Interchange: Interchange Configuration

Table of Contents

1. CONFIGURATION OF AKOPIA INTERCHANGE TM 1

2. CREATING A CATALOG 3
- 2.1. Create a Database 3
- 2.2. Features to Use 4
- 2.3. Use the Demo Catalogs 4
- 2.4. Tree design 5
- 2.5. The Essentials 5
- 2.6. The Demo Catalog 5
- 2.7. More About "simple" 5
- 2.8. IML − Interchange Markup Language 9
- 2.9. HTML Hypertext links 11
- 2.10. Images 11
- 2.11. Browser Cookies 11
- 2.12. Basic Interchange Tags 11
- 2.13. Inserting Hyperlinks 12
- 2.14. How to Order an Item 13
- 2.15. Accessing Form Values 15
- 2.16. The Catalog Directory 15
- 2.17. Setting Up Multiple Catalogs 16

3. INTERNATIONALIZATION 19
- 3.1. Setting the Locale 19
- 3.2. Interchange Locale Settings 20
- 3.3. Special Locale Keys for Price Representation 21
- 3.4. Dynamic Locale Directive Changes 22
- 3.5. Sorting Based on Locale 24
- 3.6. Placing Locale Information in a Database 25

4. PROGRAMMING 27
- 4.1. Embedded Perl Code 27
- 4.2. ASP−Like Perl 29

5. INTERCHANGE PERL OBJECTS 31

6. INTERCHANGE CONFIGURATION FILES 41

7. INTERCHANGE.CFG 43
- 7.1. ActionMap *global* 43
- 7.2. AddDirective *global* 43
- 7.3. AdminSub *global* 43
- 7.4. AllowGlobal *global* 43
- 7.5. Catalog *global* 43
- 7.6. Catch 44
- 7.7. CheckHTML *global* 44
- 7.8. ConfigAllAfter *global* 44
- 7.9. ConfigAllBefore *global* 44

Table of Contents

7.10. Counter....45
7.11. Discount−Subtotal....45
7.12. DisplayErrors *global*....45
7.13. DomainTail *global*....45
7.14. DumpStructure *global*....45
7.15. Environment *global*....46
7.16. ErrorFile *global*....46
7.17. Export....46
7.18. Filter....46
7.19. Fly−Tax....46
7.20. FormAction *global*....46
7.21. FullUrl *global*....47
7.22. GlobalSub *global*....47
7.23. HammerLock *global*....47
7.24. Handling....47
7.25. HouseKeeping *global*....48
7.26. IpHead *global*....48
7.27. IpQuad *global*....48
7.28. Legacy *global*....48
7.29. Locale *global*....48
7.30. LockoutCommand *global*....48
7.31. Log....49
7.32. Mall *global*....49
7.33. MaxServers *global*....49
7.34. NoAbsolute *global*....49
7.35. PIDcheck *global*....49
7.36. PIDfile *global*....50
7.37. SafeUntrap *global*....50
7.38. SendMailProgram *global*....50
7.39. SocketFile *global*....50
7.40. SocketPerms *global*....50
7.41. SubCatalog *global*....51
7.42. SysLog *global*....51
7.43. Tag....51
7.44. TcpHost *global*....51
7.45. TcpMap *global*....51
7.46. TemplateDir *global*....52
7.47. Time....52
7.48. Try....52
7.49. UserTag *global*....53
7.50. Variable *global*....53
7.51. VarName *global*....53

8. CATALOG.CFG....55
8.1. Programming Watch Points in Catalog.cfg....56
8.2. Configuration Directives in Catalog.cfg....57
8.3. ActionMap....58

Table of Contents

8.4. AlwaysSecure....58
8.5. AsciiTrack....58
8.6. AutoModifier....58
8.7. Autoload....59
8.8. CommonAdjust....59
8.9. ConfigDir....59
8.10. CookieDomain....60
8.11. CookieLogin....60
8.12. Cookies....60
8.13. CreditCardAuto....60
8.14. CustomShipping....62
8.15. CyberCash....62
8.16. Database....62
8.17. DefaultShipping....62
8.18. DescriptionField....62
8.19. DisplayErrors....62
8.20. DynamicData....63
8.21. EncryptProgram....63
8.22. ErrorFile....63
8.23. ExtraSecure....63
8.24. FormAction....63
8.25. FormIgnore....64
8.26. FractionalItems....64
8.27. Glimpse....64
8.28. HTMLsuffix....64
8.29. ImageAlias....64
8.30. ImageDir....64
8.31. ImageDirInternal....65
8.32. ImageDirSecure....65
8.33. Locale....65
8.34. LocaleDatabase....65
8.35. MailOrderTo....66
8.36. NoCache....66
8.37. NoImport....66
8.38. NonTaxableField....66
8.39. OfflineDir....66
8.40. OnFly....66
8.41. OrderCounter....67
8.42. OrderLineLimit....67
8.43. OrderProfile....67
8.44. OrderReport....68
8.45. PageDir....68
8.46. PageSelectField....68
8.47. ParseVariables....68
8.48. Password....68
8.49. PGP....68
8.50. Pragma....69

Table of Contents

8.51. PriceCommas....69
8.52. PriceDivide....69
8.53. PriceField....69
8.54. ProductDir....70
8.55. ProductFiles....70
8.56. ReadPermission and WritePermission....70
8.57. RemoteUser....70
8.58. Replace....70
8.59. Require....71
8.60. RobotLimit....71
8.61. Route....71
8.62. SalesTax....71
8.63. SalesTaxFunction....71
8.64. SaveExpire....72
8.65. ScratchDefault....72
8.66. ScratchDir....72
8.67. SearchProfile....72
8.68. SecureURL....73
8.69. SendMailProgram....73
8.70. SeparateItems....73
8.71. SessionDatabase....73
8.72. SessionDB....74
8.73. SessionExpire....74
8.74. SessionLockFile....74
8.75. SessionType....74
8.76. SpecialPage....75
8.77. SpecialPageDir....75
8.78. Static....75
8.79. StaticAll....75
8.80. StaticDepth....75
8.81. StaticDir....75
8.82. StaticFly....76
8.83. StaticPage....76
8.84. StaticPath....76
8.85. StaticPattern....76
8.86. StaticSuffix....76
8.87. Sub....76
8.88. TableRestrict....77
8.89. TaxShipping....78
8.90. UpsZoneFile....78
8.91. UseModifier....78
8.92. ValuesDefault....78
8.93. Variable....79
8.94. VariableDatabase....79
8.95. VendURL....79
8.96. WideOpen....79

Table of Contents

9. ADMINISTERING INTERCHANGE 81
9.1. Starting, Stopping, and Re−starting the Servers 81
9.2. UNIX and INET modes 82
9.3. User Reconfiguration 82
9.4. Making the Product Database 83
9.5. Updating Individual Records 83
9.6. Expiring Sessions 83

10. DEBUGGING 85

11. SETUP FOR HTTP SERVERS 87
11.1. UNIX−Domain Sockets 87
11.2. INET−Domain Sockets 87
11.3. Internal HTTP Server 88
11.4. Setting Up VLINK and TLINK 88
11.5. Compiling VLINK and TLINK 88
11.6. Manually Compiling VLINK and TLINK 89

12. FREQUENTLY ASKED QUESTIONS 91
12.1. Configuration Problems 91
12.2. VLINK or TLINK Compile Problems 91
12.3. You cannot run the Interchange software as> root. 92
12.4. Error −− the Interchange server was not running 93
12.5. I get messages like 'Config.pm not found.' What does it mean? 94
12.6. Your demo has a very large price for the first item, and when I change the quantity the price drops precip 94
12.7. Can't locate lib.pm in @INC. BEGIN failed−−compilation aborted. 94
12.8. Can I run Interchange on my Macintosh? 94
12.9. Shopping cart is dropped when using SSL. 94
12.10. Segmentation fault or other core dump. 95
12.11. XXXXXX.pm does not match executable version. 95
12.12. Interchange doesn't work with RedHat 5.0 (glibc). 95

13. ISP PROBLEMS 97
13.1. No shell access allowed on my ISP. 97
13.2. No C compiler to compile VLINK or SVEND. 97
13.3. Perl version too low. 97
13.4. We're sorry, the Interchange server is unavailable. 97
13.5. Document contains no data or premature end of script headers (especially on BSDI or FreeBSD). 98
13.6. My provider runs Windows NT. 98
13.7. Interchange server only runs for a while, then dies. 98

14. SETUP QUESTIONS 99
14.1. Can I run multiple catalogs on one server? 99
14.2. How do I start Interchange when I reboot? 99
14.3. How do I set up a Mall? 99
14.4. Can I attach a size or color to a product? 100
14.5. PGP encryption −− Server Error 100

Table of Contents

14.6. PGP encryption −− What do I do now that it is working?....101
14.7. How do I get the number of items in a shopping cart?....102
14.8. The demo doesn't do...(pick one)....102
14.9. Difficulty interacting with browser: Request method for form submission is not POST....102
14.10. How can I trace the source of a purchase and run a partners program?....103
14.11. How can I send an email copy of the receipt to a user?....103
14.12. Configuring catalog whatever...Use of uninitialized value at Config.pm line 1614, <CONFIG> chunk 322.103
14.13. Why isn't the error more enlightening?....104
14.14. We need some more information! Problem: unspecified error....104
14.15. Sorting doesn't work across multiple pages....104
14.16. I am searching for a string and it is not found. I know it is there!....104
14.17. How do I display Euro pricing?....104

15. PERFORMANCE ISSUES....107
15.1. Interchange runs, but it's sooo sllooowww....107
15.2. Interchange slows down over time....107
15.3. The browse.html page (from the demo) is SLOW+!....107
15.4. I am using SQL, and Interchange is so SLOW....108
15.5. My session files change to owner root every day!....108

16. AVAILABILITY AND COMPATIBILITY....109

17. DEMO APPLICATIONS....111

18. KNOWN PROBLEMS....113

1. CONFIGURATION OF AKOPIA INTERCHANGE TM

Akopia Interchange is the industry's most widely distributed and implemented open source e−commerce platform. This document provides detailed configuration instructions for Interchange and the Construct Something demo store demo included with the program, and answers a variety of frequently asked questions. The Interchange TM program must be installed before continuing. For more information on the Installation process, see the Installation (ICINSTALL.SDF) manual.

