Configuration Reference

Table of Contents

Table of Contents

Table of Contents

Table of Contents

1. Interchange Configuration Files

The Red Hat Interchange 4.8 Configuration Reference is an alphabetical reference to the configuration directives used in Interchange global and catalog configuration files.

Interchange has multiple catalog capability, and therefore splits its configuration into two pieces. One is global, `interchange.cfg`, and affects every catalog running under it. The other, `catalog.cfg` is specific to an individual catalog, and has no effect on other catalogs.

1.1. Directive syntax

Configuration directives are normally specified with the directive as the first word on the line, with its value or values following. Capitalization of the directive name is not signifigant. Leading and trailing whitespace is stripped from the line.

Including files in directives

Additional files may be called with an include file notation like this:

```
DirectiveName <includefile
```

Files included from interchange.cfg are relative to the Interchange software directory. Files included from catalog.cfg are relative to the catalog directory.

Here documents

A "here document" can be used to spread directive values over several lines, with the usual Perl <<MARKER syntax. No semicolon is used to terminate the marker. The closing marker must be the only thing on the line. No leading or trailing characters are allowed, not even whitespace. Here is a hypothetical directive using a here document:

```
DirectiveName <<EOD
    setting1
    setting2
    setting3
EOD
```

That is equivalent to:

```
DirectiveName setting1 setting2 setting3
```

Include single setting from file

Value can be pulled from a file with <file:

```
Variable MYSTUFF <file
```

This works well for includes that must be of the highest possible performance. They can be simply placed in a page with __VARIABLE__.

include

Other configuration files can be included in the current one. For example, common settings can be set in one file:

```
include common.cfg
```

Or all files in one directory:

```
include usertag/*
```

ifdef and ifndef

ifdef/endif and ifndef/endif pairs can be used:

```
Variable ORDERS_TO email_address

ifdef ORDERS_TO
ParseVariables Yes
MailOrderTo __ORDERS_TO__
ParseVariables No
endif

ifdef ORDERS_TO =~ /foo.com/
# Send all orders at foo.com to one place now
# Set ORDERS_TO to stop default setting
Variable  ORDERS_TO  1
MailOrderTo   orders@foo.com
endif

ifdef ORDERS_TO eq 'nobody@nowhere.com'
# Better change to something else, set ORDERS_TO to stop default
Variable  ORDERS_TO  1
MailOrderTo   someone@somewhere.com
endif

ifndef ORDERS_TO
#Needs to go somewhere....
MailOrderTo  webmaster@localhost
endif
```

2. interchange.cfg

The VendRoot directory, specified in the main program `interchange`, is the default location of all of the Interchange program, configuration, special, and library files. Unless changed in the call to `interchange`, the main Interchange server configuration file will be `interchange.cfg` in the VendRoot directory.

The directives defined in `interchange.cfg` affect the entire Interchange server and all catalogs running under it. Multiple Interchange servers may be run on the same machine with totally independent operation.

Following is an alphabetical listing of all global configuration directives.

2.1. ActionMap *global*

Allows setting of Interchange form actions, usually with a Perl subroutine. Actions are page names like:

```
process  Perform a processing function
order    Order items
scan     Search based on path info
search   Search based on submitted form variables
```

The global version of ActionMap applies to all catalogs. If the same action is specified in `catalog.cfg`, it will pertain. See `ActionMap` in that section.

2.2. AddDirective *global*

Adds a configuration directive that will be parsed for every `catalog.cfg` file. Accepts three parameters: the name of the directive, the name of the parser (if any), and the default value (if any). The following definition would add a directive "Foo," with parser "parse_bar," and a default value of "Hello, world!":

```
AddDirective  Foo  bar "Hello, world!"
```

If the parser is not defined, the directive value will be scalar and the same as what the user passes in the config file. If defined, the parser must be extant before it can be referenced, is always resident in Vend::Config, and begins with the string `parse_`. Examples can be found in the files in the distribution software directory `compat/`.

2.3. AdminSub *global*

Marks a global subroutine for use only by catalogs that are set to `AllowGlobal` (see below). Normally global subroutines can be referenced (in embedded Perl) by any catalog.

```
AdminSub  dangerous
```

2.4. AllowGlobal *global*

Specifies catalog identifiers that may define subroutines and UserTag entries that can operate with the full permissions of the server. **Don't use this unless the catalog user is trusted implicitly.** Default is blank.

```
AllowGlobal  simple
```

Using AllowGlobal is never necessary, and is always dangerous in a multi−user environment. Its use is not recommended.

2.5. AutoVariable *global*

Specifies directives which should be translated to Variable settings. For scalars, the directive name becomes the Variable name and yields its value, i.e. `ErrorFile` becomes `__ErrorFile__`, which would by default be `error.log`. Array variables have a `_N` added, where `_N` is the ordinal index, i.e. `SafeUntrap` becomes `__SafeUntrap_0__`, `__SafeUntrap_1__`, etc. Hash variables have a _KEY added, i.e. `SysLog` becomes `__SysLog_command__`, `__SysLog_facility__`, etc. Doesn't handle hash keys that have non−word characters or whitespace. Only single−level arrays and hashes are translated properly.

See `AutoVariable` in `catalog.cfg`.

2.6. Catalog *global*

Specifies a catalog that can run using this Interchange server. This directive is usually inserted into `interchange.cfg` by the makecat program when you build a new catalog.

There are three required parameters, as shown in this example:

```
Catalog  simple /home/interchange/simple /cgi−bin/simple
```

The first is the name of the catalog. It will be referred to by that name in error, warning, and informational messages. It must contain only alphanumeric characters, hyphens, and underscores. It is highly recommended that it be all lower case.

The second is the base directory of the catalog. If the directory does not contain a `catalog.cfg` file, the server will report an error and refuse to start.

The third is the `SCRIPT_NAME` of the link program that runs the catalog. This is how the catalog is selected for operation. Any number of alias script names may be specified as additional parameters. This allows the calling path to be different while still calling the same catalog:

```
Catalog  simple /home/interchange/simple /cgi−bin/simple /simple
```

This is useful when calling an SSL server or a members−only alias that requires a username/password via HTTP Basic authorization. All branched links will be called using the aliased URL.

The script names must be unique among CGI program paths that run on this server; the same name cannot be used for more than one catalog unless the `FullURL` directive is specified. In this case, the parameter may be specified as:

```
www.yourcompany.com/cgi−bin/simple
www.theirs.com/cgi−bin/simple
```

Each of those 'simple' catalogs would then call a different catalog.

Optionally, individual `Catalog` directives that specify each of the different parameters may be used. The equivalent of our original example directive above is:

```
Catalog simple directory /home/interchange/simple
Catalog simple script    /cgi-bin/simple
Catalog simple alias     /simple
```

Global directives may be specified that will change for that catalog only. This is mostly useful for `ErrorFile` and `DisplayErrors`:

```
Catalog simple directive ErrorFile /var/log/interchange/simple_error.log
```

2.7. CheckHTML *global*

Set to the name of an external program that will check the users HTML when they set `[flag checkhtml]` or `[tag flag checkhtml][/tag]` in their page.

```
CheckHTML  /usr/local/bin/weblint
```

2.8. ConfigAllAfter *global*

The name of a file (or files) which should be read as a part of every catalog's configuration, after any other configuration files are read. Default is `catalog_after.cfg`.

```
ConfigAllAfter   check_actions.cfg check_variables.cfg
```

2.9. ConfigAllBefore *global*

The name of a file (or files) which should be read as a part of every catalog's configuration, before any other configuration files are read. Default is `catalog_before.cfg`.

```
ConfigAllBefore   set_actions.cfg set_variables.cfg
```

2.10. ConfigParseComments *global*

Set to No if you want old−style '#include', '#ifdef', or '#ifndef' to be treated as the comments they appear to be. The default is Yes, which means both '#include' and 'include' do the same thing. (Use a space after the '#' if you really want to comment out the command.)

Interchange prior to version 4.7 used a different syntax for meta−directives 'include', 'ifdef', and 'ifndef' in configuration files. The commands were borrowed from the C preprocessor, and true to their C heritage, they started with '#': '#include', '#ifdef', '#ifndef'. Interchange configuration files, unlike C, uses '#' to begin one−line comments, which meant that a newcomer at first glance might assume that:

```
#Variable DEBUG 1
#include more.cfg
```

were both comments, when in fact the second was a live #include command.

To begin to make things more consistent, Interchange 4.7 and up now recognize those meta−directives without the leading '#', and the included demo catalog sets this directive to No so that lines beginning with '#' really are skipped as comments, regardless of what comes after.

2.11. Database *global*

Defines a database which is global and available to all catalogs. Writing can be controlled by catalog. See `Database`.

2.12. DataTrace *global*

Set DBI to trace at the level specified. Valid values are:

0 − Trace disabled.

1 − Trace DBI method calls returning with results or errors.

2 − Trace method entry with parameters and returning with results.

3 − As above, adding some high−level information from the driver and some internal information from the DBI.

4 − As above, adding more detailed information from the driver. Also includes DBI mutex information when using threaded Perl.

5 and above − As above but with more and more obscure information.

