).

The best way to set permissions on a multi–user system is to make all files group readable and writable (660 or 664 mode). If you have a system setup that places each user in their own group, make `interchange` a member of each user's group and set ownership and permissions with:

```
        find /home/user/catalogs -print | xargs chown user
        find /home/user/catalogs -print | xargs chgrp user
        find /home/user/catalogs -print | xargs chmod g+rw
```

For best results, set the user's default umask to 2, so that they will, by default, create files that have the proper permissions. If you have all users in the same group, the above is not secure. You should put `interchange` in a group of which no user is a member (perhaps interchange would be a good choice) and set all files owned by the group interchange and all directories to mode 2770:

This will make files default to the proper group when created (on most UNIX versions, anyway).

```
find /home/user/catalogs -print | xargs chown user
find /home/user/catalogs -print | xargs chgrp interchange
find /home/user/catalogs -print | xargs chmod g+rw
find /home/user/catalogs -type d -print | xargs chmod g+s
```

If you are on a virtual hosting system, the procedure varies. Making the program setuid should work for most systems. If your setup uses CGI–WRAP or another setuid scheme, it should still work. However, you may have to unset the setuid bit with `chmod u-s cgi-bin/simple` or the like. If you have a non–standard CGI setup, as some virtual host systems do, you will need to know something about UNIX and the web or engage a consultant to properly set up the paths. Usually switching to TLINK/INET mode is the easiest thing to do, though with Iserver and a few others it may take more than that.

If you used the `makecat` program to build the catalog, it should have warned you if it was not able to make the link program setuid. To set the program (in the file `cgi-bin/simple` in this example) to be setuid, use the command:

```
chmod u+s cgi-bin/simple
```

2.2. Error –– the Interchange server was not running...

This indicates that the link CGI is not communicating with the Interchange server. **Important note:** The server should always be started by the same user ID which owns the suid vlink program. (This does not apply to TLINK/INET mode.)

The server must be running, first of all. If you didn't start it, you can do so by going to the Interchange home directory and typing:

```
bin/interchange -restart
```

You can check to see if your server is running by typing:

```
Linux, BSD:         ps -ax | grep interchange
Most other systems:  ps -elf | grep interchange
```

Note: Solaris and IRIX truncate the string, and don't allow setting of the $0 parameter. You may have to grep for 'perl' instead.

If the server is not running, it may have failed due to another process occupying the TCP socket 7786. If using VLINK, try starting Interchange with `start -u`, which will not monitor the internet–domain socket.

If VLINK is not communicating with the server, there are a number of possible reasons. First, if you are trying to run Interchange on an ISP, go to the section about ISP problems. It is probably one of those. If you are running Interchange on a single machine, it is probably one of:

```
1. Permissions problems
```

```
    2. Interchange on NFS-mounted file system
```

Check the error_log file for your HTTP server —— it will almost always tell you what the problem is, unless there is a simple permissions problem.

Permissions are easy. If starting Interchange like this works:

```
        interchange -r SocketPerms=666
```

then you have a socket permission problem. Try restarting interchange without the above adjustment of SocketPerms=666, and then try accessing it again with each of these mode changes:

```
        chmod u+s cgi-bin/storename
        chmod u-s cgi-bin/storename

          cgi-bin/storename = path to your executable
```

If neither of those work, either the UID the program is owned by is wrong, or your HTTP server is interfering in some fashion. If you are running Interchange on an NFS−mounted file system, it cannot run in server mode because UNIX−domain sockets don't work on NFS. You will need to change to static mode from server mode, or better yet, put Interchange on a file system that is directly mounted.

You can use Interchange in INET mode along with the tlink.c program to allow running across NFS boundaries. If you have not changed the configured defaults, and still it will not communicate, you should try setting the LINK_HOST and LINK_PORT directives in tlink.c and recompiling.

2.3. I get messages like 'Config.pm not found.' What does it mean?

This means your Perl is not properly installed, or that Interchange is not using the proper Perl binary. On UNIX, try reinstalling Interchange and using the standard Perl installation sequence:

```
  /complete/path/to/proper/perl Makefile.PL
  make
  make test
  make install
```

Otherwise, contact your system administrator.

2.4. Can't locate lib.pm in @INC. BEGIN failed−−compilation aborted.

Again, your Perl is not properly installed. Someone has put a Perl up on your system, then either moved or removed the library directory. Contact your system administrator and request that Perl be re−installed.

2.5. Segmentation fault or other core dump.

If this happens when you run the Interchange test or server, it is always Perl that has a problem. Not sometimes, always. A proper Perl should never have a segmentation violation, period. And it should not dump core (unless you passed it a −u option somehow).

You will need to either update Perl or report the bug to the proper personnel. Depending on your situation and technical ability, this may be your system admin, ISP, or the Perl porters.

2.6. Configuring catalog whatever...Use of uninitialized value at Config.pm line 1614, <CONFIG> chunk 322.

This is a warning from Perl indicating that an empty value was found where one is expected. The warning is left in so that you know that something is missing. Whatever it is, it can be found at the specified "chunk," or line, of catalog.cfg. If you use the include capability, it would have to be factored in as well.

The usual reason is that a file is specified in one of the directives (usually one of Help, SearchProfile, OrderProfile, Buttonbars, or UpsZoneFile) and does not exist. See the documentation for the directive on how the file name should be specified.

2.7. Why isn't the above error more enlightening?

Because Perl won't tell us what exactly went wrong. See its FAQ for why.

2.8. XXXXXX.pm does not match executable version.

This is a Perl which does not have the right Perl library installed. It usually results from a naive system administrator who thinks they can bypass the 'make install' for Perl and just copy the Perl binary or directories.

If you installed Bundle::Interchange locally in your Interchange directory, it may mean that your system administrator updated Perl and failed to select the binary compatibility option.

2.9. Can I run Interchange on the Macintosh or Windows?

Interchange will not run on a MacOS 7, 8, or 9 operating system. It will run on Mac OS X and other PowerPC Unix variants.

Interchange's *files* can be manipulated by any computer. As long as uploads/downloads of database source, pages, and configuration files are done in ASCII mode, there is no reason why they can't be edited on a Mac. And with MySQL or other ODBC databases on your UNIX–based ISP, you can even directly interface to the database you use with Interchange provided you have the scarce ODBC middleware needed for the Mac.