2. CREATING A CATALOG

Interchange uses unique tags to implement catalog functions. The tags are similar to HTML, but the code is in `[square brackets]`. The Interchange tags will be referred to as either tags or elements in this document.

There are multiple steps required to set up a custom Interchange catalog. Refer to the Reference Tags (ICTAGS.SDF) manual for more information about Interchange tags and how they are used during initial set up a catalog. The demo catalogs provide examples of these tags and how they are used.

2.1. Create a Database

The first step is to develop a product database. This can contain all of the information used to display product pages, or simply the product code (SKU), a short description, and the price. At the minimum, the product code, description, and price fields are required.

The following is a list of other elements that can be included:

Image

A database field listing the name of an image file that displays the product. Specify a name here, or omit this field from the database and store images in files that are named for the product code. If using images, they will be displayed by using the `[if file file.gif] TAG [/if]` construct).

Nontaxable

If the catalog contains any items that are non−taxable, use this field to flag items that do not need to have sales tax calculated.

Size

A comma−separated string containing product size information, for example:

```
Small, Medium, Large, XL
```

Other attribute information may be defined in a database field.

Weight

Used to enter the weight of products being shipped. This is useful for UPS lookup or other weight−based shipping calculations.

Category

Used to select one−click category searches to build product directories.

Related

Elements related to the item in the record. Using the Interchange `[loop item]` tag, product subclasses and accessory sets can be built.

By default all database source files are located in the `products` subdirectory of the catalog directory. The main products database is in the file `products/products.txt` in the supplied demo catalogs.
If using one of the internal database methods, any changes made to the ASCII source file will be reflected in the database in the next user session. If the product database contains less than a thousand records, updates will be instantaneous. If the product database is substantially larger, updates will take longer. Use the `NoImport` reference tag to stop auto updating.

2.2. Features to Use

The Interchange demo catalog shows many features of Interchange, such as list and category building, controlled search or query, random selection of feature items, and templating with active elements. However, each feature has its cost in complexity and performance. Expect reduced performance if the computer has to do more for every page it displays.

The following is a list of some popular features:

Category Searches

If a catalog contains a small number of products, hard coded pages are sufficient, though they can be more difficult to maintain compared to database definitions. Regardless of the number of products in a catalog, consider categorizing them so they can easily be found. Pick a field in the database, perhaps named `category`, and classify the products for search using Interchange.

Images

Place a thumbnail image (perhaps with a link to a blowup) for the items that have images. To do this, add an image field in the database. Requiring file tests for every image in a large list can degrade system performance.

Related Items

On a product's individual display page, embed searches of similar products with the `[query ...]` or `[loop ...]` tags. Or, if customer data is developed, search a past order database and display products that would be of interest to that customer.

Reviews/Testimonials

Key the placement on the existence of a file in a certain directory. This is reasonable to do when a user is viewing a single product.

2.3. Use the Demo Catalogs

The demo catalogs supplied with Interchange are a starting point for the creation of a custom catalog. They can be modified are−named, and new icons can be added. Also, individual pages and the `catalog.cfg` file can be modified. Each demo catalog is independent, so modifications made to one catalog will not effect others.

2.4. Tree design

Determine how users will enter and exit the catalog. Complex and intelligent conditional schemes are possible, especially if the Cookies capability is used. However, it is recommended that simplicity be used. Consumers will not buy if they can't navigate their way around the catalog.

It is important to remember is that users will lose their session (and items in their shopping cart) if their browser does not accept cookies. Interchange addresses this problem by using of the `area` and `page` tags. If using frames, source all frame panes containing Interchange links from an initial page served by Interchange. If not, the user may have multiple session IDs depending on which frame generated the link. Note that Interchange can work properly with browsers that do not support cookies. In this situation Interchange inserts a session ID into each URL; if the ID is preserved as the user navigates from page to page then the session will remain intact.

2.5. The Essentials

The rest of this section describes how to create a basic Interchange catalog, one which displays pages and uses the demo shopping basket and checkout sequence. If custom features are desired, like special shipping charges and sales tax, more work is required. All of the mentioned features (and more) are demonstrated in the `simple` demo catalog.

2.6. The Demo Catalog

Sample catalog pages are in the directory `simple/`. If using these sample pages as the starting point for a custom catalog, either have the configure script install the demo automatically, or copy the files into the Interchange directory and a unique HTML directory.

To install the demo and name it `simple`:

bin/makecat simple

To install the demo and name it `foo`:

bin/makecat foo

Answer the prompts supplied by the program. Note that there are two types of paths requested: URL paths, like /cgi−bin inside `http://www.machine.com/cgi−bin/simple`, and file paths that are complete fully−qualified file path names like `/home/user/catalogs/simple` and `/home/httpd/cgi−bin`. Be prepared to do some investigation to determine the proper paths for installation.

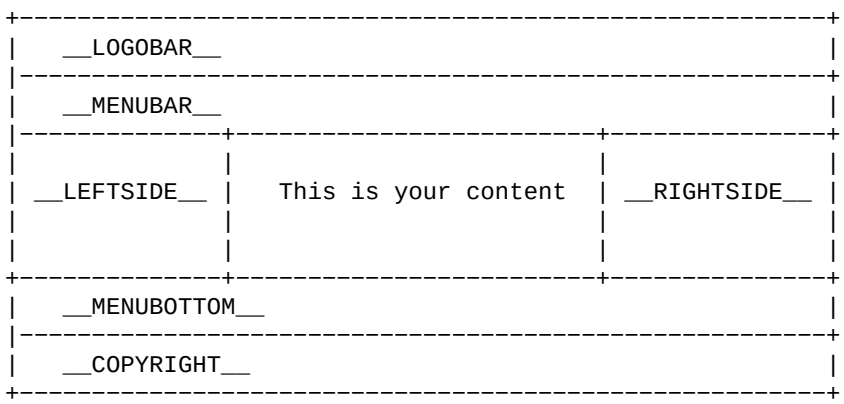
2.7. More About "simple"

The `simple` catalog really isn't. It is a full−featured demonstration of Interchange capabilities, though not nearly all of its features are used there. It uses the `Variable` feature extensively to simplify hand page editing. Basically speaking, a `Variable` is a define that permits the substitution of text for a simple `__VARIABLE__` string in a page.

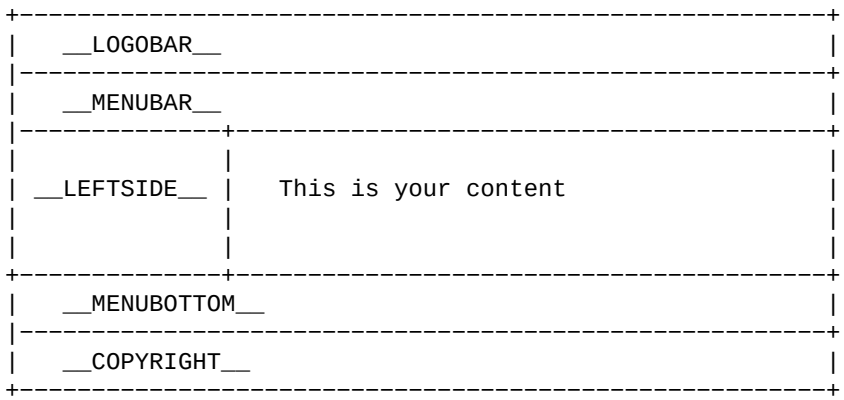
For example, in the simple demo, this is a complete page with the standard layout:

```
__LOGOBAR__
__MENUBAR__
__LEFTSIDE__

    This is unique content.

__RIGHTSIDE__
__MENUBOTTOM__
__COPYRIGHT__
```

The following illustration shows how this looks on the page.



In HTML, the `__RIGHTSIDE__` is simply defined to be empty, closing the table row in most cases. The following illustration is actually how the page looks.



The variables are defined in `catalog.cfg`, and come from files in the config directory. There are conditional templates with different color schemes.

Once familiar with this method, it can be quite convenient. But other methods may be chosen, such as using an HTML editor. Interchange can coexist with editors that use the <NOTOUCH> </NOTOUCH> or <NOEDIT> </NOEDIT> pairs to delimit information that should not be altered or interpreted by the editor. Bear in mind that this templating method and the database layout of the simple catalog is only a suggestion on how a catalog might be organized. Many users decide to create their own database structure for product display.

Yet, most users stick with the userdb and checkout strategies employed in this catalog. This is because these strategies have been developed and proven successful over a period of years. Still, they can be customized to operate like Amazon.com or another site.