Trace level 1 is best for most Interchange debug situations. Trace will only be enabled when `DebugFile` is specified, as that file is the target for the trace. Example:

```
    DataTrace   1
```

Default is 0. Directive added in 4.7.0.

2.13. DebugFile *global*

Names a file, relative to the Interchange root directory, which should store the output of `logDebug` statements, and warnings if warnings are enabled.

```
        DebugFile  /tmp/icdebug
```

2.14. DisplayErrors *global*

While all errors are reported in the error log file, errors can also be displayed by the browser. This is convenient while testing a configuration. Unless this is set, the `DisplayErrors` setting in the user catalogs will have no effect. Default is No.

```
   DisplayErrors       Yes
```

Note: This changes the value of $SIG{__DIE__} and may have other effects on program operation. This should NEVER be used for normal operation.

2.15. DomainTail *global*

Implements the domain/IP session qualifiers so that only the major domain is used to qualify the session ID. This is a compromise on security, but it allows non−cookie−accepting browsers to use multiple proxy servers in the same domain. Default is Yes.

```
DomainTail No
```

If encrypting credit cards with PGP or GPG, or are using a payment service like CyberCash, look at the `WideOpen` directive, which enables more browser compatibility at the cost of some security.

2.16. DumpStructure *global*

Tells Interchange to dump the structure of catalogs and the Interchange server to a file with the catalog name and the extension `.structure`. Use this to see how directives have been set.

2.17. EncryptProgram *global*

Specifies the default encryption program that should be used to encrypt credit card numbers and other sensitive information. Default is `gpg` if found on the system; then `pgpe`, if found; then `pgp`, and finally `none`, disabling encryption.

This is used to set the default in `catalog.cfg`, which has its own independent setting of `EncryptProgram`.

2.18. Environment *global*

Environment variables to inherit from the calling CGI link program. An example might be PGPPATH, used to set the directory which PGP will use to find its key ring.

```
Environment  MOD_PERL REMOTE_USER PGPPATH
```

2.19. ErrorFile *global*

Sets the name of the global error log. The default is `error.log` in the Interchange software directory.

```
ErrorFile  /var/log/interchange/log
```

Of course, the user ID running the Interchange server must have permission to write that file.

Optionally, syslog error logging can be set up as well. See `SysLog`.

2.20. FormAction *global*

Allows a form action (like the standard ones `return, submit, refresh,` etc.) to be set up. It requires a Perl subroutine as a target:

```
FormAction foo <<EOR
```

```
sub {
    $CGI->{mv_nextpage} = 'bar';
}
EOR
```

If it returns a true (non−zero, non−empty) value, Interchange will display the page defined in $CGI−>{mv_nextpage}. Otherwise, Interchange will not display any page. The default Interchange actions can be overridden, if desired. There is also a catalog−specific version of this directive, which overrides any action of the same name.

The global version affects all catalogs −− there is also a catalog−specific version of FormAction which is protected by Safe.

2.21. FullUrl *global*

Normally Interchange determines which catalog to call by determining the SCRIPT_NAME from the CGI call. This means that different (and maybe virtual) hosts cannot use the same SCRIPT_NAME to call different catalogs. Set `FullUrl` to Yes to differentiate based on the calling host. Then, set the server name in the Catalog directive accordingly, such as `yourdomain.com/cgi−bin/simple`. A yes/no directive, the default is No.

```
FullUrl  Yes
```

If it is set in this fashion, `all` catalogs must be defined in this fashion. NOTE: The individual catalog setting will not work, as this is used before the catalog name is known.

2.22. GlobalSub *global*

Defines a `global` subroutine for use by the `[perl sub] subname arg /perl]` construct. Use the "here document" capability of Interchange configuration files to make it easy to define:

```
GlobalSub <<EOF

sub count_orders {
    my $counter = new File::CounterFile "/tmp/count_orders", '1';
    my $number = $counter->inc();
    return "There have been $number orders placed.\n";
}
EOF
```

As with Perl "here documents," the EOF (or other end marker) must be the ONLY thing on the line, with no leading or trailing white space. Do not append a semicolon to the marker. (The above marker appears indented. It should not be that way in the file!)

IMPORTANT NOTE: These global subroutines are not subject to security checks. They can do most anything! For most purposes, scratch subroutines or catalog subroutines (also Sub) are better.

`GlobalSub` routines are subject to full Perl use strict checking, so errors are possible if lexical variables or complete package qualifications are not used for the variables.

2.23. HammerLock *global*

The number of seconds after which a locked session could be considered to be lost due to malfunction. This will kill the lock on the session. Only here for monitoring of session hand−off. If this error shows up in the error log, the system setup should be examined. Default is 30.

```
HammerLock          60
```

This mostly doesn't apply to Interchange when using the default file−based sessions.

2.24. HitCount *global*

Increments a counter in `ConfDir` for every access to the catalog. The file is named `hits.catalogname`, where `catalogname` is the short catalog identifier. A Yes/No directive, default is `No`.

```
HitCount  Yes
```

2.25. HouseKeeping *global*

How often, in seconds, the Interchange server will "wake up" and look for user reconfiguration requests and hung search processes. On some systems, this wakeup is the only time the server will terminate in response to a stop command. Default is 60.

```
HouseKeeping    5
```

2.26. Inet_Mode *global*

Determines whether INET−domain sockets will be monitored on startup. Overridden by the command−line parameter `−i`. Default is `Yes`.

2.27. IpHead *global*

Implements the domain/IP session qualifiers so that only the first `IpQuad` dot−quads of the IP address are used to qualify the session ID. The default is 1. This is a slight compromise on security, but it allows non−cookie−accepting browsers, like AOL's V2.0, to use multiple proxy servers.

`DomainTail` is preferable unless one of your HTTP servers does not do host name lookups. Default is `No`, and DomainTail must be set to `No` for it to operate.

```
IpHead Yes
```

2.28. IpQuad *global*

The number of dot−quads that IpHead will look at. Default is 1.

```
IpQuad  2
```

2.29. Locale *global*

Sets the global `Locale` for use in error messages. Normally set from a file's contents, as in the example before:

```
Locale <locale.error
```

2.30. LockoutCommand *global*

The name of a command (as it would be entered from the shell) that will lock out the host IP of an offending system. The IP address will be substituted for the first occurrence of the string %s. This will be executed with the user ID that Interchange runs under, so any commands that require root access will have to be wrapped with an SUID program.

On Linux, a host may be locked out with:

```
ipfwadm −I −i deny −S %s
```

This would require root permissions, however, under normal circumstances. Use `sudo` or another method to wrap and allow the command.

A script can be written which modifies an appropriate access control file, such as .htaccess for your CGI directory, to do another level of lockout. A simple command line containing `perl −0777 −npi −e 's/deny/deny from %s\ndeny/' /home/me/cgi−bin/.htaccess` would work as well (remember, the %s will become the IP address of the offending user).

```
LockoutCommand   lockout %s
```

2.31. LockType *global*

Allows selection of file locking method used throughout Interchange. Options are 'flock', 'fcntl', and 'none'. Added in 4.7.0.

Default is flock. See the flock(2) manpage for details.

The fcntl setting is needed for NFS filesystems; for NFS−based locking to work, the NFS lock daemon (lockd) must be enabled and running on both the NFS client and server. Locking with fcntl works on Linux and should work on Solaris, but is not guaranteed to work on all OSes.

The none setting turns off file locking entirely, but that is never recommended. It might be useful to check if locking is causing hangs on the system.

If you are only accessing sessions on an NFS−mounted directory but the rest of Interchange is on the local filesystem, you can instead set the SessionType catalog directive to 'NFS', which enables fcntl locking for sessions only on a per−catalog basis.

2.32. Mall *global*

Set to `Yes` to issue cookies only for the current catalog's script. By default, when Interchange issues a cookie it does so for the base domain. This will allow multiple catalogs to operate on the same domain without interfering with each others session ID.

A yes/no directive.

```
Mall   Yes
```

2.33. MaxServers *global*

The maximum number of servers that will be spawned to handle page requests. If more than MaxServers requests are pending, they will be queued (within the defined capability of the operating system, usually five pending requests) until the number of active servers goes below that value.

```
MaxServers     4
```

Default is 10.

2.34. NoAbsolute *global*

Whether Interchange `[file ...]` and other tags can read any file on the system (that is readable by the user id running the Interchange daemon). The default is No, which allows any file to be read. This should be changed in a multi−user environment to minimize security problems.

```
NoAbsolute     Yes
```

2.35. PIDcheck *global*

If non−zero, enables a check of running Interchange processes during the housekeeping routine. If a process has been running (or is hung) for longer than PIDcheck seconds then a kill −9 will be issued and the server count decremented. During the housekeeping routine, the number of servers checked by `MaxServers` will be recounted based on PID files.

Default is 0, disabling the check.

```
PIDcheck   300
```

If have long−running database builds, this needs to be disabled. Set it to a high value (perhaps 600, for 10 minutes), or use the offline script.

2.36. PIDfile *global*

The file which will contain the Interchange server process ID so that it can be read to determine which proces should be sent a signal for stopping or reconfiguring the server.

```
PIDfile  /var/run/interchange/interchange.pid
```

This file must be writable by the Interchange server user ID.