Interchange can be run on Windows with the Cygwin tool set (1.3.2 or higher) available from Red Hat, but there are numerous anomalies and it may be difficult to get operating reliably. It is never recommended that you run a production catalog on a Windows system; if you do get it working you should only use for catalog development.

3. SSL problems

3.1. Shopping cart is dropped when using SSL.

If you are using a separate secure and non−secure domain, this is due to the cookies from the user not matching as well as the session ID not being able to be transferred due to differing source IP addresses.

NOTE: Interchange does not support this configuration. You may be able to get it to work in some circumstances, but it is not supported. See the next set of questions for help on how you may be able to get it to work to some extent in your configuration. It will not work in every circumstance with every feature.

This is sometimes due to the HostnameLookups (Stronghold/Apache parameter) not matching for the two servers, secure and non−secure. It can also be caused by the user having different web proxy addresses for HTTP and HTTPS. If it still does not work, try changing some of the appropriate configuration parameters in interchange.cfg:

```
DomainTail   No
IpHead       Yes
```

If you still are having problems, try this combination in catalog.cfg, the catalog configuration file:

```
SessionExpire  10 minutes
WideOpen       Yes
```

The above setting will typically make Interchange work when it is possible to work. Sometimes when you have multiple Interchange servers sharing the same secure server, you will have problems after accessing the second one. (The first one issues a session ID cookie, and that causes problems).

3.2. I have a different secure server domain. Why does the shopping cart get dropped?

First of all, it is questionable business practice to not certify your secure server. Besides violating the terms of use of many certificate issuers, customers notice the changed domain and it is proven by user surveys and long experience that you will receive fewer orders as a result. Certs can be obtained for $125 US per year, less than the typical cost of one hour of a top consultant's time. Do your business a favor −− spend the money to get a cert.

If you insist on doing it anyway, probably driven by the fact that you need a dedicated IP address for a secure server, you can use the solutions in the previous FAQ question and get some relief.

But by far the best way is to have all orders and shopping cart calls go only to the secure domain. Your users may get a different session when browsing the non−secure catalog pages, but it will matter little.

To do this on the Foundation demo, place in catalog.cfg:

AlwaysSecure order ord/basket ord/checkout

A more complete list might be:

```
AlwaysSecure <<EOF
        account
        change_password
        customerservice
        login
        logout
        new_account
        ord/basket
        ord/checkout
        order
        process
        query/check_orders
        query/order_detail
        query/order_return
        returns
        saved_carts
        ship_addresses
EOF
```

Add pages of your own that need to be sure of coherent session information.

For all *forms* to be secure, make sure "process" is on that list. (Your search forms will still be non−secure if you use "[process−search]" to produce the form ACTION.)

To make individual order links secure, use this instead of "[order]":

<A HREF="[area href=order secure=1 form='mv_order_item=SKU_OF_ITEM']">Order it</A>

To make a form−based order button secure, use "[process secure=1]" as the ACTION.

3.3. My images aren't there on the secure server!!!

You have a different document root, or the permissions are not such that you can access them. You can set a different base URL for images with:

```
        ImageDirSecure    https://your.secure.server/somewhere/images
```

Don't try to set it to an http:// URL −− images will be broken anyway.

3.4. My secure pages fail when the browser is MSIE.

MSIE has several SSL bugs, particularly in V5.01. See the `Apache-SSL` or `mod_ssl` FAQ. You can sometimes fix this with an httpd.conf change:

SetEnvIf User−Agent ".*MSIE.*" nokeepalive ssl−unclean−shutdown

4. ISP Problems

The great majority of ISPs provide some CGI service, and more and more run systems that are compatible with Interchange. The new catalog configurator for Interchange makes setup much easier. A word of warning: if you chose your ISP mostly on price, you can expect problems. The low–cost providers typically have heavily–loaded machines and many domains. The more domains and the more load the unhappier you will be with Interchange. Interchange works best on a fast machine with plenty of memory.

A few Internet Service Provider (ISP) systems still have difficulty with one or the other aspect of running Interchange. A few cannot (or will not) run Interchange at all. On top of that, many times ISP personnel are too busy to help, won't help, or don't know enough to help. Some are secretive about details of the setup of their systems.

All in all, you can have a fair amount of confidence that your ISP can run Interchange. Or, you can get one who will. 8–)

4.1. No shell access allowed on my ISP.

Generally it is a waste of time to try to use Interchange without shell access.

4.2. We're sorry, the Interchange server is unavailable...

(The following assumes that you were able to start the Interchange server.)

This could be almost anything, but with a properly configured Interchange it is almost undoubtedly due to your cgi–bin and/or your Interchange directory being located on a different filesystem than the actua machine that is executing the program. VLINK uses UNIX–domain sockets, which don't work on NFS–mounted filesystems.

Iserver.com and other systems which use chroot HTTP servers require quite a bit of extra configuration to get going. If you have not been careful to set permissions properly when running in VLINK/UNIX mode, the link CGI will not be able to communicate with the Interchange server. Please read the documentation that covers this in detail.

You can run in INET mode with the `tlink` link program to prevent those problems.

4.3. Document contains no data or premature end of script headers (especially on BSDI or FreeBSD).

This usually means that your HTTP server ran out of resources during the execution of the link program. It couldn't create more sockets, is unable to create a process, or can't open any more files.

This usually happens in frames catalogs, when Interchange is sending more than one page simultaneously. And even more especially on FreeBSD and BSDI, which are often distributed with the kernel parameters SOMAXCONN and CHILD_MAX set to levels unsuitable for serving the web.

Go to <http://www.deja.com> and try searching for MAXUSERS. This should give you plenty of pointers on how to set these parameters properly.

4.4. Interchange server only runs for a while, then dies.

Many ISPs don't allow your user ID to run a program unless it is logged in! The moment a watchdog program notices a daemon running with a non−logged−in UID, it terminates the program. Or, it terminates programs that haven't been active for XX minutes. Contact your ISP about this. They may be able to do something for you.

4.5. My entire home directory is in HTML document space.

If working with an ISP where all of the files are in HTML document space, disable all access to the Interchange catalog directory with the proper HTTP access restrictions. Normally that is done by creating a .htaccess file like this:

```
<Limit GET POST>
order allow,deny
deny from all
</Limit>
```

If unable to do this, do not run Interchange unless file permissions can be set such that files will not be served. However, security will be a problem and customers' personal information could be placed at risk.