As stated previously, Interchange catalogs are all about the database. The `simple` demo has several tables:

area

A table designed to implement a dynamic navigation bar. It is read and parsed to display the areas `Galleries`, `Other Stuff`, and `Links` as distributed. Simply by changing the contents of the `area` table, the left side navigation bar can be modified. The value of the field `selector` is used to search the `cat` table to find which categories belong in each area.

cat

The product structure of `simple` follows a single−level categorization. Categories are listed in the `cat` database table. The `cat` table is designed to allow the display of meta−information about each category. The fields are:
STYLE="font−weight: bold"> **code**

The category code. Must be unique.
STYLE="font−weight: bold"> **area**

Where the area in this category should be displayed.
STYLE="font−weight: bold"> **selector**

The selector that is used to scan the `products` table for products in the category.
STYLE="font−weight: bold"> **name**

A display label to replace `code` or `selector`.
STYLE="font−weight: bold"> **banner_text**

Text to display in a banner or heading.
STYLE="font−weight: bold"> **subs**

Placeholder in the demo that is intended to allow selection of certain subcategories within the category.
STYLE="font−weight: bold"> **sort**

The sort order if an alphanumeric sort on `name` is not desired.
STYLE="font−weight: bold"> **url**

A URL to link to instead of the default search in the products database.

country

A list of countries used to build select boxes and select shipping modes based on countries.

inventory

Two fields: code and quantity. The Interchange demo will decrement the quantity based on the sale. It is recommended that a field be added with the usual shipping time of the product. For example, "ships within 24 hours" or "Back ordered. Allow 2−4 weeks for delivery."

pricing

A database that works in conjunction with the `CommonAdjust` directive to allow quantity pricing, either for one product or for a group of products (sometimes known as mix−and−match). The fields `q2,q5,q10,etc.` are for the quantity levels; the `price_group` field selects the mix−and−match category for the product.

products

The main product table. This is where product information is stored. Add, delete, or ignore fields as desired. Field by field:
STYLE="font−weight: bold"> **sku**

The unique identifier for the product. Though theoretically it can contain most any character, use only characters of the class `A−Z a−z 0−9 _ −` in order to be compatible with SQL databases, file systems, and URL encoding. Other characters can cause problems. A slash (`/`) can interfere with URLs and filenames. A colon (`:`) can interfere with database representations (or file names on some operating systems). There are other possible compatibility situations with other characters.
STYLE="font−weight: bold"> **description**

A short description for the product that is used for displaying in the shopping cart, receipt, and order report. Keep it short.
STYLE="font−weight: bold"> **title**

This is usually catalog−specific since it applies to art. However, it would also apply to books, CDs, or other items.
STYLE="font−weight: bold"> **artist**

This is usually catalog−specific since it applies to art. However, it would also apply to CDs, or some other items. It can be changed as long as the field names in the catalog pages are changed to match it. Simply look for all occurrences of the string "artist" in the `pages`, `special_pages`, and `etc` directories.
STYLE="font−weight: bold"> **comment**

A long description of the product. If using the Interchange internal database, the field size is unlimited; if using another type of database, the length will be dependent on the field type selected.
STYLE="font−weight: bold"> **display**

Catalog−specific to show the museum where the art work is displayed. This field can be used or deleted if not needed.
STYLE="font−weight: bold"> **image**

The image filename for the product. Many users add a `thumb` field to contain the name of a small image for search list display.
STYLE="font−weight: bold"> **price**

The quantity−one price of the product.
STYLE="font−weight: bold"> **category**

The category used for selection in lists. It is possible to place a product in more than one category, although this will require decisions about display and banner headings. Embedded Perl or other methods can be used.
STYLE="font−weight: bold"> **nontaxable**

If set to `1`, the product is not taxable and its price will not be used in a sales tax calculation.
STYLE="font−weight: bold"> **weight**

A numeric value of the weight used for determining shipping costs. Normally in pounds, it can be anything the shipping routines will handle.
STYLE="font−weight: bold"> **size**

A comma−separated list of options as used in `Accessories`. See the `flypage.html` and `ord/basket.html` pages for examples of how it is used.
STYLE="font−weight: bold"> **color**

A comma−separated list of options as used in `Accessories`. See the `flypage.html` and `ord/basket.html` pages for examples of how it is used.
STYLE="font−weight: bold"> **related**

Used in the demo to display "upsells." This refers to opportunities to purchase an additional item when this one is purchased. Contains a comma−separated list of SKUs to be offered.
STYLE="font−weight: bold"> **featured**

As used in this demo, the only meaningful value is `front`. If this is contained in the field, this product may be featured on the front page via random selection. See the `index.html` page for how it integrates with `[loop search=something random=3]`.

orderline

Every line item that is actually ordered is detailed in this table. See the page `query/check_orders.html` for how it might be used. See `etc/report` for how to add to it.

transactions

Each individual customer order is detailed in this table. See the page `query/check_orders.html` for how it might be used. See `etc/report` for how to add to it.

userdb

The user database used for maintaining customer address information, account information, preferences, and more. See `UserDB`.

2.8. IML − Interchange Markup Language

Interchange functions are accessed via IML, the Interchange Markup Language. The pages in the catalog are in HTML, but use IML tags to provide access to Interchange's functions. IML tags use `[square brackets]` instead of angular brackets. They are usually called Interchange tags or just tags. They are similar to HTML in syntax. There are both <I>standalone</I> and <I>container</I> tags. A standalone tag has no ending element. For example:

```
[value name]
```

This tag inserts the user's name, providing it has been provided in a form. A container tag has both a beginning and an ending element. For example:

```
[if value name]
You have a name.
[/if]
```

These tags perform various display and data modification operations for the user session. There are over 80 standard predefined tags; the `UserTag` facility permits the creation of new tags to perform user−defined functions. The user tags can be every bit as powerful as the standard tags.

As an introduction, a few of the tag names and their general function follow:

```
[accessories]     Access product accessory functions
[area]            Insert a re−written Interchange URL
[catch]
  [/catch]
[checked]         Conditionally check an HTML check/radio box
[currency]        Formats a number as currency for current locale
  [/currency]
[data]            Access a database or user session element
[error]           Check/display form processing errors
[file]            Insert the contents of a file
[flag]            Set an Interchange flag
[fly−list]        Show an item "on−the−fly" in an arbitrary page
[ [/fly−list]
[fly−tax]              Builds a tax rate
[html_table]      Create an HTML table from a query or list
[include]         Include a file with complete Interchange interpretation
[item−list]       Iterate over a shopping cart
  [/item−list]
[label]           Set a label for goto
[loop]            Iterate over an arbitrary list
  [/loop]
[mvasp]           ASP−style Perl programming area
[no−match]        Define area of region results displayed when no match
  [/no−match]
[nitems]          Show number of items for a shopping cart
[order]           Create HTML link to order an item
[page]            Create A HREF with re−written URL to call Interchange page
[perl]            Embed output of arbitrary Perl in the page
  [/perl]
[price]           Show price of an item
[process]         Create URL for Interchange form processing
[query]           Perform any of several types of SQL query
  [/query]
[record]          Set a database record
[region]          Define an area of the page as a search list/query
[row]             Used with [col] −− rudimentary text tables for order reports
  [/row]
[salestax]        Show amount of salestax for shopping cart
[scratch]         Access a scratch variable
[search−list]     Display output of an Interchange search
  [/search−list]
[selected]        Conditional selection of drop−down <SELECT ...>
[set]             Set a scratch variable
  [/set]
[sort]            Set sort order for iterating lists
  [/sort]
[subtotal]        Calculate subtotal without tax or shipping
[total−cost]      Calculate order total with tax, handling, and shipping
[try]
  [/try]
[userdb]          Access user database functions
```

```
[value]           Display form value
[value_extended]  Display form array values or do file upload functions
```

A complete list of tags and all tag syntax is available in ICTAGS.POD.

2.9. HTML Hypertext links

Normally, regular hypertext links are not used in Interchange pages. These kinds of links will not include the session ID. If the customer follows an external link back to the catalog, the list of products ordered so far will have been lost. The `area` tag is used to generate a hypertext link which includes a session ID.

Instead of:

```
<A HREF="/cgi-bin/mv/shirts">Shirts</A>
```

Use:

```
<A HREF="[area shirts]">Shirts</A>
```

2.10. Images

Inline images are placed in Interchange pages in the normal fashion with <IMG SRC="URL">. But since Interchange pages are served by a CGI program, do not use relative links. Interchange has the capability of defining an image directory (with the `ImageDir` and `ImageDirSecure` directives) that automatically adjusts the image path to a set base directory.

2.11. Browser Cookies

Enable the `Cookies` directive so that users with cookie−capable browsers will retain session context. Then, standard HREF and Interchange page links can be intermixed without fear of losing the shopping basket. Cookie capability is also required to use search caching, page caching, and statically generated pages. If the user's browser does not support cookies, the cache will be ignored.

If planning to use more than one host name within the same domain for naming purposes (perhaps a secure server and non−secure server), set the domain with the `CookieDomain` directive. This must contain at least two periods (.) as per the cookie specification, and must be located in the same server as the domain.

2.12. Basic Interchange Tags

Note: In the descriptions that follow, parameters marked with an asterisk (*) are optional. When a tag is separated by an underscore, as in `item_list`, a dash is just as appropriate (i.e., `item−list`). They are interchangeable, except that the ending tag and beginning tag should match (don't use `[item−list] list [/item_list]`). `*` *indicates an optional argument*

2.13. Inserting Hyperlinks

Named parameters: [area href="dir/page" secure=1* arg="argument"* form="form string"*]
Positional parameters: [area pg arg*]
Produces the URL to call an Interchange page, without the surrounding `A HREF` notation. This can be used to get control of HREF items, perhaps to place an ALT string or a JavaScript construct. It was originally named `area` because it also can be used in a client−side image map.

```
<A HREF="[area catalog]" ALT="Main catalog page">
```

The optional arg is used just as in the page tag.
The optional `form` argument is used to encode a form in the link.

```
<A HREF="[area form="
        mv_order_item=99−102
        mv_order_size=L
        mv_order_quantity=1
        mv_separate_items=1
        mv_todo=refresh"
]"> Order t−shirt in Large size </A>
```

See above for more information.

[/page], [/order]

Expands into . Used with the page element, such as:

```
[page shirts]Our shirt collection[/page].
```

TIP: A small efficiency boost in large pages is to just use the </A> tag.

[page ...]