2.37. Profiles *global*

Names a file (or files) which contain `OrderProfile` and `SearchProfile` settings that will apply for all catalogs.

```
Profiles     etc/profiles.common
```

2.38. SafeUntrap *global*

Sets the codes that will be untrapped in the `Safe.pm` module and used for embedded Perl and conditional operations. View the Safe.pm documentation by typing `perldoc Safe` at the command prompt. The default is `ftfile sort`, which untraps the file existence test operator and the sort operator. Define it as blank to prevent any operators but the default restrictive ones.

```
SafeUntrap     ftfile sort ftewrite rand
```

2.39. SendMailProgram *global*

Specifies the program used to send email. Defaults to '/usr/lib/sendmail'. If it is not found at startup, Interchange will return an error message and refuse to start.

```
SendMailProgram     /bin/mailer
```

A value of 'none' will disable the sending of emailed orders. Orders must be read from a tracking file, log, or by other means.

2.40. SOAP *global*

If set to Yes, allows handling of SOAP rpc requests.

2.41. SOAP_Host

The list of hosts that are allowed to connect to for SOAP rpc requests. Default is `localhost 127.0.0.1`.

2.42. SOAP_MaxRequests

The maximum number of requests a SOAP rpc server will handle before it commits suicide and asks for a replacement server. This prevents runaway memory leaks.

2.43. SOAP_Perms

The permissions that should be set on a SOAP UNIX−domain socket. Default is `0660`, which allows only programs running as the same UID as Interchange to access the socket.

2.44. SOAP_Socket

A list of sockets which should be monitored for SOAP requests. If they fit the form NNN.NNN.NNN.NNN:PPPP, they are IP addresses and ports for monitoring INET−domain sockets, any other pattern is assumed to be a file name for monitoring in the UNIX domain.

```
SOAP_Socket 12.23.13.31:7770 1.2.3.4:7770 /var/run/interchange/soap
```

2.45. SOAP_StartServers

The number of SOAP servers which should be started to handle SOAP requests. Default is 1.

```
SOAP_StartServers   10
```

2.46. SocketFile *global*

The name of the file which is used for UNIX−domain socket communications. Must be in a directory where the Interchange user has write permission.

```
SocketFile  /var/run/interchange/interchange.socket
```

Default is `etc/socket` or the value of the environment variable MINIVEND_SOCKET. If set, it will override the environment. It can be set on the command line as well:

```
bin/interchange −r SocketFile=/tmp/interchange.socket
```

2.47. SocketPerms *global*

The permissions (prepend a 0 to use octal notation) that should be used for the UNIX−domain socket. Temporarily set this to 666 on the command line to debug a permission problem on `vlink`.

```
bin/interchange −r SocketPerms=0666
```

2.48. StartServers

The number of Interchange page servers which should be started to handle page requests when in `PreFork` mode. Default is 1.

```
SOAP_StartServers   10
```

2.49. SubCatalog *global*

Allows definition of a catalog which shares most of the characteristics of another catalog. Only the directives that are changed from the base catalog are added. The parameters are: 1) the catalog ID, 2) the base catalog ID, 3) the directory to use (typically the same as the base catalog), and 4) the SCRIPT_NAME that will trigger the catalog. Any additional parameters are aliases for the SCRIPT_NAME.

The main reason that this would be used would be to conserve memory in a series of stores that share most of the same pages or databases.

```
SubCatalog   sample2 sample /usr/catalogs/sample /cgi-bin/sample2
```

2.50. SysLog *global*

Set up syslog(8) error logging for Interchange.

```
SysLog  command  /usr/bin/logger
SysLog  tag      int1
SysLog  alert    local3.warn
SysLog  warn     local3.info
SysLog  info     local3.info
SysLog  debug    local3.debug
```

This would cause global errors to be logged with the command:

```
/usr/bin/logger -t int1 -p local3.alert
```

and cause system log entries something like:

```
Oct 26 17:30:11 bill int1: Config 'co' at server startup
Oct 26 17:30:11 bill int1: Config 'homefn' at server startup
Oct 26 17:30:11 bill int1: Config 'simple' at server startup
Oct 26 17:30:11 bill int1: Config 'test' at server startup
Oct 26 17:30:13 bill int1: START server (2345) (INET and UNIX)
```

This would work in conjunction with a UNIX syslogd.conf entry of:

```
    # Log local3 stuff to Interchange log
    local3.*               /var/log/interchange.log
```

A custom wrapper can be created around it to get it to behave as desired. For instance, if you didn't want to use syslog but instead wanted to log to a database (via DBI), you could create a Perl script named "logdatabase" to log things:

```
#!/usr/bin/perl

my $script_name = "logdatabase";
use DBI;
use Getopt::Std;

getopts('d:p:T:k:')
    or die "$script_name options: $@\n";

use vars qw/$opt_d $opt_p $opt_T $opt_k/;

my $dsn   = $opt_d || $ENV{DBI_DSN};
my $template = $opt_T
    || "insert into log values ('~~KEY~~', '~~LEVEL~~', '~~MSG~~')";

my $dbh = DBI->connect($dsn)
    or die "$script_name cannot connect to DBI: $DBI::errstr\n";

my %data;

$data{KEY} = $opt_k || '';

local ($/);
```

```
$data{MSG} = <>;

$data{LEVEL} = $opt_p || 'interchange.info';

$template =~ s/\~\~(\w+)\~\~/$dbh->quote($data{$1})/;

my $sth = $dbh->prepare($template)
    or die "$script_name error executing query: $template\n";

$sth->execute()
    or die "$script_name error executing query: $template\n";

exit;
```

2.51. TcpHost *global*

When running in INET mode, using `tlink`, specifies the hosts that are allowed to send/receive transactions from any catalog on this Interchange server. Can be either an name or IP number, and multiple hosts can be specified in a space−separated list. Default is localhost.

```
TcpHost         localhost secure.domain.com
```

2.52. TcpMap *global*

When running in INET mode, using `tlink` or the internal HTTP server, specifies the port(s) which will be monitored by the Interchange server. Default is 7786.

To use the internal HTTP server (perhaps only for password−protected queries), a catalog may be mapped to a port. If three catalogs were running on the server `www.akopia.com`, named `simple`, `sample`, and `search`, the directive might look like this:

```
TcpMap    7786 −  7787 simple 7788 sample 7789 search
```

Note: To map large numbers of ports, use the <<MARKER here document notation in interchange.cfg. With this in effect, the internal HTTP server would map the following addresses:

```
*:7786   mv_admin
*:7787   simple
*:7788   sample
*:7789   search
```

Note: This does not pertain to the use of `tlink`, which still relies on the CGI SCRIPT_PATH. To enable this, the SCRIPT_PATH aliases /simple, /sample, etc. must be set in the `Catalog` directive. This would look like:

```
Catalog  simple  /home/interchange/catalogs/simple /cgi−bin/simple /simple
```

To bind to specific IP addresses, add them in the same fashion that they would as an Apache Listen directive:

```
TcpMap <<EOF
    127.0.0.1:7786       −
    www.akopia.com:7787  −
EOF
```

Note: As usual, the EOF should be at the beginning of a line with no leading or trailing whitespace.

2.53. TemplateDir *global*

Sets a directory which will be searched for pages if not found in the user's `pages` directory. Interchange uses this; use it to supply some default pages so the user will not have them in their directory.

```
TemplateDir     /usr/local/interchange/default_pages
```

The user's page, if it exists, will take precedence. There is also a catalog−specific version of this directive. If a page is found in that directory (or directories), it will take precedence.

2.54. TolerateGet *global*

Set to 'Yes' to enable parsing of both GET data and POST data when a POST has been submitted. The default is 'No', which means that GET data is ignored during a POST. Unfortunately this has to be a global setting because at URL parse time, the Interchange daemon doesn't yet know which catalog it's dealing with (due to catalog aliases, etc.).

2.55. UrlSepChar *global*

Sets the character which separates form parameters in Interchange−generated URLs. Default is &.

2.56. Unix_Mode *global*

Determines whether the UNIX−domain socket will be monitored on startup. Overridden by the command−line parameter `−u`. Default is `Yes`.

2.57. UserTag *global*

This defines a UserTag which is global in nature, meaning not limited by the `Safe.pm` module, and is is available to all Interchange catalogs running on the server. Otherwise, this is the same as a catalog UserTag.

2.58. Variable *global*

Defines a global variable that will be available in all catalogs with the notation @@VARIABLENAME@@. Variable identifiers must begin with a capital letter, and can contain only word characters (**A−Z,a−z,0−9** and underscore). They are case−sensitive. If using the `ParseVariables` directive, only variables in ALL CAPS will be parsed. These are substituted first in any Interchange page, and can contain any valid Interchange tags including catalog variables.

```
Variable   DOCUMENT_ROOT   /usr/local/etc/httpd/htdocs
```

If a variable is called with @_VARIABLE_@, and there is no catalog Variable with its name, the global Variable value will be inserted.

There are several standard variables which should not be used:

MV_FILE

Name of the last file read in, as in `[file ...]` or an externally located perl routine.

MV_NO_CRYPT

Set this to 1 to disable encrypted passwords for the `AdminUser`.

MV_PAGE

Name of the last page read in, as in the page called with mv_nextpage or mv_orderpage.