5. SYSTEM CONFIGURATION

5.1. Can I run multiple catalogs on one server?

Yes. Interchange supports multiple independent catalogs. There are users who run more than 500 catalogs on a single machine. The capacity is usually a function of how busy the catalogs are and how much memory and processor speed your system has.

5.2. How do I start Interchange when I reboot?

Use the standard facility on your operating system. For BSD−style systems, the file is usually called rc.local (in the /etc directory).

On SVR4 systems, it is quite a bit more complex. Look for the /etc/rc.d directory and see what other programs do. Often the file is called S99startup or something similar.

Important note: Interchange must not run as root, which is the user identity that the startup file executes. (Interchange will refuse to start if executed as root.) The technique to start up depends on the facility of your su(1) command. This should work on most operating systems:

```
su interchange <<EOF
/your/interchange/dir/bin/restart
EOF
```

The **EOF** must be the only thing on the line (no leading or trailing whitespace). If your su(1) command has a −c option (as most System 5 UNIXes do), you can just set:

```
su −c /your/interchange/dir/bin/restart interchange
```

Interchange supplies a `restart` script which tries to do the above portably. It works on many operating systems.

5.3. I installed the Interchange RPM, and I can't restart.

This usually means that you tried to run /usr/lib/interchange/bin/interchange, which fails to take into account the Linux Standard Base (LSB) file setup. Instead, run

/etc/rc.d/init.d/interchange restart

or

/usr/sbin/interchange −r

5.4. How do I set up a mall?

Interchange can share product databases, session files, and any other databases. It has many features which support mall building. You can easily build separate and mostly identical catalogs which you link to via HTML. But building a mall is as much an exercise in data and process as in software. Consider the following

questions:

1. Who will be clearing payment?
2. What happens if everyone doesn't have the same tax rate?
3. How will you clear orders to multiple vendors?
4. How will you bring together multiple types of shipping?
5. How will the vendors get product data (including images) to you?

If you cannot answer those questions and visualize how to build a mall, you probably should not try.

6. PRODUCT OPTIONS

6.1. Can I attach a size or color to a product?

Interchange has product modifiers, or attributes, which can be carried around with the product. Inside an item list or the product page (flypage), the [item−options] tag will automatically place suitable widgets on an HTML form, and "remember" what should be selected. See the Interchange documentation for **Item Options**.

You can use the `SeparateItems` directive or set the `mv_separate_items` variable on the order form to cause ordered items to be put on separate lines in the shopping basket. (This is the default in the demo catalogs.)

6.2. Can I change the price based on size or color (or other attribute)?

Yes. Use the Interchange UI to set up your product options. It operates on the `options` database table to set up options that can effect price. Or see the next question.

6.3. How are simple product options structured?

Interchange has three types of options; *simple*, *matrix*, and *modular*. They are based on the `options` database table.

To enable options for a product, it needs to have a master record in "options" with the SKU as the key. The only fields that matter in the master record are:

```
code        The SKU of the item

o_master    Indicates not a product, but an option for a product in another database

o_enable    Options enabled for that item

o_matrix    Set to 1 for all-in-one widgets, 2 for separate widgets

o_modular   Modular options (alpha)
```

If o_enable is set, but neither o_matrix or o_modular are, the item is using simple options.

For the option itself in simple mode, the following fields apply:

```
code        Arbitrary key
sku         SKU this option applies to
o_group     The attribute name of the option
o_label     The label the widget for the option will bear
o_value     The options, in IC option format
o_widget    The widget type used to display
o_height    The widget height (if any)
o_width     The widget width (if any)
price       Price adjustment
```

Here are the fields for an item with a simple size option:

```
code:os28009
o_master:1
o_enable:1
o_matrix:0
o_modular:0
#
code:os28009-size
sku:os28009
o_group:size
o_label:Size
o_value:S=Small,M=Medium,L=Large,XL=Extra Large
o_widget:select
o_height:
o_width:
price:S=-1.00,XL=1.00
#
```

The price field accepts option modifiers based on the option value; for example, to adjust the price of an S down 1.00 and the price of an XL up 1.00, you use the values shown above. This works in conjunction with the special ==:options atom in CommonAdjust. To activate the pricing adjustment, you must have something like this for your `CommonAdjust` setting:

```
 CommonAdjust    :sale_price, ;:price, ==:options
```

The actual names of the fields used for these can be changed with the Variable MV_OPTION_TABLE_MAP, i.e.

```
Variable MV_OPTION_TABLE_MAP <<EOM
        o_widget   widget
        o_value    value
EOM
```

That would allow you to use "widget" and "value" in place of o_widget and o_value as field names.

6.4. But what do these options do? Where do they live?

If you know Perl, you know what a hash reference is. An Interchange shopping cart consists of an *array of hash references*. If you dump the structure of the `main` shopping cart you would see something like:

```
        [
                {
                        mv_ip       => '0',
                        price_group => 'general',
                        mv_ib       => 'products',
                        code        => 'os28080',
                        quantity    => '1',
                },
                {
                        mv_ip       => '1',
                        price_group => 'general',
                        mv_ib       => 'products',
                        code        => 'os28080',
                        size        => 'L',
                        color       => 'black',
                        quantity    => '1',
                },
        ]
```

Each key of the hash is an attribute. There are a number of special attributes:

code The item SKU sku The SKU of the base item (in the case of matrix options) mv_ip The line number of the shopping cart (minus 1) mv_ib The database table the product was ordered from quantity The number on order group The order group for a master item or subitem mv_si Subitem indicator mv_mi Master item code mv_mp Modular item mv_price Price of the item (to directly set pricing) mv_order_route Special order route for this item

Any attribute besides the above is a product *option* or *modifier*, and can be displayed with `[item-modifier attribute_name]`.

6.4. But what do these options do? Where do they live?

7. ENCRYPTION

7.1. PGP encryption –– Server Error

As always, check the error log. The most common problem is something like:

```
akopia.com 3Ex5lvta:akopia.com – [01/Sep/1997:09:08:43] simple /cgi-bin/simple
> Encryption error:
>
```

Also, check the `ScratchDir` (usually tmp/) for `.err` files; they will contain PGP or GPG's error output.