Named parameters: `[page href="dir/page" arg="argument" secure=1* form="form string"]`
Positional parameters: `[page dir/page arg*] (only two positional parameters)`
Insert a hyperlink to the specified catalog page pg. For example, [page shirts] will expand into <a href="http://machine.company.com/cgi−bin/vlink/shirts?id=WehUkATn&pc=33">. The catalog page displayed will come from "shirts.html" in the pages directory.
If the user has sent a cookie to Interchange (meaning the second page they access), and the scratch value `mv_no_session_id` is set in their session, the session ID will not be appended to the URL. If the scratch value `mv_no_count` is set, then the page count will not be appended. This is not dependent on cookies. So, if placed in the initial page:

```
[set mv_no_session_id]1[/set]
[set mv_no_count]1[/set]
[set mv_add_dot_html]1[/set]
```

or put in `catalog.cfg`:

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

No session ID or count will be shown. That makes the URL shown above to be http://machine.company.com/cgi−bin/vlink/shirts.html. Once again, this is on the second page the user accesses, if they are taking and sending cookies. If the user has a pre−existing `MV_SESSION_ID` or `MV_USERNAME` cookie from a prior session, the effect will be immediate.
The `argument` will be passed to Interchange and placed in the `mv_arg` session parameter. This allows programming of a conditional page display based on where the link came from. The argument is then available with the tag [data session arg], or the embedded Perl session variable $Session−>{arg}. If the catalog configuration option NewEscape is set, which is the default, then spaces and some other characters will be escaped with the %NN HTTP−style notation and unescaped when the argument is read back into the session.
A bit of magic occurs if Interchange has built a static plain HTML page for the target page. Instead of generating a normal Interchange−parsed page reference, a static page reference will be inserted if the user has accepted and sent back a cookie with the session ID.
The optional `form` argument allows for encoding a form in the link.

```
[page form="
        mv_order_item=99−102
        mv_order_size=L
        mv_order_quantity=1
        mv_separate_items=1
        mv_todo=refresh"] Order t−shirt in Large size </A>
```

The two form values `mv_session_id` and `mv_arg` are automatically added when appropriate. (mv_arg is the `arg` parameter for the tag.)
If the parameter `href` is not supplied, process is used, causing normal Interchange form processing. If the `href` points to an http://link no Interchange URL processing will be done, but the `mv_session_id`. This would generate a form that ordered item number 99−102 on a separate line (`mv_separate_items` being set), with size `L`, in quantity 2. Since the page is not set, the user will go to the default shopping cart page. At the same time, `mv_orderpage=yourpage` could be set to go to `yourpage`. Theoretically, any form can be submitted with this, though none of the included values can have newlines or trailing whitespace. To do so requires the writing of a UserTag. Foreign forms can also be submitted, but they will not touch the href if it begins with http:, ftp:, or the like.

[/page], [/order]

Expands into . Used with the page element, such as:

```
[page shirts]Our shirt collection[/page].
```

\TIP: A small efficiency boost in large pages is to just use the </A> tag.

2.14. How to Order an Item

Interchange can either use a form−based order or a link−based order to place an item in the shopping cart. The link−based order uses the special `[order item−code]` tag:

[order code]

Named parameters:

```
[order code="sku" href="page" cart="cartname"* base="table"]
```

Expands into a hypertext link which will include the specified code in the list of products to order and display the order page. `Code` should be a product SKU listed in one of the "products" tables. The optional argument `cart/page` selects the shopping cart where the item will be placed (begin with / to use the default cart `main`) and the order page that will display the order. The optional argument base constrains the order to a particular products file. If not specified, all tables defined as products files will be searched in sequence for the item.
Example:

```
Order a [order TK112]Toaster[/order] today.
```

Note that this is the same as:

```
Order a [page order TK112]Toaster</A> today.
```

You can change frames for the order with:

```
Order a <A HREF="[area order TK112]" TARGET=newframe>Toaster</A> today.
```

[/order]

Expands into . Used with the order element, such as: Buy a `[order TK112]`Toaster`[/order]` today.

To order with a form, you set the form variable `mv_order_item` to the item−code/SKU and use the `refresh` action:

```
<FORM ACTION="[process−target]" METHOD=POST>
<INPUT TYPE=hidden  NAME="mv_todo"        VALUE="refresh">
<INPUT TYPE=hidden  NAME="mv_order_item"  VALUE="TK112">

Order <INPUT NAME="mv_order_quantity" SIZE=3 VALUE=1> toaster

<INPUT TYPE=submit VALUE="Order!">
</FORM>
```

Groups of items may be batched:

```
<FORM ACTION="[process−target]" METHOD=POST>
<INPUT TYPE=hidden  NAME="mv_todo"        VALUE="refresh">

<INPUT TYPE=hidden  NAME="mv_order_item"  VALUE="TK112">
<INPUT NAME="mv_order_quantity" SIZE=3> Standard Toaster

<INPUT TYPE=hidden  NAME="mv_order_item"  VALUE="TK200">
<INPUT NAME="mv_order_quantity" SIZE=3> Super Toaster

<INPUT TYPE=submit VALUE="Order!">
</FORM>
```

Items that have a quantity of zero (or blank) will be skipped. Only items with a positive quantity will be placed in the basket.

Attributes like size or color may be specified at time of order. Order the T−shirt from the more details page of the `simple` demo to see how it is done.

2.15. Accessing Form Values

When a form is sent to Interchange, the program reads the values and places them in the user session. The following form:

```
<FORM ACTION="[area index]" METHOD=POST>
<INPUT TYPE=hidden NAME=mv_action VALUE=return>
<INPUT TYPE=hidden NAME=foo VALUE=bar>
<INPUT TYPE=submit NAME="Make foo=bar">
</FORM>
```

will make the `[value foo]` tag equal to `bar` on the next page, then, send you to the page `index`(.html). Same for this link:

```
<A HREF="[area  href=index
                    form='
                      foo=bar
                      mv_action=return
                    ']">Make foo=bar</A>
```

[value name=field set="new value" filter="filter" hide=1]

\Positional: `[value varname]`
The `[value ...]` ITL tag expands into the current value of the customer/form input field named by field. If the `set` value is present, the form variable value will be set to it and the new value returned. Use this to "uncheck" a checkbox or set other form variable values to defaults. If `HIDE` is set, the value will be set but not returned to the page.
The `filter="filter1 filter1"` parameter applies any of the standard Interchange filter types to the value. See `Filters`. When the value is returned, any Interchange tags present in the value will be escaped. This prevents users from entering Interchange tags in form values, which could be a serious security risk.

2.16. The Catalog Directory

Interchange pages are NOT in normal HTML space. They are contained in the catalog directory. Each individual catalog must have its own base directory. The catalog directory has the following structure by default:

catalog.cfg

File containing configuration directives for this catalog. (Subcatalogs have differing information in a file named for the subcatalog.)

config

Directory that will be read when directives are set with the `filename` notation. For example, the file `config/static.pages` will be read when the following directive is encountered in the `catalog.cfg` file.

```
StaticPage  <static.pages
```

This directory also contains template information used with the `makecat` program.

error.log

File which contains catalog−specific errors. It is also where any syntax errors in embedded Perl code will be shown.

etc

Directory normally used for tracking files, order profiles, and other configuration and log information.

pages

Directory that contains the pages of the catalog. This can be considered to be the "document root" of the catalog. Pages contained therein are called with the path information after the script name. For example:

```
/cgi−bin/simple/products/gold will call the page in the file
pages/products/gold.html.
```

products

Directory containing database source files, including the special Interchange databases shipping.asc, pricing.asc (and other shipping database files).

session

Directory which contains session files.

tmp

The temporary or scratch directory for various uses like retired ID numbers, search paging files, sort tests, import temporary files, etc. This is the default set by ScratchDir. It can be redefined to be located on another partition.

2.17. Setting Up Multiple Catalogs

Interchange has multiple catalog capability, and therefore breaks the configuration files 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.

The global `interchange.cfg` file is located in the main Interchange directory, and has only a few server−wide configuration parameters. The most important is the Catalog directive, which defines the catalogs will be created at server startup. The Catalog directive is often set up by the `makecat` program, which can be used to configure a catalog.

Here is an example `Catalog` directive:

Catalog simple /home/catalogs/simple /cgi−bin/simple /secure−bin/simple

simple

The catalog identifier, used as the name of the catalog on command lines. In the supplied demo configuration this would be `simple`. The identifier can contain characters from the set [**A−Za−z0−9_**].

/home/catalogs/simple

The directory where the catalog.cfg file may be found, and usually the directory where pages and databases are kept.

/cgi−bin/simple /secure−bin/simple

The script names which, when containing an Interchange link program, will cause that catalog to be called. At least one must be supplied, and the same name may not 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 may call a different catalog.
```

3. INTERNATIONALIZATION

Interchange has a rich set of I18N features to allow conditional message display, differing price formats, different currency definitions, price factoring, sorting, and other settings. The definitions are maintained in the catalog.cfg file through the use of built−in POSIX support and Interchange's `Locale` directive. All settings are independent for each catalog and each user of that catalog. Customers can access the same catalog in an unlimited number of languages and currencies.

3.1. Setting the Locale

The locale could be set to `fr_FR` (French for France) in one of two ways:

[setlocale locale=locale* currency=locale* persist=1*]

This tag is for new−style tags only and will not work for `[old]`.
Immediately sets the locale to `locale`, and will cause it to persist in future user pages if the `persist` is set to a non−zero, non−blank value. If the `currency` attribute is set, the pricing and currency−specific locale keys and Interchange configuration directives are modified to that locale. If there are no arguments, it sets it back to the user's default locale as defined in the scratch variables `mv_locale` and `mv_currency`.
This allows:

```
Dollar Pricing:

[setlocale en_US]
[item−list]
[item−code]: [item−price]<BR>
[/item−list]

Franc Pricing:

[setlocale fr_FR]
[item−list]
[item−code]: [item−price]<BR>
[/item−list]

[comment] Return to the user's default locale [/comment]
[setlocale]
```

This tag is only available in new mode ("NewTags Yes" or `[new]`).

[page process/locale/fr_FR/page/catalog]

This is the same as `[page catalog]`, except when the link is followed it will set the locale to `fr_FR` before displaying the page. This is persistent.

[page process/locale/fr_FR/currency/en_US/page/catalog]

This is the same as `[page catalog]`, except when the link is followed it will set the locale to `fr_FR` and the pricing/number display to the locale `en_US` before displaying the page. This is persistent.

Once the locale is persistently set for a user, it is in effect for the duration of their session.

3.2. Interchange Locale Settings

The `Locale` directive has many possible settings allowing complete internationalization of page sets and currencies. The `Locale` directive is defined in a series of key/value pairs with a key (which contains word characters only) followed by a value. The value should be surrounded by double quotes if it contains whitespace. In this example, the key is `Value setting`.

```
Locale fr_FR "Value setting" "Configuration de valeur"
Locale de_DE "Value setting" Werteinstellung
```

When accessed via the special tag `[L]Value setting[/L]`, the value `Configuration de valeur` will be displayed **only** if the locale is set to `fr_FR`. If the locale is set to `de_DE`, the string `Werteinstellung` will be displayed. If it is neither, the default value of `Value setting` will be displayed.

The `[L]` and `[/L]` must be capitalized. This is done for speed of processing as well as easy differentiation in text.

Another, perhaps better, way to do this is right in the page. The `[LC]` ... `[/LC]` pragma pair permits specification of locale−dependent text.