CURRENCY, MV_CURRENCY

The current locale for currency.

LANG, MV_LANG

The current locale for language.

2.59. VarName *global*

Sets the names of variables that will be remapped to and from the URL when Interchange writes it. For instance, to display the variable `mv_session_id` as `session` in the users URL:

```
VarName   mv_session_id  session
```

The default can also be set in the `etc/varnames` file after the first time Interchange is run. Setting it in `interchange.cfg` is probably better for clarity.

There is also a catalog−specific version of this setting.

3. catalog.cfg

Each catalog must have a `catalog.cfg` file located in its base catalog directory. It contains most of the configurable parameters for Interchange. Each is independent from catalog to catalog.

Additional configuration techniques are available in the `catalog.cfg` file. First, set a `Variable` and use its results in a subsequent configuration setting if `ParseVariables` is on:

```
Variable   SERVER_NAME  www.akopia.com
Variable   CGI_URL      /cgi-bin/demo

ParseVariables Yes
VendURL    http://__SERVER_NAME____CGI_URL__
ParseVariables No
```

Define subroutine watches

Almost any configuration variable can be set up to be tied to a subroutine if the Tie::Watch module is installed. It uses a notation like the <<HERE document, but <&HERE is the notation. See Interchange Programming for details.

3.1. Programming Watch Points in catalog.cfg

Almost any configuration variable can be set up to be tied to a subroutine if the `Tie::Watch` module installed. It uses a notation like the <<HERE document, but <&HERE is the notation. Here is a simple case:

```
MailOrderTo orders@akopia.com
MailOrderTo <&EOF
sub {
    my($self, $default) = @_;
    if($Values->{special_handling}) {
        return 'vip@akopia.com';
    }
    else {
        return $default;
    }
}
EOF
```

When the order is mailed out, if the user has a variable called `special_handling` set in their session (from UserDB, perhaps), the order will be sent to 'vip@akopia.com.' Note the single quotes to prevent problems with the @ sign. Otherwise, the order will get sent to the previously defined value of `orders@akopia.com`.

If the configuration value being watched is a SCALAR, the subroutine gets the following call:

```
&{$subref}(SELF, PREVIOUS_VALUE)
```

The subroutine should simply return the proper value.

SELF is a reference to the Tie::Watch object (read its documentation for what all it can do) and `PREVIOUS_VALUE` is the previously set value for the directive. If set after the watch is set up, it will simply have the effect of destroying the watch and having unpredictable effects. (In the future, a "Store" routine may

be able to be set up that can subsequently set values).

If the configuration value being watched is an ARRAY, the subroutine gets the following call:

```
&{$subref}(SELF, INDEX, PREVIOUS_VALUE)
```

INDEX is the index of the array element being accessed. Setting up watch points on array values is not recommended. Most Interchange subroutines call arrays in their list context, and no access method is provided for that.

If the configuration value being watched is a HASH, the subroutine gets the following call:

```
&{$subref}(SELF, KEY, PREVIOUS_VALUE)
```

KEY is the index into the hash, an example of HASH type Interchange configuration values. NOTE: The following is not recommended for performance reasons. The Variable is a commonly used thing and should not bear the extra overhead of tieing, but it illustrates the power of this operation:

```
Variable TESTIT Unwatch worked.

Variable <&EOV
sub {
    my ($self, $key, $orig) = @_;
    if($key eq 'TESTIT') {
        # only the first time
        if($Scratch->{$key}++) {
            $self->Unwatch();
            return $orig->{TESTIT};
        }
        else {
            return "Tie::Watch works! -- name=$Values->{name}";
        }
    }
    else {
        return $orig->{$key};
    }
}
EOV
```

The first time __TESTIT__ is called for a particular user, it will return the string "Tie::Watch works! -- name=" along with their name set in the session (if that exists). Any other variables will receive the value that they were set to previously. Once the TESTIT key has been accessed for that user, the watch is dropped upon the next access.

3.2. Configuration Directives in catalog.cfg

All directives except MailOrderTo and VendURL have default values and are optional, though most catalogs will want to configure some of them.

3.3. ActionMap

Allows setting of Interchange actions, usually with a Perl subroutine. Actions are page names like:

```
process  Perform a processing function
```

```
order    Order items
scan     Search based on path info
search   Search based on submitted form variables
```

These are the standard supplied actions for Interchange. They can be overwritten with user−defined versions if desired. For example, to ignore the `order` action, set:

```
ActionMap  order  sub { return 1 }
```

When the leading part of the incoming path is equal to `order`, it will trigger an action. The page name will be shifted up, and the `order` stripped from the page name. So this custom `order` action would essentially perform a no−op, and a URL like:

```
<A HREF="[area order/nextpage]"> Go to the next page </A>
```

would be the equivalent of "[area nextpage]." If the action does not return a true (non−zero, non−blank) status, no page will be displayed by Interchange, not even the special `missing` page. A response may also be generated via Perl or MVASP.

The standard `process` action has a number of associated `FormAction` settings. Besides using Perl, Interchange tags may be used in an action, though they are not nearly as efficient.

3.4. AlwaysSecure

Determines whether checkout page operations should always be secure. Set it to the pages that should always be secure, separated by spaces and/or tabs.

```
AlwaysSecure    ord/checkout
```

3.5. AsciiTrack

A file name to log formatted orders in. Unless preceded by a leading '/', will be placed relative to the catalog directory. Disabled by default.

```
AsciiTrack     etc/tracking.asc
```

If a `Route` is set up to `supplant`, this is ignored.

3.6. Autoend

Sets an action that is automatically performed at the end every access. It is performed after any page parsing occurs, just before the transaction ends. See `Autoload`.

3.7. Autoload

Sets an action that is automatically performed for every access. It is performed before any page parsing occurs, and before the action or page is even determined. Can contain ITL tags or a global subroutine name. If the return value is true, a normal display of $CGI−>{mv_nextpage} will occur −− if it returns a false (zero, undef, or blank) value, no page will be processed.

As an example, to remap any `mv_nextpage` accesses to the `private` subdirectory of pages, set:

```
Autoload   [perl] $CGI->{mv_nextpage} =~ s:^private/:public/:; [/perl]
```

3.8. AutoModifier

Sets an attribute in a shopping cart entry to the field of the same name in the `ProductsFile` pertaining to this item. This is useful when doing shipping calculations or other embedded Perl that is based on item attributes. To set whether an item is defined as "heavy" and requires truck shipment, set:

```
AutoModifier  heavy
```

When an item is added to the shopping cart using Interchange's routines, the `heavy` attribute will be set to the value of the `heavy` field in the products database. In the default demo that would be `products`. Any changes to `ProductFiles` would affect that, of course.

Some values are used by Interchange and are not legal:

```
mv_mi
mv_si
mv_ib
group
code
quantity
item
```

3.9. AutoVariable

Specifies directives which should be translated to Variable settings. For scalars, the directive name becomes the Variable name and yields its value, i.e. `DescriptionField` becomes `__DescriptionField__`, which would by default be description. Array variables have a `_N` added, where `_N` is the ordinal index, i.e. ProductFiles becomes `__ProductFiles_0__`, `__ProductFiles_1__`, etc. Hash variables have a `_KEY` added, i.e. `SpecialPage` becomes `__SpecialPage_missing__`, `__SpecialPage_violation__`, etc. Doesn't handle hash keys that have non−word characters or whitespace. Only single−level arrays and hashes are translated properly.

3.10. CommonAdjust

Settings for Interchange pricing. See `Chained pricing`.

```
CommonAdjust    pricing:q2,q5,q10,q25, ;products:price, ==size:pricing
```

3.11. ConfigDir

The default directory where directive values will be read from when using the <file notation. Default is `config`. The name is relative to the catalog directory unless preceded by a /.

```
ConfigDir      variables
```

This can be changed several times in the `catalog.cfg` file to pick up values from more than one directory. Another possibility is to use a `Variable` setting to use different templates based on a setting:

```
Variable   TEMPLATE   blue

ParseVariables Yes
ConfigDir  templates/__TEMPLATE__
ParseVariables No
Variable   MENUBAR   <menubar
Variable   LEFTSIDE  <leftside
Variable   BOTTOM    <bottom
ConfigDir config
```

This will pick the `templates/blue` template. If TEMPLATE is set to `red`, it would read the variables from `templates/red`.

3.12. CookieDomain

Allows a domain to be set so that multiple servers can handle traffic. For example, to use server addresses of `secure.yourdomain.com` and `www.yourdomain.com`, set it to:

```
CookieDomain    .yourdomain.com
```

More than one domain can be set. It must have at least two periods or browsers will ignore it.

3.13. CookieLogin

Allows users to save their username/password (for Vend::UserDB) in a cookie. Expiration is set by SaveExpire and is renewed each time they log in. To cause the cookie to be generated originally, the CGI variable `mv_cookie_password` or `mv_cookie_username` must be set. The former causes both username and password to be saved; the latter just the username.

```
CookieLogin  Yes
```

Default is `No`.

3.14. Cookies

Determines whether Interchange will send (and read back) a cookie to get the session ID for links that go outside the catalog. Allows arbitrary HREF links to be placed in Interchange pages, while still saving the contents of the session. The default is Yes.

```
Cookies         Yes
```

If the Cookies directive is enabled, and `mv_save_session` is set upon submission of a user form (or in the CGI variables through a Perl `GlobalSub`), the cookie will be persistent for the period defined by `SaveExpire`.