Probable causes:

Interchange user ID doesn't have keyring

You must have a `.pgp` or `.gnupg` directory in the home directory of the user running Interchange. It is also possible to set an environment variable (variously PGPPATH or GNUPGHOME) to orient the program correctly.

EncryptProgram directive set wrong

In Interchange 4.7.7 and above, you only need specify "gpg", "pgp", or "pgpe". The key is set in `EncryptKey`. If you don't set a value for EncryptProgram, Interchange will look for `gpg` first, then `pgpe`, then finally `pgp`, using the first it finds. If it can't find one of those, it is set to none and encryption can't be done. You can specify a full path to the program in the directive, but no arguments need be set. NOTE: old values in EncryptProgram will still work, just are not needed.

7.2. PGP encryption –– What do I do now that it is working?

This depends on what you do with orders once you receive them by email. Some PC mail agents (notably Eudora) will decrypt the PGP message embedded within the message text. In that case, you can simply embed the `[value mv_credit_card_info]` call right in the message and be done with it.

If your mailer will not decrypt on the fly, the best way to read the credit card number is to set up MIME encoding of the order email. To do this, find the order report you are using. In the standard demos it is `pages/ord/report.html` or `etc/report`.

Set up two MIME regions in that file. First, at the top of the file:

```
[tag mime type TEXT/PLAIN; CHARSET=US-ASCII][/tag]
[tag mime Order Text]

ORDER DATE: [calc]localtime[/calc]
ORDER NUMBER: [value mv_order_number]

Name: [value name]
Company: [value company]

(Rest of order text, including item list)
[/tag]
```

Then, at the bottom of the report.html file, put the credit card info:

```
[if value mv_credit_card_info]
[tag mime type application/pgp-encrypted][/tag]
[tag mime Credit Card Information]

[value mv_credit_card_info]

[/tag]
[/if]
```

Once this is done, you can read mail using your PGP client as a helper application to decode the MIME attachment. This does not require a fancy setup −− you can use the standard MIT PGP 2.6.2 if desired. If you are using UNIX, set up as the helper for the MIME type **application/pgp−encrypted**:

```
xterm -e pgp -m %s
```

More automated or user−friendly setups are left as an exercise for the user.

8. How do I....

8.1. How do I get the number of items in a shopping cart?

If it is simply the total number, extended according to quantity, you can use the [nitems] tag. If you need this number for use in an embedded Perl script, you can use:

```
$number = $Tag->nitems();
```

If it is the number of line items you need, then you can use a Perl script:

```
[perl]
    return scalar @{$Carts->{main}};
[/perl]
```

(The 'main' refers to the main shopping cart.)

If you have SeparateItems in effect, and need the number of unique items, you could use:

```
[perl]
    my $cart = $Carts->{main};
    foreach my $item (@$cart) {
@items = split /\|/, $items;
$count = 0;
for (@items) {
$count++ unless $seen{$_}++;
}
$count;
[/perl]
```

8.2. The demo doesn't do ... (pick one)

That is because it is a demo. It is not intended to be a finished catalog, just a starting point.

8.3. How can I trace the source of a purchase and run a partners program?

Interchange has a facility that adds a parameter called *source* to the session database for that user. You should give your partners a source code, which must contain at least one letter character (A–Za–z only). It is placed in the sourcing URL as a query string of:

```
mv_pc=Source1
```

If this is appended to the URL with which the user calls Interchange, it will then be placed in the session identifier `source`.

This URL:

```
http://yourcatalog.com/cgi-in/yourcat/sp_offer?mv_pc=Source1
```

will yield `Source1` from the Interchange tag [data session source].

The Minivend 3 idiom `?;;Source1` continues to be supported, so existing partner sites should work without change.

8.4. How can I send an email copy of the receipt to a user?

There are several ways, but this is a more complex question than it may seem like it is. You will have to deal with bad email addresses, deciding which information to send, showing delivery times, etc. You also have to be very careful with credit card information. If you have not taken the proper security measures (by enabling PGP credit card encryption or using CyberCash), you might just mail them their own unencrypted credit card number!

This is supported in Interchange via a UserTag, [email ...]. See the "simple" and "basic" demos.

8.5. How do I display Euro pricing?

You can use Interchange's II8N facilty via the Locale directive. In catalog.cfg:

```
# to define the euro-Settings (PriceDivide is for converting from DM)
Locale eur_EUR PriceDivide        1.95583
Locale eur_EUR p_cs_precedes      0
# this is great - you can even use HTML-Tags to display an euro-image
Locale eur_EUR currency_symbol    "<IMG src="/path/to/image/euro.gif">"
Locale eur_EUR p_sep_by_space     2
Locale eur_EUR mon_decimal_point  ,

# and the DM
Locale de_DE
Locale de_DE p_cs_precedes  0
Locale de_DE p_sep_by_space 2
```

Note: Be sure to use the latest exchange rates when you establish your catalog.

On your pages (this is from a search results page, the [item−.... ...] notation may be different depending on your context):

```
[item-price]<br><!-- german is default -->
[setlocale eur_EUR]
  [currency convert=1][item-field price][/currency]<br><!-- the euro -->
[setlocale]
```

Any questions? Read the docs about "Internationalization."

8.6. How do I empty the shopping cart?

Here are three examples of ways to empty/drop/clear the shopping cart contents.

8.6.1. Empty shopping cart

```
[calc]
        @{$Carts->{$CGI->{mv_cartname} || 'main'}} = ();
[/calc]
```

8.6.2. Empty shopping cart

```
[set clear_basket]
        [calc]
                @{$Carts->{$CGI->{mv_cartname} || 'main'}} = ();
        [/calc]
[/set]

[button
        text="Clear Basket"
        src="clear_basket.gif"
        hidetext=1
        form=basket
        ]
                mv_todo=refresh
                mv_click=clear_basket
[/button]
```

8.6.3. Erase the user session (includes shopping cart)

```
[button
        text="Clear Basket"
        src="clear_basket.gif"
        hidetext=1
        form=basket
        ]
                mv_todo=cancel
                mv_nextpage=index
[/button]
```

8.6. How do I empty the shopping cart?

9. Errors

9.1. Sorting doesn't work across multiple pages.

If you are using the [sort table:field] idiom, it cannot. It sorts data present in the list only.