```
[LC]
          This is the default text.
  [fr_FR] Text for the fr_FR locale. [/fr_FR]
  [de_DE] Text for the de_DE locale. [/de_DE]
[/LC]
```

Or place an entire new page in place of the default one if the locale key is defined. When locale is in force, and a key named `HTMLsuffix` is set to the locale, Interchange first looks for a page with a suffix corresponding to the locale. For example:

<A HREF="[area index]">Catalog home page</A>

If a page index.html exists, it will be the default. If the current locale is `fr_FR`, a page "index.fr_FR" exists, and `Locale` looks like this:

```
Locale fr_FR HTMLsuffix  fr_FR
```

Then, the `.fr_FR` page will be used instead of the `.html` page. For longer series of strings, the configuration file recognizes:

```
Locale fr_FR <<EOF
{
    "Value setting",
    "Configuration de valeur",

    "Search",
    "Recherche"
}
EOF
```

The above sets two string substitutions. As long as this is valid Perl syntax describing a series of settings, the text will be matched. It can contain any arbitrary set of characters that don't contain `[L]` and `[/L]`. If using

double quotes, string literals like \n and \t are recognized.

A database can also be used to set locale information. It can be added to any database in the `catalog.cfg` file, and values in it will overwrite previous settings. See *LocaleDatabase*. The `[L]default text[/L]` is set before any other page processing. It is equivalent to the characters "default text" or the appropriate `Locale` translation for all intents and purposes. Interchange tags and Variable values can be embedded.

Since the `[L] message [/L]` substitution is done before any tag processing, the command `[L][item-data table field][/L]` will fail. There is an add-on `[loc] message [/loc]` *UserTag* supplied with the distribution beginning at V3.09. It does the same thing as `[L] [/L]` except after all tag substitution is done. See <interchange.cfg.dist> for the definition.

Care is required when editing pages with localization information. Changing even one character of the message will change the key value and invalidate the message for other languages. To prevent this, use:

```
[L key]The default.[/L]
```

The key `msg_key` will then be used to index the message. This may be preferable for many applications.

A `localize` script is included with Interchange. It will parse files included on the command line and produce output that can be easily edited to produce localized information. Given an existing file, it will merge new information where appropriate.

3.3. Special Locale Keys for Price Representation

Interchange honors the standard POSIX keys:

```
mon_decimal_point decimal_point
mon_thousands_sep thousands_sep
currency_symbol   int_currency_symbol
frac_digits       p_cs_precedes
```

See the POSIX setlocale(3) man page for more information. These will be used for formatting prices and will approximate the number format used in most countries. To set the price format to a custom format, use these special keys:

price_picture

Interchange will format a currency number based on a "picture" given to it. The basic form is:

```
Locale en_US price_picture "$ ###,###,###.##"
```

This is locale `en_US`, the United States, and it would display `4452.3` as `$ 4,452.30`. The same display can be achieved with:

```
Locale en_US mon_thousands_sep ,
Locale en_US mon_decimal_point .
Locale en_US p_cs_precedes     1
Locale en_US currency_symbol   $
```

A common price_picture for European countries would be `###.###.###,##`, which would display the

same number as `4.452,30`. To add a franc notation at the end for the locale `fr_FR`, use the setting:

```
Locale fr_FR price_picture "##.###,## fr"
```

IMPORTANT NOTE: The decimal point in use, set by `mon_decimal_point`, and the thousands separator, set by `mon_thousands_sep` must match the settings in the picture. The `frac_digits` setting is not used in this case. It is derived from the location of the decimal (if any).

The same setting for `fr_FR` above can be achieved with:

```
Locale fr_FR mon_thousands_sep .
Locale fr_FR mon_decimal_point ,
Locale fr_FR p_cs_precedes     0
Locale fr_FR currency_symbol   fr
```

If the number of digits is greater than the # locations in the picture, the digits will be changed to asterisks. An overflow number above would show as `**.***,** fr`.

picture

Same as `price_picture`, but sets the value returned if the `[currency]` tag is not used. If the number of digits is greater than the # locations in the picture, the digits will be changed to asterisks, displaying something like `**,***.**`.

3.4. Dynamic Locale Directive Changes

If Locale keys are set to correspond to some Interchange `catalog.cfg` directives, those values will be set when setting the locale.

PageDir

To use a different page directory for different locales, set the `PageDir` key. To have two separate language page sets, French and English, set:

```
# Establish the default at startup
PageDir   english
Locale fr_FR  PageDir  francais
Locale en_US  PageDir  english
```

ImageDir

To use a different image directory for different locales, set the `ImageDir` key. To have two separate language button sets, French and English, set:

```
# Establish the default at startup
ImageDir   /images/english/
Locale fr_FR  ImageDir   /images/francais/
Locale en_US  ImageDir   /images/english/
```

ImageDirSecure

Same as ImageDir.

PriceField

To use a different field in the products database for pricing based on locale, set the `PriceField` locale setting.

```
# Establish the default at startup
PriceField    price
Locale fr_FR  PriceField  prix
```

The default will always be `price`, but if the locale `fr_FR` is set, the `PriceField` directive will change to `prix` to give prices in francs instead of dollars.
If `PriceBreaks` is enabled, the field `prix` from the `pricing` database will be used to develop the quantity pricing. NOTE: If no `Locale` settings are present, it will always be `price`, regardless of the `PriceField` setting. Otherwise, it will always match `PriceField`.

PriceDivide

Normally used to enable penny pricing with a setting of 100, this can be used to do an automatic conversion factor based on locale.

```
# Default at startup is 1 if not set
# Franc is strong these days!
Locale fr_FR  PriceDivide  .20
```

The price will now be divided by `.20`, yielding franc prices five times that of the dollar.

PriceCommas

This controls whether the `mon_thousands_sep` will be used for standard currency formatting. Ignored if using `price_picture`. Set to 1 or 0, to enable and disable respectively. DO NOT USE Yes and No.

```
# Default at startup is Yes if not set
PriceCommas  Yes
Locale fr_FR  PriceCommas  0
Locale en_US  PriceCommas  1
```

UseModifier

Changes the fields from the set shopping cart options.

```
# Default at startup is 1 if not set
# Franc is strong these days!
UseModifier format
Locale fr_FR  UseModifier formats
```

If a previous setting was made for an item based on another locale, it will be maintained.

PriceAdjustment

Changes the fields from `UseModifier` that will be used to adjust pricing to an automatic conversion factor based on locale.

```
# Default at startup
PriceAdjustment  format
Locale fr_FR  PriceAdjustment  formats
```

TaxShipping,SalesTax

Same as the standard directives.

DescriptionField

This changes the field accessed by default with the `[item-description]` and `[description code]` tags.

```
# Establish the default at startup
DescriptionField    description
Locale fr_FR  DescriptionField desc_fr
```

The [locale] tag

Standard error messages can be set based on Locale settings. Make sure not to use any of the predefined keys. It is safer to begin a key with msg_ or such. The default message is set between the `[locale key]` and `[/locale]` tags. See the example above.

3.5. Sorting Based on Locale

The Interchange `[sort database:field]` keys will use the LC_COLLATE setting for a locale provided that:

- The operating system and C compiler support locales for POSIX, and have the locale definitions set.
- The locale setting matches any configured locales.

It is beyond the scope of this document to discuss these issues. Contact your system administrator or local wizard for help setting up locales on your system. NOTE: Windows locales are not supported for sorting.

If this arbitrary database named `letters`:

```
code        letter
00-0011     f
99-102      é
19-202      a
```

and this loop:

```
[loop 19-202 00-0011 99-102]
[sort letters:letter]
[loop-data letters letter]   [loop-code]
[/loop]
```

used the default C setting for LC_COLLATE, it would display:

```
a  19-202
f  00-0011
é  99-102
```

If the proper LC_COLLATE settings for locale `fr_FR` were in effect, then it would become:

```
a  19-202
é  99-102
f  00-0011
```

3.6. Placing Locale Information in a Database

Interchange has the capability to read its locale information from a database, named with the `LocaleDatabase` directive. The database can be of any valid Interchange type. The locales are in columns, and the keys are in rows. For example, to set up price information:

```
key                 en_US   fr_FR   de_DE
PriceDivide         1       .1590   .58
mon_decimal_point   .       ,       ,
mon_thousands_sep   ,       .
currency_symbol     $        frs    DM
ps_cs_precedes      1       0       0
```

This would translate *exactly* into:

```
Locale en_US PriceDivide         1
Locale en_US mon_decimal_point   .
Locale en_US mon_thousands_sep   ,
Locale en_US currency_symbol     $
Locale en_US ps_cs_precedes      1

Locale fr_FR PriceDivide         .1590
Locale fr_FR mon_decimal_point   ,
Locale fr_FR mon_thousands_sep   .
Locale fr_FR currency_symbol     " frs"
Locale fr_FR ps_cs_precedes      0

Locale de_DE PriceDivide         .58
Locale de_DE mon_decimal_point   ,
Locale de_DE mon_thousands_sep   " "
Locale de_DE currency_symbol     "DM "
Locale de_DE ps_cs_precedes      1
```

These settings append and overwrite any that are set in the catalog configuration files, including any #include files. IMPORTANT NOTE: This information is only read during catalog configuration. It is not reasonable to access a database for translation or currency conversion in the normal course of events.

4. PROGRAMMING

Interchange has a powerful paradigm for extending and enhancing its functionality. It uses two mechanisms, user−defined tags and user subroutines on two different security levels, global and catalog. In addition, `Embedded Perl Code` may be used to build functionality into pages.

User−defined tags are defined with the UserTag directive in either `interchange.cfg` or `catalog.cfg`. The tags in `interchange.cfg` are global, i.e., they are not constrained by the `Safe` Perl module as to which opcodes and routines they may use. Normally, the user−defined tags in `catalog.cfg` are constrained by `Safe`, but if the `AllowGlobal` global directive is set for the particular catalog in use, its UserTag and `Sub` definitions will have global capability.

Many of the internal Interchange routines can be accessed by the savvy programmer who reads the source and finds the entry points. Also, many internal Interchange routines can be overridden:

```
GlobalSub <<EOS
sub just_for_overriding {
    package Vend::Module;
    use MyModule;
    sub to_override {
        &MyModule::do_something_funky($Values->{my_variable});
    }
}
EOS
```

The effect of the above is to override the `to_override` routine in the module `Vend::Module`. This is preferable to hacking the code for functionality changes that are not expected to change frequently. In most cases, updating the Interchange code will not affect the hacked code. Note that internal entry points are not guaranteed to exist in future versions of Interchange.

4.1. Embedded Perl Code

Perl code can be directly embedded in Interchange pages. The code is specified as:

```
[perl]
    $name    = $Values->{name};
    $browser = $Session->{browser};
    return "Hi, $name! How do you like your $browser?";
[/perl]
```

ASP syntax can be used with:

```
[mvasp]
    <%
    $name    = $Values->{name};
    $browser = $Session->{browser};
    %>
    Hi, <%= $name %>!
    <%
        HTML "How do you like your $browser?";
    %>
[/mvasp]
```

The two snippets above are essentially equivalent.

The [perl] tag enforces Safe.pm checking, and many standard Perl operators are not available. This prevents user access to all files and programs on the system without the Interchange daemon's permissions. See GlobalSub and UserTag for ways to make external files and programs available to Interchange.

Named parameters:

```
[perl tables="tables-to-open"* subs=1 global=1* no_return=1*]
```

Required parameters: none
Parameter information:
STYLE="font-weight: bold"> **global**

When set to 1, and when the catalog is allowed to use global permissions via the interchange.cfg directive AllowGlobal, Safe.pm checking is turned off. In this case, the embedded Perl code can do anything that the user ID running it can! This is not recommended when a single Interchange server is shared by multiple companies/user ids.
When in global mode, full use strict checking is on. Optionally, it can be turned off with no strict;. In fact, this is recommended since strict errors will cause silent failure.
STYLE="font-weight: bold"> **subs**

When set to 1, any GlobalSub routines can be accessed by name.
STYLE="font-weight: bold"> **tables**

Opens specified Interchange database tables, preparing them for access inside the Perl code.
When running under Safe, some database operations are restricted due to the inability to create objects in this mode. For SQL, most operations can be performed if the Safe::Hole module is installed. Otherwise, use global to use data from SQL tables. Interchange databases can be accessed as long as they are pre-opened by using an item first.
Example:

```
[perl tables="products pricing"]
    $key = $Values->{sku};
    $title = $Tag->data('products', 'title', $key):
    return "You looked up the title $title";
[/perl]
```

If the tables=products> was not specified, there would be a syntax error trap from Safe.pm.

Any Interchange tag (except ones using SQL) can be accessed via the $Tag object. If using SQL queries inside a Perl element, AllowGlobal permissions are required and and the global=1 parameter must be set. Installing the module Safe::Hole along with sharing the database table with <tables=tablename> will enable SQL use.

Examples:

```
# If the item might contain a single quote
[perl]
$comments = $Values->{comments};
[/perl]
```

IMPORTANT NOTE: Global subroutines are not subject to the stringent security checking of the Safe module, so almost anything goes there. The subroutine will be able to modify any variable in

Interchange, and will be able to write to read and write any file that the Interchange daemon has permission to write. Though this gives greater power, it should be used with caution. They are defined in the main `interchange.cfg` file, so should be safe from individual users in a multi−catalog system.

Global subroutines are defined in `interchange.cfg` with the `GlobalSub` directive, or in user catalogs which have been enabled via `AllowGlobal`. Catalog subroutines are defined in `catalog.cfg`, with the `Sub` directive. They are subject to the stringent Safe.pm security restrictions that are controlled by the global directive `SafeUntrap`.

The code can be as complex as desired, but cannot use any operators that modify the file system or use "unsafe" operations like "system," "exec," or backticks. These constraints are enforced with the default permissions of the standard Perl module Safe. Operations may be untrapped on a system−wide basis with the `SafeUntrap` directive.

The result of this tag will be the result of the last expression evaluated, just as in a subroutine. If there is a syntax error or other problem with the code, there will be no output.

Here is a simple one which does the equivalent of the classic hello.pl program:

```
[perl] my $tmp = "Hello, world!"; $tmp; [/perl]
```

Of course, there is no need to set the variable. It is there just to show the capability.

To echo the user's browser, but within some HTML tags:

```
[perl]
my $html = '<H5>';
$html .= $Session−>{browser};
$html .= '</H5>';
$html;
[/perl]
```

To show the user their name and the current time:

```
[perl arg=values]

my $string = "Hi, " . $Values−>{name} ". The time is now ";
$string .= $Tag−>time();
$string;

[/perl]
```

4.2. ASP−Like Perl

Interchange supports an ASP−like syntax using the `mvasp` tag.

```
[mvasp]
<HTML><BODY>
    This is HTML.<BR>

<% HTML "This is code<BR>"; %>
    More HTML.<BR>
<% $Document−>write("Code again.<BR>") %>
```

```
[/mvasp]
```

If no closing `[/mvasp]` tag is present, the remainder of the page will be seen as ASP.

ASP is simple. Anything between <% and %> is code, and the string %> is not allowed anywhere inside. Anything not between those anchors is plain HTML that will be placed unchanged on the page. Interchange variables and [L][/L] and [LC][/LC] areas will still be inserted, but any Interchange tags will not.

There is a shorthand <% = $foo %>, which is exactly equivalent to <% $Document−>write($foo); %> or <% HTML $foo; %>

```
[mvasp]
<HTML><BODY>
    This is HTML.<BR>
    [value name] will show up as &#91;value name].<BR>

    &#95_VARIABLE__ value is equal to: __VARIABLE__

<% = "This is code<BR>" %>
```

The `__VARIABLE__` will be replaced by the value of `Variable VARIABLE`, but [value name] will be shown unchanged.

IMPORTANT NOTE: If using the `SQL::Statement` module, the catalog must be set to `AllowGlobal` in `interchange.cfg`. It will not work in "Safe" mode due to limitations of object creation in Safe. Also, the `Safe::Hole` module must be installed to have SQL databases work in Safe mode.

5. INTERCHANGE PERL OBJECTS

Interchange gives ready access to all objects associated with the catalog and the user settings, with opcode restrictions based on the standard Perl module `Safe.pm`. There are some unique things to know about programming with Interchange.

Under `Safe`, certain things cannot be used. For instance, this is not legal when running Safe:

```
$variable = `cat file/contents`;
```

The backtick operator violates a number of the default Safe opcode restrictions. Direct file opens are not allowed, either:

```
open(SOMETHING, "something.txt")
    or die;
```

This will also cause a trap, and the code will fail to compile. However, Interchange equivalent routines can be used:

```
# This will work if your administrator doesn't have NoAbsolute
# set
$users = $Tag->file('/home/you/list');

# This will always work, file names are based in the
# catalog directory
$users = $Tag->file('userlist');
```

The standard objects are:

$CGI

This is a hash reference to `%CGI::values`, the value of user variables as submitted in the current page/form. To get the value of a variable submitted as

```
<INPUT TYPE=hidden NAME=foo VALUE=bar>
```

use

```
<% $Document->write("Value of foo is $CGI->{foo}"); %>
```

Remember, multiple settings of the same variable are separated by a NULL character. To get the array value, use $CGI_array.

$CGI_array

This is a hash reference to `%CGI::values_array`, arrays containing the value or values of user variables as submitted in the current page/form. To get the value of a variable submitted as

```
<INPUT TYPE=hidden NAME=foo VALUE=bar>
<INPUT TYPE=hidden NAME=foo VALUE=baz>
```

use

```
<% = "The values of foo are", join (' and ', @{$CGI_array->{'foo'}}) %>
```

Remember, multiple settings of the same variable are separated by a NULL character. To get the array value, use $CGI_array.

$Carts

A reference to the shopping cart hash $Vend::Session->{carts}. The normal default cart is "main". An alias is $Items in the normal course of events.
Shopping carts are an array of hash references. Here is an example of a session cart array containing a `main` and a `layaway` cart.

```
{
    'main' => [
                {
                    'code' => '00-0011',
                    'mv_ib' => 'products',
                    'quantity' => 1,
                    'size' => undef,
                    'color' => undef
                },
                {
                    'code' => '99-102',
                    'mv_ib' => 'products',
                    'quantity' => 2,
                    'size' => 'L',
                    'color' => 'BLUE'
                }
            ],
    'layaway' => [
                {
                    'code' => '00-341',
                    'mv_ib' => 'products',
                    'quantity' => 1,
                    'size' => undef,
                    'color' => undef
                }
            ]
}
```

In this cart array, $Carts->{main}[1]{code} is equal to `99-102`. In the normal course of events, it would be equivalent to $Items->[1]{code}.

$Config

A reference to the $Vend::Cfg array. Normally use this with a liberal dose of the Interchange source code, but for simple things use something like:

```
# Turn off CyberCash for this user
$Config->{CyberCash} = 0;
```

Changes are not persistent except when running in the foreground (Debug mode or on Windows).

%Db

A hash of databases shared with the `[mvasp tables="foo"]` parameter to the tag call. Once the

database is shared, it is opened and can be accessed by any of its methods. Again, this will not work with SQL unless AllowGlobal is set for the catalog.
To get a reference to a particular table, specify its hash element:

```
$ref = $Db{products};
```

The methods available:

```
# access an element of the table
$field = $ref->field($key, $column);

# set an element of the table
$ref->set_field($key, $column_name, $value);

# atomic increment of an element of the table
$ref->inc_field($key, $column_name, 1);

# see if element of the table is numeric
$is_numeric = $ref->numeric($column_name);

# Quote for SQL query purposes
$quoted = $ref->quote($value, $column_name);

# Check configuration of the database
$delimiter = $ref->config('DELIMITER');

# Find the names of the columns (not including the key)
@columns = $ref->columns();
# Insert the key column name
unshift @columns, $ref->config('KEY');