Note: This should almost always be "Yes."

Caching, timed builds, and static page building will never be in effect unless this directive is enabled.

3.15. CreditCardAuto

If set to Yes, enables the automatic encryption and saving of credit card information. In order for this to work properly, the `EncryptProgram` directive must be set to properly encode the field. The best way to set `EncryptProgram` is with PGP in the ASCII armor mode. This option uses the following standard fields on Interchange order processing forms:

`mv_credit_card_number`

The actual credit card number, which will be wiped from memory after checking to see if it is a valid Amex, Visa, MC, or Discover card number. This variable will never be carried forward in the user session.

`mv_credit_card_exp_all`

The expiration date, as a text field in the form MM/YY (will take a four−digit year as well). If it is not present, the fields `mv_credit_card_exp_month` and `mv_credit_card_exp_year` are looked at. It is set by Interchange when the card validation returns, if not previously set.

`mv_credit_card_exp_month`

The expiration date month, used if the `mv_credit_card_exp_all` field is not present. It is set by Interchange when the card validation returns, if not previously set.

`mv_credit_card_exp_year`

The expiration date year, used if the `mv_credit_card_exp_all` field is not present. It is set by Interchange when the card validation returns, if not previously set.

`mv_credit_card_error`

Set by Interchange to indicate the error if the card does not validate properly. The error message is not too enlightening if validation is the problem.

`mv_credit_card_force`

Set this value to 1 to force Interchange to encrypt the card despite its idea of validity. Will still set the flag for validity to 0 if the number/date does not validate. Still won't accept badly formatted expiration dates.

`mv_credit_card_separate`

Set this value to 1 to cause Interchange encrypt only the card number and not accompany it with the expiration date and card type.

`mv_credit_card_info`

Set by Interchange to the encrypted card information if the card validates properly. If PGP is used in ASCII armor mode, this field can be placed on the order report and embedded in the order email, replete with markers. This allows a secure order to be read for content, without exposing the credit card number to risk.

`mv_credit_card_valid`

Set by Interchange to true, or 1, if the the card validates properly. Set to 0 otherwise.

PGP is recommended as the encryption program, though remember that U.S. commercial organizations may require a license for RSA. Interchange will work with GPG, the Gnu Privacy Guard.

```
CreditCardAuto     Yes
```

3.16. CustomShipping

If not blank, causes an error log entry if the shipping file entry is not found. Not otherwise used for shipping. See `SHIPPING` for how to go about doing that.

```
CustomShipping      Yes
```

3.17. Database

Definition of an arbitrary database, in the form "Database database file type," where "file" is the name of an ASCII file in the same format as the products database. The file is relative to VendRoot. Records can be accessed with the `[data database field key]` tag. Database names are restricted to the alphanumeric characters (including the underscore), and it is recommended that they be either all lower or all upper case. See `DATABASES`.

```
Database      reviews  reviews.txt  CSV
```

3.18. DatabaseDefault

Defines default parameters for a database. This can be used to set a default WRITE_CONTROL setting, set a default USER or PASSWORD, etc. It accepts any scalar setting, which means all **except**:

ALTERNATE_* BINARY COLUMN_DEF DEFAULT FIELD_ALIAS FILTER_* NAME NUMERIC POSTCREATE WRITE_CATALOG

This default setting is made when the table is initially defined, i.e. explicit settings for the database itself override the defaults set.

```
DatabaseDefault      WRITE_CONTROL   1
DatabaseDefault      WRITE_TAGGED    1
```

This setting must be made *before* the database is defined. To reset its value to empty, use the `Replace` directive.

```
Replace DatabaseDefault
```

3.19. DefaultShipping

This sets the default shipping mode by initializing the variable `mv_ship_mode`. If not set in `catalog.cfg`, it is default.

```
DefaultShipping     UPS
```

Somewhat deprecated, the same thing can be achieved with:

```
ValuesDefault   mv_shipmode UPS
```

3.20. DescriptionField

The field that will be accessed with the `[item-description]` element.

```
DescriptionField    description
```

Default is `description`. It is not a fatal error if this field does not exist. This is especially important for on-the-fly items. If there is an attribute set to the same name as `DescriptionField`, this will be used for display.

3.21. DirConfig

`DirConfig` allows you to batch-set a bunch of variables from files. The syntax:

DirConfig directive-name directory-glob

`directive-name` is usually `Variable`, but could be any hash-based directive. (No other standard directives currently make sense to set this way.)

`directory-glob` is a filespec that could encompass multiple directories. Files are ignored.

The directories are read for file *names* that contain only word characters, i.e. something that would be a valid Variable. (This alone might make it not suitable for other uses, but picking up the junk from the in-directory-backup-file people would be intolerable.)

Then the contents of the file is used to set the variable of the file name.

The source file name is kept in $Vend::Cfg->{DirConfig}{Variable}{VARNAME}, for use if dynamic_variables Pragma is set.

Pragma `dynamic_variables` enables dynamic updating of variables from files. Pragma `dynamic_variables_files_only` restricts dynamic variables to files only -- otherwise variables are dynamically read from the VarDatabase definition as well.

With dynamic variables, all @_VARIABLE_@ and __VARIABLE__ settings are checked first to see if the source file is defined. If there is a key present, even if its contents are blank, it is returned. Example: in the case of this catalog.cfg entry:

DirConfig Variable templates/foundation/regions

If the file NOLEFT_TOP is present at catalog config time, __NOLEFT_TOP__ will equal `[include templates/foundation/regions/NOLEFT_TOP]`.

3.22. DisplayErrors

If the administrator has enabled DisplayErrors globally, setting this to "Yes" will display the error returned from Interchange in case something is wrong with embedded Perl programs, tags, or Interchange itself. Usually, this will be used during development or debugging. Default is No.

```
DisplayErrors       Yes
```

3.23. DynamicData

When set to one or more Interchange database identifiers, any pages using data items from the specified database(s) will not be cached or built statically. This allows dynamic updating of certain arbitrary databases (even the products database) while still allowing static/cached page performance gains on pages not using those data items.

```
DynamicData         inventory
```

Overridden by `[tag flag build][/tag]`, depending on context.

3.24. EncryptProgram

Contains a program command line specification that indicates how an external encryption program will work. Two placeholders, `%p` and `%f`, are defined, which are replaced at encryption time with the password and temporary file name respectively. See `Order Security`. This is separate from the PGP directive, which enables PGP encryption of the entire order.

If PGP is the encryption program (Interchange determines this by searching for the string `pgp` in the command string), no password field or file field need be used. The field `mv_credit_card_number` will never be written to disk in this case.

```
EncryptProgram      /usr/local/bin/pgp −feat sales@company.com
```

If the order `Route` method of sending orders is used (default in the demo), this sets the default value of the `encrypt_program` attribute.

3.25. ErrorFile

This is where Interchange will write its runtime errors for THIS CATALOG ONLY. It can be shared with other catalogs or the main Interchange error log, but if it is root−based, permission to write the file is required.

```
ErrorFile   /home/interchange/error.log
```

3.26. ExtraSecure

Disallows access to pages which are marked with `AlwaysSecure` unless the browser is in HTTPS mode. A Yes/No directive, the default is 'No.'

```
ExtraSecure  Yes
```

3.27. Filter

Assigns one or more filters (comma separated) to be automatically applied to a variable.

As an example, multiple form variable submissions on the same page come back null−separated, like 'value1\0value2\0value3'. To automatically change those nulls to spaces, you could use this directive:

```
Filter  mail_list  null_to_space
```

Of course you could just as easily use the regular [filter] tag on the page if the filter is only going to be used in a few places.

See the ictags document for more information, including a complete list of filters.

3.28. FormAction

Allows set up of a form action (like the standard ones `return, submit, refresh,` etc.). It requires a Perl subroutine as a target:

```
FormAction foo <<EOR
sub {
    $CGI−>{mv_nextpage} = 'bar';
}
EOR
```

If it returns a true (non−zero, non−empty) value, Interchange will display the page defined in $CGI−>{mv_nextpage}. Otherwise, Interchange will not display any page. The default Interchange actions can be overridden if desired. There is also a global version of this directive, which is overridden if a catalog−specific action exists.

3.29. FormIgnore

Set to the name(s) of variables that should not be carried in the user session values. Must match exactly and are case sensitive.

```
FormIgnore    mv_searchtype
```

3.30. FractionalItems

Whether items in the shopping cart should be allowed to be fractional, i.e., 2.5 or 1.25. Default is No.

```
FractionalItems     Yes
```

3.31. Glimpse

The pathname for the `glimpse` command, used if `glimpse` searches are to be enabled. To use `glimpseserver`, the `−C`, `−J`, and `−K` tags must be used.

```
Glimpse  /usr/local/bin/glimpse −C −J srch_engine −K2345
```

3.32. History

How many of the most recent user clicks should be stored in the session history. Default is 0.