9.2. I am searching for a string and it is not found. I know it is there!

Set `mv_substring_match` to `yes` (`su=yes` in one–clicks). This most commonly happens when searching for non–ISO–8859 (Cyrillic, or characters like umlaut and eacute) characters in word–match mode. The problem is, that unless your locale is set up properly, Perl doesn't think a non–ISO–8859 and a space character is a boundary.

Also, if you are searching for non–alpha characters, they will also not be interpreted as word characters and the boundary problems will still exist.

9. Errors

10. Performance Issues

Interchange is not a lightweight program. If you are running it on a low−end ISP, whose major selling point is low cost, you will frequently find that Interchange performance is very poor. This is due to either:

Not enough memory

If memory is low, the program will "swap" to disk. If lots of swap is used, you can expect very bad performance. This is the most common speed problem.
If your ISP uses IDE hard disks, you can expect REALLY bad performance. IDE disks are very slow for multi−user machines, which should have SCSI if ANY swapping is to be done.

Too many domains

If there is a huge amount of traffic on the system, then it can run at a very high "load average." If the machine's load average is routinely above 2, you can expect problems.

Underpowered machine

If it is an old machine, it may be too slow for Interchange. A Pentium of less than 300MHz is probably not good enough unless it is completely dedicated to Interchange. The faster the processor, the happier you will be.

10.1. Interchange runs, but it's sooo sllooowww...

This is almost certainly due to a system that has inadequate memory or network bandwidth. On a moderately fast ISP server with sufficient memory, pages should start displaying in less than 2 seconds. On a fast server, pages should start loading almost instantaneously.

10.2. Interchange slows down over time.

There are many possible reasons for this, but most have to do with memory or session database size.

- See the documentation on Interchange administration (icadmin) and learn how to expire your session database. If it is megabytes in size, accessing a key will significantly slow down the session.
- Second, if your machine is memory−poor, you will find that Interchange gets swapped to disk. Unless your system is very fast, this will greatly hurt performance. Interchange works best on a machine that rarely if ever swaps to disk.
- Third, this often has nothing to do with Interchange at all but has to do with your HTTP server. Any long−running daemon has the potential for a memory leak. Try stopping and starting your HTTP server and seeing what happens to performance.

10.3. I am using SQL, and Interchange is slow ...

It isn't Interchange. First of all, did you index your 'SKU' or other key fields? The reason Interchange doesn't do it for you is that every SQL database seems to do that a bit differently. Even then, you can try Interchange's COLUMN_DEF parameter:

Database products COLUMN_DEF code=char(16) PRIMARY KEY

This will at least index the `code` field for MySQL. Other databases differ.

Interchange can return VERY fast SQL search results. But you need to at least give it something to work with. The proper method for fast selection is:

```
[query sql="select code,category,title,price from products" ]

Category: [sql-param category]<BR>
Title:    <A HREF="[area [sql-code]]"> [sql-param title] </A><BR>
Price:    <A HREF="[area order [sql-param 0]]"> [sql-param price] </A><BR>

[/sql]
```

This is especially powerful when you consider a joined query like:

```
        SELECT code, price, title, extended.desc
        FROM   products, extended
        WHERE  products.category = 'Renaissance'
```

Note that the extended.desc field will be accessed as [sql–param desc]. Don't forget that you must index your fields if you want fast searching with them as a criteria.

10. Performance Issues

11. Using Interchange with Apache and SUEXEC

Apache with SUEXEC: VLINK/UNIX socket mode will not work well unless installed as a normal user. If supporting multiple users, the TLINK/INET mode must be used.

11. Using Interchange with Apache and SUEXEC

12. A friendly reminder

When in doubt, restart the server. It won't take but a few seconds, and changes in configurable options don't take effect until it is done. You may even change a page and not see the effect until the server is restarted.

13. Tips and tricks

These are slightly edited postings to the Interchange−users mail list made by Mike Heins, lead author of Interchange.

13.1. Locking down your system

Interchange has lots of built−in protections to make developing your catalogs pretty care−free. But it will definitely pass you the ammo to shoot yourself in the foot, as will any templating system that has power. So you have to be careful, as you do in any scripting environment.

Most of the protections have to do with:

1. Tags like [cgi foo], [data ...] and such are not reparsed for tags.
2. The [value foo] tag never allows a left square bracket to be output.
3. Safe is used for Perl, which means that arbitrary perl code which reads/writes or uses IO is not possible.
4. The Interchange files don't need to be readable or writable by any other user ID, so CGI/PHP programs run by the web server can't get at them.
5. Dangerous operations are allowed only via global UserTag, and not by catalog UserTag.

But there are ways that user−entered data could end up getting parsed for tags. The most common breach is to take possibly tainted user data entered into a form and put it in a database without filtering it first. There are several ways to do that:

- For user−entered form fields like name, address, etc you should always use the [value ...] tag to display them.
- You can filter lists of variables automatically with

```
Filter   name      textarea_put
Filter   address   textarea_put
Filter   comments  textarea_put
Filter   email     textarea_put
```

  in catalog.cfg.
- You can prevent problems in some cases by using the data tag (i.e. [data table=foo col=bar key="[item−param something]"]) which is not reparsed for tags as [item−data ....] is. This is not foolproof unless you control the key in some fashion −− in most cases, you will use [data session username] or something you control.
- You should never put user−entered data directly in a Scratch variable, which can be called with mv_click and mv_check.
- Process all user−entered data before putting it in a database that could be displayed in a page later. Or process it before it is displayed.
- When putting data inside a container tag, the output of which is reparsed for more tags by default, consider setting reparse=0.

There are other ways to nail down your system and make it more difficult to have a security problem.

- Use the WRITE_CONTROL database settings.

```
Database products WRITE_CONTROL 1
```

> It is the default for MV DBM databases (i.e. you have to have a [flag type=write table=foo] to write) but not for SQL.
> The UI does the proper write flagging, so this should have no effect on it.
> AUTHORS NOTE: I should have made WRITE_CONTROL the default for SQL in MV4, my bad. As long as MV3 compatibility was as poor as it turned out to be, I should have gone for it.