# See if a column is in the table
$is_a_column = defined $ref->test_column($column_name);

# See if a row is in the table
$is_present = $ref->record_exists($key);

# Create a subroutine to return a single column from the table
$sub = $ref->field_accessor($column);
for (@keys) {
    push @values, $sub->($key);
}

# Create a subroutine to set a single column in the database
$sub = $ref->field_settor($column);
for (@keys) {
    $sub->($key, $value);
}

# Create a subroutine to set a slice of the database
$sub = $ref->row_settor(@columns);
for (@keys) {
    $sub->($key, @values);
}

# Retrurn a complete array of the database (minus the key)
@values = $ref->row($key);

# Retrurn a complete hash of the database row (minus the key)
$hashref = $ref->row_hash($key);
```

```
# Delete a record/row from the table
$ref->delete_record($key);
```

%Sql

A hash of SQL databases that you shared with the `[mvasp tables="foo"]` parameter to the tag call. It returns the DBI database handle, so things like the following can be done:

```
<%
  my $dbh = $Sql{products}
      or return HTML "Database not shared.";
  my $sth = $dbh->prepare('select * from products')
      or return HTML "Couldn't open database.";
  $sth->execute();
  my @record;
  while(@record = $sth->fetchrow()) {
      foo();
  }
  $sth = $dbh->prepare('select * from othertable')
      or return HTML "Couldn't open database.";
  $sth->execute();
  while(@record = $sth->fetchrow()) {
      bar();
  }
%>
```

Again, this will not work with unless AllowGlobal is set for your catalog.

$DbSearch

A search object which will search a database without using the text file. It is the same as Interchange's `db` searchtype. Options are specified in a hash and passed to the object. All multiple-field options should be passed as array references. Before using the $DbSearch object, it must be told which table to search. For example, to use the table `foo`, it must have been shared with `[mvasp foo]`.
There are three search methods: `array`, `hash`, and `list`.

```
array    Returns a reference to an array of arrays (best)
hash     Returns a reference to an array of hashes (slower)
list     Returns a reference to an array of tab-delimited lines
```

\Example:

```
$DbSearch->{table} = $Db{foo};

$search = {

        mv_searchspec => 'Mona Lisa',
        mv_search_field => [ 'title', 'artist', 'price' ],
        mv_return_fields    => [ 'title' ]

    };

my $ary = $DbSearch->array($search);

if(! scalar @$ary) {
    return HTML "No match.\n";
}
```

```
for(@$ary) {
```

$Document

This is an object that has several routines associated with it.

```
HTML $foo;                      # Append $foo to the write buffer array
$Document->write($foo);         # object call to append $foo to the write
                                # buffer array
$Document->insert($foo);        # Insert $foo to front of write buffer array
$Document->header($foo, $opt); # Append $foo to page header
$Document->send();              # Send write buffer array to output, done
                                # automatically upon end of ASP, clears buffer
                                # and invalidates $Document->header()
$Document->hot(1);              # Cause writes to send immediately
$Document->hot(0);              # Stop immediate send
@ary = $Document->review();     # Place contents of write buffer in @ary
$Document->replace(@ary)        # Replace contents of write buffer with @ary
$ary_ref = $Document->ref();    # Return ref to output buffer
```

$Document->write($foo)

Write $foo to the page in a buffered fashion. The buffer is an array containing the results of all previous `$Document->write()` operations. If $Document->hot(1) has been set, the output immediately goes to the user.

$Document->insert($foo)

Insert $foo to the page buffer.

```
$Document->write("23");
$Document->insert("1");
$Document->send();
```

will output "123", while

```
$Document->write("23");
$Document->write("1");
$Document->send();
```

will output "231".

$Document->header($foo, $opt)

Add the header line $foo to the HTTP header. Use this to change the page content type, cache options, or other attributes. The code below changes the content type (MIME type) to text/plain:

```
$Document->header("Content-type: text/plain");
```

There is an optional hash that can be sent, with the only valid value being "replace." The below code scrubs all previous header lines:

```
$Document->header("Content-type: text/plain", { replace => 1 } );
```

Once output has been sent with $Document->send(), this can no longer be done.

$Document−>hot($foo)

If the value of $foo is true (in a Perl sense), then all $Document−>write() operations will be immediately sent until a $Document−>hot(0) is executed.

$Document−>send()

Causes the document write buffer to be sent to the browser and empties the buffer. Any further $Document−>header() calls will be ignored. Can be used to implement non−parsed−header operation.

$Document−>review()

Returns the value of the write buffer.

```
@ary = $Document->review();
```

$Document−>replace(@new)

Completely replaces the write buffer with the arguments.

$Document−>ref()

Returns a reference to the write buffer.

```
# Remove the first item in the write buffer
my $ary_ref = $Document->ref();
shift @$ary_ref;
```

HTML

Writes a string (or list of strings) to the write buffer array. The call

```
HTML $foo, $bar;
```

is exactly equivalent to

```
$Document->write($foo, $bar);
```

Honors the $Document−>hot() setting.

$Items

A reference to the current shopping cart. Unless an Interchange `[cart ...]` tag is used, it is normally the same as $Carts−>{main}.

$Scratch

A reference to the scratch values ala `[scratch foo]`.

```
<% $Scratch->{foo} = 'bar'; %>
```

is equivalent to:

```
[set foo]bar[/set]
```

$Session

A reference to the session values ala `[data session username]`.

```
<%
    my $out = $Session->{browser};
    $Document->write($out);
%>
```

is equivalent to:

```
[data session browser]
```

Values can also be set. If the value of `[data session source]` needed to be changed, for example, set:

```
<%
    $Session->{source} = 'New_partner';
%>
```

$Tag

Using the $Tag object, any Interchange tag including user-defined tags can be accessed.

IMPORTANT NOTE: If the tag will access a database that has not been previously opened, the table name must be passed in the ASP call. For example:

HTML style:

```
<HTML MV="mvasp" MV.TABLES="products pricing">
```

or
Named parameters:

```
[mvasp tables="products pricing"]
```

or
Positional parameters:

```
[mvasp products pricing]
```

Any tag can be called.

```
<%
    my $user = $Session->{username};
    my $name_from_db = $Tag->data('userdb', 'name', $user );
    $Document->write($name_from_db);
%>
```

is the same as:

```
[data table=userdb column=name key="[data session username]"]
```

If the tag has a dash (−) in it, use an underscore instead:

```
# WRONG!!!
$Tag->shipping-desc('upsg');
# Right
$Tag->shipping_desc('upsg');
```

There are two ways of specifying parameters. Either use the positional parameters as documented (for an authoritative look at the parameters, inspect the %Routine value in Vend::Parse), or specify it all with an option hash parameter names as in any named parameters as specified in an Interchange tag. The calls

```
$Tag->data('products', 'title', '00-0011');
```

and

```
my $opt = {
                table   => 'products',
                column  => 'title',
                key     => '00-0011',
            };

$Tag->data( $opt );
```

are equivalent for the `data` tag.
If using the option hash method, and the tag has container text, either specify it in the hash parameter `body` or add it as the next argument. The two calls:

```
$Tag->item_list( {
                    'body' => "[item-code] [item-field title]",
                });
```

and

```
$Tag->item_list( { }, "[item-code] [item-field title]")
```

are equivalent.
Parameter names are ALWAYS lower case.

$Values

A reference to the user form values ala `[value foo]`.

```
<% $Document->write($Values->{foo}); %>
```

is equivalent to:

```
[value foo]
```

&Log

Send a message to the error log (same as ::logError in GlobalSub or global UserTag).

```
<%
    Log("error log entry");
%>
```

It prepends the normal timestamp with user and page information. To supress that information, begin the message with a backslash (\).

```
<%
    Log("\\error log entry without timestamp");
    Log('\another error log entry without timestamp');
    Log("error log entry with timestamp");
%>
```

6. INTERCHANGE CONFIGURATION FILES

Interchange can and usually does run multiple catalogs on the same server.

Cconfiguration directives are normally called with the directive as the first word on the line, with it's value or values following. Leading whitespace is stripped from the value. Additional files may be called with an #include file statement. Files are relative to the catalog directory (or Interchange software directory, if in the main `interchange.cfg` file).

A type of "here document" can be used to specify Interchange directives, with the usual Perl <<MARKER syntax. No semicolon is used to terminate the marker.

7. 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.

7.1. ActionMap *global*

Specifies an action for use as a default. If the same action is specified in `catalog.cfg`, it will pertain. See `ActionMap` in that section.

7.2. 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/`.

7.3. 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
```

7.4. 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
```

7.5. Catalog *global*

Specifies a catalog that can run using this Interchange server. There are three required parameters, separated by spaces and/or tabs.

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.

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 directive is very important to get right. It is the `SCRIPT_NAME` of the vlink program that runs the catalog. This name must be unique from other CGI program paths that run on this server because this is how the catalog is selected for operation.

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

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

Optionally, individual `Catalog` lines that specify the different parameters may be called. The equivalent of the 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
```

7.6. Catch

7.7. 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
```

7.8. 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
```

7.9. 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
```

7.10. Counter

Inserts the value of a persistent counter. The counter value is stored in the given file, which is created if necessary. If the file name begins with a "/," then it is an absolute path. Otherwise, it is relative to VendRoot. If the tag causes the counter file to be created, the value is initialized to start. If value is given, the counter's value is set to the given value. If value is not supplied, the counter is incremented and its value is returned, unless decrement is supplied. In this case, the counter is decremented and its value is returned. The counter is implemented using Perl's `File::Counter` module, which protects the file against simultaneous access by multiple processes.

```
[counter file=FILE start=START value=VALUE decrement*]
```

7.11. Discount−Subtotal

Inserts the discounted subtotal of the ordered items.

7.12. 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.

7.13. 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.

7.14. 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.

7.15. 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
```

7.16. 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`.

7.17. Export

7.18. Filter

Applies any of Interchange's standard filters to an arbitray value, or custom filters may be defined. The filters are also available as parameters to the `cgi`, `data`, and `value` tags. Filters can be applied in sequence and as many as needed can be applied. Here is an example. If author or artist names are stored in the database ``LAST, First'' so that they sort properly, they may be displayed normally as ``First Last''. This call

```
[filter op="name namecase"]WOOD, Grant[/filter] will display as
Grant Wood Another way to do this would be:
[data table=products column=artist key=99−102 filter="name namecase"]
```

7.19. Fly−Tax

Builds a tax rate from `taxarea`, `taxrate`, `taxshipping`, variable values, and the `SalesTax` directive value.