3.33. HTMLsuffix

The file extension that will be seen as a page in the `pages` directory. Default is `.html`.

```
HTMLsuffix .htm
```

3.34. ImageAlias

Aliases for images, ala Apache/NCSA, ScriptAlias, and Alias directives. Relocates images based in a particular directory to another for Interchange use; operates after `ImageDir`. Useful for editing Interchange pages with an HTML editor. Default is blank.

```
ImageAlias  /images/  /thiscatalog/images/
```

3.35. ImageDir

The directory where all relative IMG and INPUT source file specifications are based. IT MUST HAVE A TRAILING / TO WORK. If the images are to be in the `DocumentRoot` (of the HTTP server or virtual server) subdirectory images, for example, use the `ImageDir` specification '/images/'. This would change SRC="order.gif" to SRC="/images/order.gif" in IMG and INPUT tags. It has no effect on other SRC tags.

```
ImageDir /images/
```

Can be set in the Locale settings to allow different image sets for different locales (MV3.07 and up).

3.36. ImageDirInternal

A value for `ImageDir` only when the internal HTTP server is in use. It must have a trailing / to work, and should always begin with a fully−qualified path starting with `http://`.

```
ImageDirInternal http://www.server.name/images/
```

3.37. ImageDirSecure

A value for `ImageDir` only when the pages are being served via HTTPS. It must have a trailing / to work, and should always begin with a fully−qualified path starting with `http://`.

```
ImageDirSecure   /secure/images/
```

This is useful if using separate HTTPS and HTTP servesr, and cannot make the image directory path heads match.

3.38. Locale

Sets the special locale array. Tries to use POSIX `setlocale` based on the value of itself, then tries to accept a custom setting with the proper definitions of `mon_decimal_point`, `thousands_sep`, and `frac_digits`, which are the the only international settings required. Default, if not set, is to use US−English settings.

Example of the custom setting:

```
Locale     custom mon_decimal_point , mon_thousands_sep . frac_digits 0
```

Example of POSIX setlocale for France, if properly aliased:

```
Locale     fr
```

See `setlocale(3)` for more information. If embedded Perl code is used to sort search returns, the `setlocale()` will carry through to string collation.

See Internationalization.

3.39. LocaleDatabase

Set to the Interchange database identifier of a table that contains `Locale` settings. These settings add on to and overwrite any that are set in the catalog configuration files, including any include files.

```
Database       locale  locale.asc  TAB
LocaleDatabase locale
```

3.40. MailOrderTo

Specifies the e−mail address to mail completed orders to.

```
      MailOrderTo  orders@xyzcorp.com
```

If 'none' is specified, no e−mailed order will be sent.

3.41. NoCache

The names of Interchange pages that are not to be built statically if STATIC PAGE BUILDING is in use. If the name is a directory, no pages in that directory (or any below it) will be cached or built statically.

```
NoCache    ord
NoCache    special
```

3.42. NoImport

When set to one or more Interchange database identifiers, those database(s) will never be subject to import. Useful for SQL databases or databases that will "never" change.

```
NoImport   inventory
```

3.43. NonTaxableField

The name of the field in the products database that is set (to 1 or Yes) if an item is not to be taxed. Interchange will log an error and tax it anyway if the field doesn't exist in the database. Blank by default, disabling the feature.

```
NonTaxableField    wholesale
```

3.44. OfflineDir

The location of the offline database files for use with the Interchange offline database build command. Set to "offline" as the default, and is relative to VendRoot if there is no leading slash.

```
OfflineDir          /usr/data/interchange/offline
```

3.45. OnFly

Enables on−the−fly item additions to the shopping cart. If set to the name of a valid UserTag, that tag definition will be used to parse and format the item with the following call:

```
$item = Vend::Parse::do_tag($Vend::Cfg->{OnFly},
                            $code,
                            $quantity,
                            $fly[$j],
                          );
```

$fly[$j] is the value of `mv_order_fly` for that item. An `onfly` tag is provided by Interchange. See <On−the−fly> ordering.

3.46. OrderCounter

The name of the file (relative to catalog root if no leading /) that maintains the order number counter. If not set, the order will be assigned a string based on the time of the order and the user's session number.

```
OrderCounter       etc/order.number
```

Bear in mind that Interchange provides the order number as a convenience for display, and that no internal functions depend on it. Custom order number routines may be defined and used without fear of consequences.

If a `Route` is set up to `supplant` and the `counter` attribute is set there, this is ignored.

3.47. OrderLineLimit

The number of items that the user is allowed to place in the shopping cart. Some poorly−mannered robots may "attack" a site by following all links one after another. Some even ignore any `robots.txt` file that may have been created. If one of these bad robots orders several dozen or more items, the time required to save and restore the shopping cart from the user session may become excessive.

If the limit is exceeded, the command defined in the Global directive `LockoutCommand` will be executed and the shopping cart will be emptied. The default is 0, disabling the check. Set it to a number greater than the number of line items a user is ever expected to order.

```
OrderLineLimit   50
```

3.48. OrderProfile

Allows an unlimited number of profiles to be set up, specifying complex checks to be performed at each of the steps in the checkout process. The files specified can be located anywhere. If relative paths are used, they are relative to the catalog root directory.

```
OrderProfile    etc/profiles.order etc/profiles.login
```

The actions defined here are also used for `mv_click` actions if there is no action defined in `scratch` space. They are accessed by setting the `mv_order_profile` variable to the name of the order profile. Multiple profiles can reside in the same file, if separated by `__END__` tokens, which must be on a line by themselves.

The profile is named by placing a name following a `__NAME__` pragma:

```
__NAME__ billing
```

The `__NAME__` must begin the line, and be followed by whitespace and the name. The search profile can then be accessed by <mv_order_profile="billing">. See Advanced Multi−level Order Pages.

3.49. OrderReport

The location of the simple order report file. Defaults to etc/report.

```
OrderReport          /data/order−form
```

3.50. PageDir

Location of catalog pages. Defaults to the pages subdirectory in the VendRoot directory.

```
PageDir    /data/catalog/pages
```

Can be set in the `Locale` settings to allow different page sets for different locales.

3.51. PageSelectField

Sets a products database column which can be used to select the on−the−fly template page. This allows multiple on−the−fly pages to be defined. If the field is empty (no spaces), the default `flypage` will be used.

```
PageSelectField    display_page
```

3.52. ParseVariables

Determines whether global and catalog variables will be parsed in the configuration file. Default is No. The foundation catalog.cfg turns ParseVariables on and usually expects it to be on.

```
Variable STORE_ID  topshop
ParseVariables Yes
StaticDir  /home/__STORE_ID__/www/cat
ParseVariables No
```

3.53. Password

The encrypted or unencrypted password (depending on Variable MV_NO_CRYPT) that will cause internal authorization checks for RemoteUser to allow access.

Below is the encrypted setting for a `blank` password.

```
Password                bAWoVkuzphOX.
```

3.54. PGP

If credit card information is to be accepted, and the e−mailed order will go over an insecure network to reach its destination, PGP security should be used. The key ring to be used must be for the user that is running the Interchange server, or defined by the environment variable `PGPPATH`, and the key user specified must have a key on the public key ring of that user.

```
PGP        /usr/local/bin/pgp −feat orders@company.com
```

If this directive is non−null, the PGP command string as specified will be used to encrypt the entire order in addition to any encryption done as a result if `CreditCardAuto`. If, for some reason, an error comes from PGP, the customer will be given the special page `failed`.

If a `Route` is set up to `supplant`, this is ignored.

3.55. Pragma

Sets the default value of an Interchange pragma. The directive is set like this:

```
Pragma my_pragma_name
```

To enable a pragma for only a particular page, set it anywhere in the page:

```
[pragma my_pragma_name]
```

To disable a pragma for a particular page, set it anywhere in the page:

```
[pragma my_pragma_name 0]
```

Descriptions of each pragma follow.

dynamic_variables

dynamic_variables_file_only

no_html_parse

Disallows HTML tag parsing. This is a **big** parser performance gain and is enabled in the demo catalog.

When this pragma is set, you can't encapsulate Interchange tags inside HTML tags like this:

```
<P MV="if scratch something"> ... </P>
```

Note that a page with no HTML parsing is a good place to put a DTD (document type descriptor).

no_image_rewrite

Prevents image locations in pages from being altered by Interchange. Added in Interchange 4.7.0.

Interchange normally rewrites image locations to point to ImageDir. This applies to image locations mentioned in <img src="...">, <input src="...">, <body background="...">, <table background="...">, and <tr/th/td background="...">.

When this pragma is **not** set, the following tag:

```
<img src="fancy.gif">
```

Would, assuming an ImageDir set to /foundation/images, be transformed into:

```
<img src="/foundation/images/fancy.gif">
```

When pragma no_image_rewrite **is** set, the <img> tag would remain unchanged.

safe_data

By default Interchange does not allow data returned from databases to be reparsed for Interchange tags. Setting the safe_data pragma eliminates this restriction.