- Set "NoAbsolute Yes" in interchange.cfg, it prevents [file /some/dir] or [include /some/rogue/file] from being used. NOTE: This is the default starting in IC 4.7.x.
- Split the admin server onto a different IC iteration from your production catalog, and disable the UI for the production catalog. That allows you to set the pages/ directory to be read−only for the production server, and to put WRITE_CONTROL in the production catalog_after.cfg or etc/<catalog id>.after file.

  You could also use a different username to access the SQL data, and make the production server username have read−only access.

  You can use a different UID for the admin server iteration, and make all directories except products/, upload/, tmp/, session/, and logs/ read−only for the production server UID.
- Make as much stuff read−only as you can. Develop a script which sets things read−write while you are admin−ing, and read−only otherwise.
- Always make ICDIR/*.cfg, ICDIR/bin/*, and ICDIR/lib/* read−only for the IC user.
- Don't let the interchange daemon user ID have read or write permission on things it doesn't need.
- Never use AllowGlobal in production if you have user−entered data going into a database. (AllowGlobal is not the default, so many people won't know what it is −− see Interchange Configuration.)

All in all, Interchange has been proven to be securable over time. But we all have to do our part and think about what we are doing with user−entered data.

13.2. Optimizing lists

Interchange has powerful search capabilities that allow you to produce lists of items for use in category lists, product lists, indexes, and other navigation tools.

These are a two−edged sword, though. Lists of hundreds or thousands of entries can be returned, and techniques that work well displaying only a few items may slow to a crawl when a large list is returned.

In general, when you are returning one item (i.e. a flypage) or a small list (i.e. a shopping cart) you can be pretty carefree in your use of [if ...] and [calc] and [perl] tags. When there are hundreds of items, though, you cannot; each complex test or embedded Perl snippet causes the Safe module to have to evaluate code, and each ITL tag requires parsing and argument building.

The Safe module is pretty fast considering what it does, but it can only generate a few thousand instances per second even on a fast system. And the ITL tag parser can likewise only parse thousands of tags per CPU second.

What to do? You want to provide complex conditional tests but you don't want your system to slow to a crawl. Luckily, there are techniques which can speed up complex lists by orders of magnitude.

13.2.1. Benchmarking

A non−precise benchmark of different iteration options can be done with the following global UserTag. Place this in a file in the usertag/ directory in the Interchange root:

```
UserTag benchmark Order start display
UserTag benchmark AddAttr
UserTag benchmark Routine <<EOR
my $bench_start;
my @bench_times;
sub {
    my ($start, $display, $opt) = @_;
    my @times = times();
    if($start or ! defined $bench_start) {
        $bench_start = 0;
        @bench_times = @times;
        for(@bench_times) {
            $bench_start += $_;
        }
    }
    my $current_total;
    if($display or ! $start) {
        for(@times) {
            $current_total += $_;
        }
        unless ($start) {
            $current_total = sprintf '%.3f', $current_total - $bench_start;
            for(my $i = 0; $i < 4; $i++) {
                $times[$i] = sprintf '%.3f', $times[$i] - $bench_times[$i];
            }
        }
        return $current_total if ! $opt->{verbose};
        return "total=$current_total user=$times[0] sys=$times[1] cuser=$times[2] csys=$times[3]"
    }
    return;
}
EOR
```

Then at the beginning of the code to check, call

```
[benchmark start=1]
```

to start the measurement. At the end

```
[benchmark]
```

will display the time used. Bear in mind that it is not precise, and that there may be variation due to system conditions. Also, the longer the times and the bigger the list, the better the comparison.

To see the system/user breakdown, do:

```
[benchmark verbose=1]
```

In general, "user" time measures Interchange processing time and and the rest are indicative of the database access overhead, which can vary widely from database to database.

13.2.2. Optimizations

- [PREFIX−tag] is faster than [parsed−tag]

```
[loop prefix=foo search="ra=yes"]

        [foo-data products image]
                is slightly faster than
        [foo-field image]
                which is MUCH faster than
        [data products image [foo-code]]
                which is faster than
        [data table=products column=image key="[foo-code]"]

[/loop]
```

The loop tags are interpreted by means of fast regular expression scans of the loop container text, and fetch an entire row of data in one query. The [data ...] ITL tag interpretation is delayed until after the loop is finished, whereby the ITL tag parser must find the tag, build a parameter list, then fetch the data with a separate query. If there are repeated references to the same field in the loop, the speedup can be 10x or more.

- Pre−fetch data with rf=field1,field2,field3 and access with [PREFIX−param field1].

mv_return_fields (otherwise known as "rf" in one−click terminology) sets the fields that are returned from a search. Once they are returned, they can be accessed with [PREFIX−param field]. They can also be referenced with [PREFIX−pos N], where N is a digit representing the ordinal position (i.e. starting with 0) in the list of fields.

The following are equivalent:

```
Benchmark loop-field list: [benchmark start=1]
<!-- [loop search="ra=yes/st=db"]
        [loop-code] price: [loop-field price] [/loop] -->
TIME: [benchmark]

Benchmark loop-param list: [benchmark start=1]
<!-- [loop search="ra=yes/st=db/rf=sku,price"]
        [loop-code] price: [loop-param price] [/loop] -->
TIME: [benchmark]
```

but the second is much, much faster.

- [PREFIX−alternate N] is available for row counting and display.

A common need when building tables is to conditionally close the table row or data containers. I see a lot of:

```
[loop search="ra=yes"]
[calc] return '<TR>' if [loop-increment] == 1; return[/calc]
[calc] return '' if [loop-increment] % 3; return '</TR>' [/calc]
[/loop]
```

Much faster, by a few orders of magnitude, is:

```
[loop search="ra=yes"]
[loop-change 1][condition]1[/condition]<TR>[/loop-change 1]
```

```
[loop-alternate 3]</TR>[/loop-alternate]
[/loop]

If you think you need to close the final row by checking the
final count, look at this:

[loop search="ra=yes"]
[on-match]
        <TABLE>
        <TR>
[/on-match]

[list]
                <TD>[loop-code]</TD>
        [loop-alternate 3]</TR><TR>[/loop-alternate]
[/list]

[on-match]
        </TR>
        </TABLE>
[/on-match]

[no-match]
        No match, sorry.
[/no-match]
```

```
[/loop]
```

This is a hundred times faster than anything you can build with multiple [calc] tags.

- Use simple go/nogo comparisons in [if ...]

Consider these two snippets:

```
[if scratch|value|cgi key] THEN [/if]
```

and:

```
[if scratch|value|cgi key == '1'] THEN [/if]
```

The first one doesn't require Perl evaluation. It simply checks to see if the value is blank or 0, and returns true
if it is anything but. Of course this requires setting your test values to blank or 0 instead of "No" or " " or
somesuch, but it is anywhere from 20–35% faster.

Try it on the foundation demo:

```
---- begin test ---

Overhead:
[benchmark start=1]
        <!-- [loop search="ra=yes"][set cert][loop-field gift_cert][/set][/loop] -->
[benchmark]
<P>


if scratch compare:
[benchmark start=1]
        <!--
```

```
            [loop search="ra=yes"]
            [set cert][loop-field gift_cert][/set]
            [loop-code] [if scratch cert] YES [else] NO [/else][/if]
            [loop-code] [if scratch cert] YES [else] NO [/else][/if]
            [loop-code] [if scratch cert] YES [else] NO [/else][/if]
            [loop-code] [if scratch cert] YES [else] NO [/else][/if]
            [loop-code] [if scratch cert] YES [else] NO [/else][/if]
            [/loop]
            -->

    [benchmark]
    <P>

    if scratch compare eq 1:
    [benchmark start=1]
            <!--
            [loop search="ra=yes"]
            [set cert][loop-field gift_cert][/set]
            [loop-code] [if scratch cert == 1] YES [else] NO [/else][/if]
            [loop-code] [if scratch cert == 1] YES [else] NO [/else][/if]
            [loop-code] [if scratch cert == 1] YES [else] NO [/else][/if]
            [loop-code] [if scratch cert == 1] YES [else] NO [/else][/if]
            [loop-code] [if scratch cert == 1] YES [else] NO [/else][/if]
            [/loop]
            -->
    [benchmark]
    <P>

    [page @@MV_PAGE@@]Again[/page]

    ---- end test ---
```

- Use [PREFIX−calc] instead of [calc] or [perl]

You can execute the same code as [calc] with [PREFIX−calc], which has two benefits:

1. It doesn't require ITL parsing.
2. It is executed during the loop instead of after it.

The [PREFIX−calc] object has complete access to all normal embedded Perl objects like $Values, $Carts, $Tag, and such. If you want to make a data table (i.e. "products" or "pricing") available for access inside of it, just do:

```
    [perl tables="products pricing"] [/perl]
```

prior to list start. Now you can do something like:

```
[loop search="ra=yes"]
    [loop-calc]
        $desc = $Tag->data('products', 'description', '[loop-code]');
        $link = $Tag->page('[loop-code]');
        return "$link $desc </A>";
    [/loop-calc] <BR>
[/loop]
```

- ADVANCED: Precompile and execute with [PREFIX−sub] and [PREFIX−exec]

For repetitive routines, you can achieve a considerable savings in CPU by pre−compiling your embedded Perl code.

In the "Construct Something" demo, the bar_link() routine in catalog_before.cfg is an example of compiling the subroutine once at catalog configuration time.

You can also compile routines at the time of the list execution with [item−sub routine] CODE [/item−sub]. This means only one Safe evaluation is done −− every time the [loop−exec routine] is called, it is done fast as a call to the routine. This can be 10 times or more faster than separate [calc] calls, or 5 times faster than separate [PREFIX−calc] calls.

Example:

```
[benchmark start=1]
loop-calc:
  <!--
        [loop search="st=db/fi=country/ra=yes/ml=1000"]
        [loop-calc]
                my $code = q{[loop-code]};
                return "code '$code' reversed is " . reverse($code);
        [/loop-calc]
        [/loop]
  -->

[benchmark]

<P>

[benchmark start=1]
loop-sub and loop-exec:
  <!--
        [loop search="st=db/fi=country/ra=yes/ml=1000"]
        [loop-sub country_compare]
                my $code = shift;
                return "code '$code' reversed is " . reverse($code);
        [/loop-sub]
        [loop-exec country_compare][loop-code][/loop-exec]
        [/loop]
  -->

[benchmark]
```

- ADVANCED: Execute and save with [query ...], then use an embedded Perl routine.

You can run [query arrayref=myref sql="query"], which saves the results of the search/query in a Perl reference. It is then available in $Tmp−>{myref}. (Of course, "myref" can be any arbitrary name.)

This is the fastest possible method to display a list.

Observe:

```
--- begin test code ---
[set waiting_for]os28004[/set]

[benchmark start=1] Embedded Perl
<!--
[query arrayref=myref sql="select sku, price, description from products"]
```

```
            <!-- make query, this container text is not used. -->
    [/query]

    [perl]
            # Get the query results, has multiple fields
            my $ary = $Tmp->{myref};
            my $out = '';
            foreach $line (@$ary) {
                    my ($sku, $price, $desc) = @$line;
                    if($sku eq $Scratch->{waiting_for}) {
                            $out .= "We were waiting for this one!!!!\n";
                    }
                    $out .= "sku: $sku price: $price description: $desc\n";
            }
            return $out;
    [/perl]
    -->
    TIME: [benchmark]

    [benchmark start=1] All loop
    <!--
    [query list=1 sql="select sku, price, description from products"]
            [if scratch waiting_for eq '[sql-code]']We were waiting for this one!!!!
            [/if] sku: [sql-code]price: [sql-param price] desc: [sql-param description]
    [/query]
    -->

    TIME: [benchmark]

    --- end test code ---
```

• Other things that help:

    ♦ Avoid interpolate=1 when possible. A separate tag parser must be spawned every time you do this. Many times people use this without needing it.

    ♦ Avoid saving large values to Scratch, as these have to be written to the users session. If you need them only for the current page, clear at the end by using [tmp scratch_var] contents [/tmp], which is the same as [seti scratch_var] contents [/seti] except clears the value before the session is written. You can also use [scratchd scratch_var] to return the contents and delete them from the session at the same time.

    ♦ Use the [more−list] facility to break up your large searches. You can use them in [query ....] and [loop ...] searches as well −− see the docs.

14. Using Interchange with Oracle

Question: should we be using the DBI ChopBlanks setting for Oracle or is Interchange trimming trailing space from CHAR fields itself?