If for some reason you want to have tags in your database, for example, to use [page ...] for catalog−internal hyperlinks in your product descriptions, you need to enable safe_data. Some things to consider:

1. It may be better to use the safe_data attribute available to certain tags instead of the pragma, or perhaps to use [pragma] for a whole page or [tag pragma] ... [/tag] for a small block, instead of a catalog−wide Pragma directive.
2. In any case it is strongly recommended that you surround the area with [restrict] ... [/restrict] tags to allow only the specific (hopefully relatively safe) set of tags you expect to appear, such as [page] or [area]. Expect security compromises if you allow [calc] or [perl], or other extremely powerful tags.
3. Be certain that you know everywhere the data in your database will be used. Will it always be possible to reparse for tags? What about when it's used to create an emailed plain−text receipt −− will a literal '[page ...]' tag show up in the product description on the receipt? Would the desired output of '<a href="...">' be any better in a plaintext situation? What if you access your database from applications other than Interchange? You'll then have to decide what to do with such tags; perhaps you can simply strip them, but will the missing tag output cause you any trouble?

In short, safe_data is disabled by default for a reason, and you should be very careful if you decide to enable it.

(Watch out for parse order with [tag pragma] or [restrict] when used with lists that retrieve data from the database, as in [prefix−*] and the flypage. Loops parse before regular tags like [tag] and [restrict], and thus aren't affected by it.)

strip_white

Set this to strip whitespace from the tops of HTML pages output by Interchange. Such whitespace usually comes from Interchange tags at the top of the page. The pragma's purpose is mostly to make 'view source' in the browser a slightly more tolerable experience.

Default is off; whitespace is unchanged.

3.56. PriceCommas

If no commas are desired in price numbers (for the `[item−price]` tag), set this to `No`. The default is to use commas (or whatever is the thousands separator for a locale).

```
PriceCommas         no
```

This is overridden if a Locale `price_picture` is set.

3.57. PriceDivide

The number the price should be divided by to get the price in units (dollars or such). The default is one. If penny pricing is used, set it to 100.

```
PriceDivide         100
```

Can be set in the `Locale` settings to allow a price adjustment factor for different currencies.

3.58. PriceField

The field in the product database that will be accessed with the `[item−price]` element. Default is "price."

```
PriceField          ProductPrice
```

Can be set in the `Locale` settings to allow different price fields for different currencies.

3.59. ProductDir

Location of the database files. Defaults to the products subdirectory of the VendRoot directory. May not be set to an absolute directory unless `NoAbsolute` is defined as No.

```
ProductDir          /data/catalog/for−sale
```

Most people never set this directive and use the default of `products`.

3.60. ProductFiles

Database tables that should be seen as the "products" database.

```
ProductFiles    vendor_a vendor_b
```

The key thing about this is that each will be searched in sequence for a product code to order or an `[item-field ....]` or `[loop-field ...]` to insert. The main difference between `[item-field ....]` and `[item-data table ...]` is this fall-through behavior.

Default is `products`.

3.61. ReadPermission and WritePermission

By default, only the user account that Interchange runs under (as set by the SETUID permission on vlink) can read and write files created by Interchange. `WritePermission` and `ReadPermission` can be set to `user`, `group`, or 'world'.

```
ReadPermission      group
WritePermission     group
```

3.62. RemoteUser

The value of the HTTP environment variable `REMOTE_USER` that will enable catalog reconfiguration. HTTP basic authentication must be enabled for this to work. Default is blank, disabling this check.

```
RemoteUser   interchange
```

3.63. Replace

Causes a directive to be emptied and re-set (to its default if no value is specified). Useful for directives that add to the value by default.

```
Replace NoCache ord special multi reconfig query
```

Capitalization must be exact on each directive.

3.64. Require

Forces a Perl module, global `UserTag`, or `GlobalSub` to be present before the catalog will configure. This is useful when transporting catalogs to make sure they will have all needed facilities.

```
Require usertag   email
Require globalsub form_mail
Require module    Business::UPS
```

3.65. RobotLimit

The RobotLimit directive defines the number of consecutive pages a user session may access without a 30 second pause. If the limit is exceeded, the command defined in the Global directive `LockoutCommand` will be executed and catalog URLs will be rewritten with host 127.0.0.1, sending the robot back to itself. The default is 0, disabling the check.

```
RobotLimit  200
```

3.66. Route

Sets up order routes. See `Custom Order Routing`. There are examples in the demo `simple`.

3.67. SalesTax

If non−blank, enables automatic addition of sales tax based on the order form. The value is a comma−separated list of the field names (as placed in order.html) in priority order, which should be used to look up sales tax percentage in the `salestax.asc` database. This database is not supplied with Interchange. It is typically received from a third party by quarterly or monthly subscription.

```
SalesTax            zip state
```

3.68. SalesTaxFunction

A Perl subroutine that will return a hash reference with the sales tax settings. This can be used to query a database for the tax for a particular vendor:

```
SalesTaxFunction  <<EOR
      my $vendor_id = $Session->{source};
    my $tax = $TextSearch->hash( {
                        se => $vendor_id,
                        fi => 'salestax.asc',
                        sf => 'vendor_code',
                        ml => 1000,
                        } );
    $tax = {} if ! $tax;
    $tax->{DEFAULT} = 0.0;
    return $tax;
EOR
```

or simply produce a table:

```
SalesTaxFunction  <<EOR
 return {
     DEFAULT => 0.0,
     IL => 0.075,
     OH => 0.065,
 };
EOR
```

A `DEFAULT` value must always be returned or the function will be ignored.

3.69. SaveExpire

The default amount of time that a cookie will be valid (other than the MV_SESSION_ID cookie). The ones used in Interchange by default are `MV_USERNAME` and `MV_PASSWORD` for the `CookieLogin` feature. Specified the same as `SessionExpire`, with an integer number followed by one of `minutes`, `hours`, `days`, or `weeks`.

```
SaveExpire 52 weeks
```

Default is `30 days`.

3.70. ScratchDefault

The default scratch variable settings that the user will start with when their session is initialized. To disable placing URL rewrite strings after the user has given a cookie, set:

```
ScratchDefault  mv_no_session_id  1
ScratchDefault  mv_no_count       1
ScratchDefault  mv_add_dot_html   1
```

3.71. ScratchDir

The directory where temporary files will be written, notably cached searches and retired session IDs. Defaults to `tmp` in the catalog directory.

```
ScratchDir          /tmp
```

3.72. SearchProfile

Allows an unlimited number of search profiles to be set up, specifying complex searches based on a single click. The directive accepts a file name based in the catalog directory if the path is relative:

```
SearchProfile    etc/search.profiles
```

As an added measure of control, the specification is evaluated with the special Interchange tag syntax to provide conditional setting of search parameters. The following file specifies a dictionary−based search in the file 'dict.product':

```
__NAME__ dict_search
mv_search_file=dict.product
mv_return_fields=1
[if value fast_search]
  mv_dict_limit=-1
  mv_last=1
[/if]
__END__
```

The `__NAME__` is the value to be specified in the `mv_profile` variable on the search form, as in

```
<INPUT TYPE=hidden NAME=mv_profile VALUE="dict_search">
```

or with mp=profile in the one−click search.

```
[page scan se=Renaissance/mp=dict_search]Renaissance Art[/page]
```

Multiple profiles can reside in the same file, if separated by __END__ tokens. __NAME__ tokens should be left−aligned, and __END__ must be on a line by itself with no leading or trailing whitespace.

3.73. SecureURL

The base URL for secure forms/page transmissions. Normally it is the same as VendURL except for the `https:` protocol definition. Default is blank, disabling secure access.

```
SecureURL   https://machine.com/xyzcorp/cgi−bin/vlink
```

3.74. SendMailProgram

The location of the sendmail binary, needed for mailing orders. Must be found at startup. This often needs to be set for FreeBSD or BSDI.

```
SendMailProgram    /usr/sbin/sendmail
```

If set to `none`, no mail can be sent by standard Interchange facilities. The default is the value in `interchange.cfg` and varies depending on operating system.

3.75. SeparateItems

Changes the default when ordering an item via Interchange to allowing multiple lines on the order form for each item. The default, `No`, puts all orders with the same part number on the same line.

Setting `SeparateItems` to `Yes` allows the item attributes to be easily set for different instances of the same part number, allowing easy setting of things such as size or color.

```
SeparateItems       Yes
```

Can be overridden with the `mv_separate_items` variables (both scratch and values).

3.76. SessionDatabase

When storing sessions, specify the name of the directory or DBM file to use. The file extensions of .db or .gdbm (depending on the DBM implementation used) will be appended. If the default file−based sessions are used, it is the name of the directory.

```
SessionDatabase     session−data
```

Can be an absolute path name, if desired.

It is possible for multiple catalogs to share the same session file, as well as for multiple Interchange servers to serve the same catalogs. If serving a extremely busy store, multiple parallel Interchange servers can share the same NFS−based file system and serve users in a "ping−pong" fashion using the file−based sessions. On huge

systems, the level of directory hashing may be changed. By default, only 48 * 48 hashing is done. See the source for `SessionFile.pm`.

3.77. SessionDB

The name of the Interchange database to be used for sessions if DBI is specified as the session type. This is not recommended.

3.78. SessionExpire

A customer can exit the browser or leave the catalog pages at any time, and no indication is given to the web server aside from the lack of further requests that have the same session ID. Old session information needs to be periodically expired. The `SessionExpire` specifies the minimum time to keep track of session information. Defaults to one day. Format is an integer number, followed by s(econds), m(inutes), h(ours), d(ays), or w(eeks).

```
SessionExpire       20 minutes
```

If `CookieLogin` is in use, this can be a small value. If the customer's browser has the Interchange session cookie stored, he/she will be automatically logged back in with the next request. Note, however, that the customer's cart and session values will be reset.