IC daemon user should have environment variables ORACLE_HOME and possibly NLS_LANG set.

Mark Johnson (Red Hat Professional Services) wrote this trigger on TABLE_NAME to update the MOD_TIME column on insert or update. The user must have been granted the RESOURCE role to create triggers. Here it is:

CREATE TRIGGER tr_modtime_for_TABLE_NAME BEFORE INSERT OR UPDATE ON TABLE_NAME FOR EACH ROW BEGIN

new.MOD_TIME := SYSDATE; END; /

14. Using Interchange with Oracle

15. Using Interchange with PostgreSQL

Make sure you have DBD::Pg installed and tested. Make sure POSTGRES_INCLUDE and POSTGRES_LIB environment variables are set.

16. Using Interchange with MySQL

Permissions. test_ databases usually special.

17. Using Interchange with Apache

Slightly modified article posted to the old minivend–users mail list. Minivend–users is now interchange–users.

```
Date: Thu, 7 Sep 2000 12:08:37 -0700
From: Bill Randle <billr@exgate.tek.com>
To: minivend-users@minivend.com
Subject: Re: [mv] no /cgi-bin/storename/

On Sep 6,  5:13am, Victor Nolton wrote:
} Subject: [mv] no /cgi-bin/storename/
} ******    message to minivend-users from Victor Nolton <ven@pragakhan.com> ******
}
} I've noticed some of the catalogs I've done are not indexed well with
} the search engine, though most pages have meta tags, there is a
} robot.txt file and so on and so forth.I assume it's due to the
} cgi-bin in the url (not sure).
}
} I'd like to start having stores be like
}
} http://www.yourdomain.com/index.html
} http://www.yourdomain.com/ord/basket.html
} instead of
} http://www.yourdomain.com/cgi-bin/yourstore/index.html
} http://www.yourdomain.com/cgi-bin/yourstore/ord/basket.html
}
} how do you accomplish this? I assume it can be done somehow.

In addition to using mod_minivend, previosuly suggested, you can do this
with Apache rewrite rules in the VirtualHost directive for yourdomain.com:

<VirtualHost a.b.c.d>
    ServerAdmin support@mainhost.com
    DocumentRoot /home/httpd/html/yourstore
    ServerName www.yourdomain.com
    ErrorLog logs/yourdomain-error_log
    CustomLog logs/yourdomain-access_log common
    ScriptAlias /cgi-bin/ "/home/httpd/cgi-bin/"
    RewriteEngine On
    RewriteRule ^$  /cgi-bin/yourstore/index.html         [PT,L]
    RewriteRule ^/$ /cgi-bin/yourstore/index.html         [PT,L]
    RewriteRule ^/index\.html$ /cgi-bin/yourstore/index.html [PT,L]
    RewriteRule ^/cgi-bin/yourstore/.* -                  [PT,L]
    RewriteRule ^/.*images/.* -                           [PT,L]
    RewriteRule ^/(.*) /cgi-bin/yourstore/$1              [PT,L]
</VirtualHost>

I just did this for a client and it works quite well (as long as you're
using a fairly recent version of Apache as your webserver).

        -Bill
```

18. Perl/Interchange FAQ

18.1. Cameron Prince's local Perl installation how−to

1. Login as user. In this example, we'll call the user bob. Bob's home directory is /home/bob.
2. Get the perl tarball and extract it in /home/bob. (tar −xzvf perl−5.6.0.tar.gz)
3. Create a directory for the local perl. (mkdir /home/bob/local−perl)
4. Compile perl.
    1. cd perl−5.6.0
    2. sh Configure
    3. Choose all the defaults until you get to: "Directories to use for library searches?" Here you want to enter the new local perl path, as well as the defaults. So you should enter something like: /home/bob/local−perl/lib /usr/local/lib /lib /usr/lib
    4. Continue choosing defaults till you get to: "Any additional ld flags (NOT including libraries)?" This should be: −L/home/bob/local−perl/lib
    5. Continue choosing defaults till you get to: "Installation prefix to use? (~name ok)" This should be: /home/bob/local−perl
    6. Choose all defaults till you get to: "Directory /home/bob/local−perl/bin doesn't exist. Use that name anyway?" Enter y.
    7. Continue choosing defaults till you get to: "Do you want to install perl as /usr/bin/perl?" Enter n.
    8. Continue choosing defaults till you get to: "Directory /home/bob/local−perl/bin doesn't exist. Use that name anyway?" Enter y.
    9. Directory /home/bob/local−perl/bin doesn't exist. Use that name anyway? Enter y.
    10. Continue taking defaults till you return to a prompt.
    11. make
    12. make test
    13. make install
5. /home/bob/local−perl/bin/perl −v
   You should see: This is perl, v5.6.0
6. edit /home/bob/.bash_rc
   Change: PATH=$PATH:$HOME/bin
   To: PATH=/home/bob/local−perl/bin:$PATH:$HOME/bin
7. Logout and log back in.
8. which perl
   You should see: ~/local−perl/bin/perl or /home/bob/local−perl/bin/perl
9. perl −MCPAN −e 'install Bundle::Interchange'
   Keep running this until you see:

```
MD5 is up to date.
MIME::Base64 is up to date.
URI is up to date.
Net::FTP is up to date.
MIME::Base64 is up to date.
Digest::MD5 is up to date.
HTML::Tagset is up to date.
HTML::Parser is up to date.
HTML::HeadParser is up to date.
LWP is up to date.
Term::ReadKey is up to date.
Term::ReadLine::Perl is up to date.
Business::UPS is up to date.
```

```
SQL::Statement is up to date.
Storable is up to date.
DBI is up to date.
Safe::Hole is up to date.
```

You may need to get the modules via ftp and install them by hand. For instance, during the test used to create this document, I had to get URI and LWP and install by hand before everything reported that it was up to date. To do this, follow these steps:

1. ftp ftp.cpan.org
2. cd /CPAN/modules/by−module/URI
3. bin
4. get URI−1.10.tar.gz
5. quit
6. tar −xzvf URI−1.10.tar.gz
7. cd URI−1.10
8. perl Makefile.pl
9. make
10. make test
11. make install

Use the same basic steps for any module not properly installed by using perl −MCPAN −e 'install Bundle::Interchange'

Now, install Interchange as normal.