3.79. SessionLockFile

The file to use for locking coordination of the sessions.

```
SessionLockFile     session-data.lock
```

This only applies when using DBM-based sessions. It is possible for multiple catalogs to share the same session file. `SessionDatabase` needs to be set appropriately if the database is to be shared. Defaults to `session.lock`, which is appropriate for separate session files (and therefore standalone catalogs). Can be an absolute path name, if desired.

3.80. SessionType

The type of session management to be used. Use one of the following:

```
DB_File     Berkeley DB
DBI         DBI (don't use this, normally)
File        File-based sessions (the default)
NFS         File-based sessions, forces use of fcntl locking
GDBM        GDBM
```

The default is file-based sessions, which provides the best performance and reliablility in most environments.

If you are planning on running Interchange servers with an NFS-mounted filesystem as the session target, you must set SessionType to "NFS". The other requisites are usually:

1. fcntl() supported in Perl 2. lock daemon running on NFS server system 3. lock daemon running on Interchange server

See also the global directive LockType.

3.81. SpecialPage

Sets a special page to other than its default value. Can be set as many times as necessary. Will have no effect if not one of the Interchange Required Pages.

```
SpecialPage         checkout ord/checkout
SpecialPage         failed special/error_on_order
SpecialPage         interact special/browser_problem
SpecialPage         noproduct special/no_product_found
SpecialPage         order  ord/basket
SpecialPage         search srch/results
```

3.82. SpecialPageDir

The directory where special pages are kept. Defaults to `special_pages` in the catalog directory.

```
SpecialPageDir      pages/special
```

3.83. Static

A Yes/No directive. Enables static page building and display features. Default is `No`.

```
Static   Yes
```

3.84. StaticAll

A Yes/No directive. Tells Interchange to try and build all pages in the catalog statically when called with the static page build option. This is subject to the settings of `StaticFly`, `StaticPath`, and `NoCache`. Default is No. Pages that have dynamic elements will not be built statically, though that may be overridden with `[tag flag build][/tag]` on the page in question.

```
StaticAll   Yes
```

3.85. StaticDepth

The number of levels of static search building that will be done if a search results page contains a search. Default is one, though it could be very long if set higher. Set to 0 to disable re−scanning of search results pages.

```
StaticDepth 2
```

3.86. StaticDir

The absolute path of the directory which should be used as the root for static pages. The user ID executing Interchange must have write permission on the directory (and all files within) if this is to work.

```
StaticDir   /home/you/www/catalog
```

3.87. StaticFly

A Yes/No directive. If set to `Yes`, static builds will attempt to generate a page for every part number in the database using the on−the−fly page build capability. If pages are already present with those names, they will be overwritten. The default is `No`.

```
StaticFly   Yes
```

3.88. StaticPage

Tells Interchange to build the named page (or pages, whitespace separated) when employing the static page−building capability of Interchange. Not necessary if using `StaticAll`.

```
StaticPage   info/about_us  info/terms_and_conditions
```

3.89. StaticPath

The path (relative to HTTP document root) which should be used in pages built with the static page−building capability of Interchange.

```
StaticPath    /catalog
```

3.90. StaticPattern

A perl regular expression which is used to qualify pages that are to be built statically. The default is blank, which means all pages qualify.

```
StaticPattern  ^info|^help
```

3.91. StaticSuffix

The extension to be appended to a normal Interchange page name when building statically. Default is `.html`. Also affects the name of pages in the Interchange page directory. If set `to .htm`, the pages must be named with that extension.

```
StaticSuffix   .htm
```

3.92. Sub

Defines a catalog subroutine for use by the `[perl][/perl]` or [mvasp] embedded perl languages. Use the "here document" capability of Interchange configuration files to make it easy to define:

```
Sub <<EOF
sub sort_cart_by_quantity {
    my($items) = @_;
    $items = $Items if ! $items;
    my $out = '<TABLE BORDER=1>';
    @$items = sort { $a->{quantity} <=> $b->{quantity} } @$items;
    foreach $item (@$items) {
        my $code = $item->{code};
```

```
        $out .= '<TR><TD>';
        $out .= $code;
        $out .= '</TD><TD>';
        $out .= $Tag->data('products', 'name', $code);
        $out .= '</TD><TD>';
        $out .= $Tag->data('products', 'price', $code);
        $out .= '</TD></TR>';
    }
    $out .= '&lt/TABLE>';
    return $out;
}
EOF
```

As with Perl "here documents," the EOF (or other end marker) must be the ONLY thing on the line, with no leading or trailing white space. Do not append a semicolon to the marker. The above would be called with:

```
[perl]
    my $cart = $Carts->{main};
    return sort_cart($cart);
[/perl]
```

and will display an HTML table of the items in the current shopping cart, sorted by the quantity. Syntax errors will be reported at catalog startup time.

Catalog subroutines may not perform unsafe operations. The Safe.pm module enforces this unless global operations are allowed for the catalog. See `AllowGlobal`.

3.93. Suggests

Generates a warning message when a Perl module, global `UserTag`, or `GlobalSub` is not present at catalog configuration time. Same as the `Require` directive except not fatal.

```
Suggest usertag   table_editor
Suggest globalsub file_info
Suggest module    Business::UPS
```

3.94. TableRestrict

Used to provide "views" in database−based searches. Does not affect the text searches. Affects the table being searched.

Takes the form of `field=session_param`, where `field` is a column in the table being iterated over, and `session_param` is a `$Session` key (i.e., [data session username]).

```
TableRestrict  products  owner=username
```

The above would prevent the database search from returning any records except those where the column `owner` contains the current value of `[data session username]`.

Probably most usefully set by embedded Perl code in certain situations. For example:

```
[calc]
    # Restrict edit to owned fields
    $Config->{TableRestrict}{products} = 'owner=username';
```

```
    return;
[/calc]
```

When using SQL−based databases, in effect it turns the base search query

```
select * from products
```

into

```
select * from products where owner = '[data session username]'
```

Interchange databases are similarly affected, though the methodology is different. Also may be useful in "mall" situations, where user is allowed to only see products from the current store ID.

3.95. TaxShipping

A comma or space−separated list of states or jurisdictions that tax shipping cost, i.e., UT. Blank by default, never taxing shipping.

```
TaxShipping         UT,NV,94024
```

3.96. TrackFile

Name of a logfile that tracks user traffic. This is used in the back office administration report on traffic by affiliate.

The default is that no such file is kept.

3.97. UpsZoneFile

The file containing the UPS zone information, specified relative to the catalog directory unless it begins with a /. It can be in the format distributed by UPS or can be in a tab−delimited format, with the three−letter zip prefix of the customer used to determine the zone. It interpolates based on the value in `mv_shipmode`. A user database named the same as the mv_shipmode variable must be present or the lookup will return zero.

IMPORTANT NOTE: Zone information and updated pricing from UPS must be obtained in order for this to work properly. The zone information is specific to a region!

```
UpsZoneFile         /usr/interchange/data/ups_zone.asc
```

3.98. UseModifier

Determines whether any attributes, the modifiers specified in the directive, can be attached to the item. See `Item Attributes`. The default is no modifier. Don't use a value of `quantity` or this directive will not work properly.

```
UseModifier         size,color
```

Some values are used by Interchange and are not legal:

```
mv_mi
mv_si
mv_ib
group
code
quantity
item
```

3.99. ValuesDefault

Sets the initial state of the user values, i.e., [value key] or $Values−>{key}.

```
ValuesDefault   fname  New
ValuesDefault   lname  User
```

When the user session starts, `[value fname] [value lname]` will be "New User."

3.100. Variable

Defines a catalog variable that will be available in the current catalog with the notation __`Variable`__. Variable identifiers must begin with a capital letter, and can contain only word characters (**A−Z,a−z,0−9** and underscore). These are substituted second (right after global Variables) in any Interchange page, and can contain any valid Interchange tags except global variables.

```
Variable   DOCUMENT_ROOT   /usr/local/etc/httpd/htdocs
```

3.101. VariableDatabase

The name of a database containing a field Variable which will be used to set Interchange variable values. For example, a database defined as:

```
Database  var var.txt TAB
VariableDatabase var
```

and containing

```
code    Variable
HELLO   Hi!
```

would cause __`HELLO`__ to appear as `Hi!`.

The field name is case−sensitive, and `variable` would not work.

The values are inserted at time of definition. Any single−level hash−oriented Interchange directive, such as `SpecialPage`, `ScratchDefault`, or `ValuesDefault`, can be set in the same way. If the `VariableDatabase` named does not exist at definition time, a database of the default type with an ASCII file source appending `.txt` is assumed. In other words:

```
VariableDatabase variable
```

is equivalent to

```
Database         variable variable.txt TAB
VariableDatabase variable
```

3.102. VendURL

Specifies the base URL that will run vlink as a cgi−bin program.

```
VendURL  http://machine.company.com/cgi-bin/vlink
```

3.103. WideOpen

Disables IP qualification of user sessions. **This degrades catalog security.** Do not use unless using encryption or a real−time payment gateway.