7.20. 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.

7.21. 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.

7.22. 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.

7.23. 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
```

7.24. Handling

Calculates and inserts handling costs. Accepts the same noformat and convert arguments as the shipping tag.

7.25. 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
```

7.26. 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
```

7.27. IpQuad *global*

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

```
IpQuad  2
```

7.28. Legacy *global*

Makes a few optimizations for newer catalogs. Set to `Yes` for the maximum compatibility with older Interchange catalogs.

7.29. Locale *global*

Sets the global `Locale` for use in error messages.

7.30. 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
```

7.31. Log

7.32. 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.

7.33. 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.

7.34. 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
```

7.35. 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.

7.36. 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.

7.37. SafeUntrap *global*

Sets the codes that will be untrapped in the `Safe.pm` module and used for embedded Perl and conditional operations. The Safe.pm documentation is visible by typing `perldoc Safe` at the command prompt. The default is `249 148` for Perl 5.003, and `ftfile sort` for Perl 5.003_20 and above, 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
```

7.38. 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.

7.39. 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
```

7.40. 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
```

7.41. 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
```

7.42. 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
```

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)
```

A custom wrapper can be created around it to get it to behave as desired.

7.43. Tag

7.44. 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
```

7.45. 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.

7.46. TemplateDir *global*

Sets a directory which will be searched for pages if not found in the users `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.

7.47. Time

7.48. Try

7.49. 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.

7.50. Variable *global*

Defines a global variable that will be available in all catalogs 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). 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
```

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

The current locale for currency.

LANG

The current locale for language.

7.51. 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.

8. CATALOG.CFG

If multiple catalogs are to be run, each must have a `catalog.cfg` file located in the catalog base directory. It contains most of the configurable parameters for Interchange. Each is independent from catalog to catalog.

In the catalog configuration file, the directives `MailOrderTo` and `VendURL` are required. They are not defined with defaults, and no catalog will operate unless and until they are set with a value.

Many powerful procedures 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
```

Values can be defined in multiline form by using the <<HERE document method:

```
Variable MYSTUFF <<EOF
    This is my stuff, and I
    am pretty proud of it.
EOF
```

As with all here documents, the terminating string must be the ONLY THING ON THE LINE. No leading or trailing characters are allowed, even whitespace.

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 multiple settings from file

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
```

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.

8.1. 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. See `Interchange Programming` for details.

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.

8.2. 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.

8.3. 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, IML tags may be used in an action, though they are not nearly as efficient.

8.4. 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
```

8.5. 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.

8.6. 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
```

8.7. 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.

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]
```

8.8. CommonAdjust

Settings for Interchange pricing. See `Chained pricing`.

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

8.9. 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`.

8.10. 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.

8.11. 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`.

8.12. 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.

8.13. 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
```

This should be turned off if using `CyberCash`.

8.14. 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
```

8.15. CyberCash

A Yes/No directive, default is No. Enables CyberCash payment protocols. See Using CyberCash.

```
CyberCash           Yes
```

8.16. 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, and is put in DataDir if no path is provided. 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
```

8.17. 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
```

8.18. 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.

8.19. 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
```

8.20. 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.

8.21. 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.

8.22. 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
```

8.23. 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
```

8.24. 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.

8.25. 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
```

8.26. 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
```

8.27. 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
```

8.28. HTMLsuffix

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

```
HTMLsuffix .htm
```

8.29. 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/
```

8.30. 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).

8.31. 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/
```

8.32. 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.

8.33. 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.

8.34. 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
```

8.35. 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.

8.36. 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
```

8.37. 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
```

8.38. 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
```

8.39. 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
```

8.40. 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.

8.41. 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.

8.42. 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
```

8.43. 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.

8.44. OrderReport

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

```
OrderReport          /data/order-form
```

8.45. 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.

8.46. 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
```

8.47. ParseVariables

Determines whether global and catalog variables will be parsed in the configuration file. Should be set to No until parsing is needed, turned on for the parsed directives, then, set back to No. Default is No.

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

8.48. 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.
```

8.49. 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.

8.50. Pragma

Sets the default value of an Interchange pragma. The only one to date is:

no_html_parse

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

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

```
[pragma no_html_parse]
```

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

```
[pragma no_html_parse 0]
```

Note: This is a good place to place a DTD.

8.51. 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.

8.52. 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.

8.53. 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.

8.54. 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`.

8.55. 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`.

8.56. 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
```

8.57. 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
```

8.58. 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.

8.59. Require

Forces a 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
```

8.60. 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. The default is 0, disabling the check.

```
RobotLimit  200
```

8.61. Route

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

8.62. 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
```

8.63. 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
sub {
    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
sub {
    return {
        DEFAULT => 0.0,
        IL => 0.075,
        OH => 0.065,
    };
}
EOR
```

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

8.64. 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`.

8.65. 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
```

8.66. 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
```

8.67. 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.

8.68. 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
```

8.69. 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.

8.70. 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).

8.71. 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`.

8.72. SessionDB

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

8.73. SessionExpire

A customer can exit their browser or leave the catalog pages at any time, and no indication is given to the HTTPD 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.

8.74. 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.

8.75. SessionType

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

```
DB_File     Berkeley DB
DBI         DBI (don't use this!)
File        File-based sessions (the default)
GDBM        GDBM
```

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

8.76. 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
```

8.77. SpecialPageDir

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

```
SpecialPageDir      pages/special
```

8.78. Static

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

```
Static   Yes
```

8.79. 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
```

8.80. 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
```

8.81. 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
```

8.82. 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
```

8.83. 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
```

8.84. 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
```

8.85. 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
```

8.86. 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
```

8.87. 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`.

8.88. 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.

8.89. 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
```

8.90. 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
```

8.91. 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
```

8.92. 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."

8.93. 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
```

8.94. 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
```

8.95. VendURL

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

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

8.96. WideOpen

Disables IP qualification of user sessions. THIS DEGRADES CATALOG SECURITY. Do not use unless using PGP/CreditCardAuto or CyberCash.

9. ADMINISTERING INTERCHANGE

Some utilities are supplied in the VendRoot/bin directory:

```
compile_link Compiles an Interchange vlink or tlink CGI link
dump         Dumps the session file for a particular catalog
expire       Expires sessions for a particular catalog
expireall    Expires all catalogs
offline      Does offline build of the database(s)
update       Does in-place update of the database(s)
makecat      Make catalog
```

Some example scripts for other functions are in the `eg/` directory of the software distribution; see the readme file there.

Some thought should be given to where the databases, error logs, and session files should be located, especially on an ISP that might have multiple users sharing an Interchange server. In particular, put all of the session files and logs in a directory that is not writable by the user. This eliminates the possibility that the catalog may crash if the directory or file is corrupted.

To test the format of user catalog configuration files before restarting the server, set (from VendRoot):

```
bin/interchange -test
```

This will check all configuration files for syntax errors, which might otherwise prevent a catalog from booting. Once a catalog configures properly, user reconfiguration will not crash it. It will just cause an error. But, it must come up when the server is started.

9.1. Starting, Stopping, and Re-starting the Servers

The following commands need to have VENDROOT changed to the main directory where Interchange is installed. If the Interchange base directory is `/home/interchange/`, the start command would be `/home/interchange/bin/interchange`.

Do a `perldoc VENDROOT/bin/interchange` for full documentation.

To start the server with default settings:

```
VENDROOT/bin/interchange
```

It is usually best to issue a restart, otherwise the server will not run anew if a server is already running.

```
VENDROOT/bin/interchange -restart
```

Assuming the server starts correctly, the names of catalogs as they are configured will be displayed, along with a message stating the process ID it is running under.

To re-start the server:

```
VENDROOT/bin/interchange -restart
```

`-r` is the same as `-restart`.

This is typically done to force Interchange to re−read its configuration. A message will be displayed stating that a TERM signal has been sent to the process ID the servers are running under. This information is also sent to `home/interchange/error.log`. Check the error.log file for confirmation that the server has restarted properly.

To stop the server:

```
VENDROOT/bin/interchange −stop
```

A message will be displayed stating that a TERM signal has been sent to the process ID the server is running under. This information is also sent to `/home/interchange/error.log`.

Because processes waiting for selection on some operating systems block signals, they may have to wait for HouseKeeping seconds to stop. The default is 60.

To terminate the Interchange server with prejudice, in the event it will not stop:

```
VENDROOT/bin/interchange −kill
```

9.2. UNIX and INET modes

Both UNIX−domain and INET−domain sockets can be used for communication. INET domain sockets are useful when more than one server, connected via a local−area network (LAN), is used for accessing an Interchange server.

IMPORTANT NOTE: When sending sensitive information like credit card numbers over a network, always ensure that the data is secured by a firewall, or that the Interchange server runs on the same machine as any SSL−based server used for encryption.

If only running with one method of communication, use the −i and −u flags.

```
# Start only in UNIX mode
VENDROOT/bin/interchange −r −u

# Start only in INET mode
VENDROOT/bin/interchange −r −i
```

9.3. User Reconfiguration

The individual catalogs can be reconfigured by the user by running the [reconfig] support tag. This should be protected by one of the several forms of Interchange authentication, preferably by HTTP basic authorization. See `RemoteUser`.

The command line can be reconfigured (as the Interchange user) with:

```
VENDROOT/bin/interchange −reconfig <catalog>
```

It is easy for the administrator to manually reconfigure a catalog. Interchange simply looks for a file `etc/reconfig` (based in the Interchange software directory) at HouseKeeping time. If it finds a script name that matches one of the catalogs, it will reconfigure that catalog.

9.4. Making the Product Database

The DBM product databases can be built offline with the `offline` command. The directory to be used for output is specified either on the command line with the `-d` option, or is taken from the `catalog.cfg` directive `OfflineDir` -- `offline` in the catalog directory by default. The directory must exist. The source ASCII files should be present in that directory, and the DBM files are created there. Existing files will be overwritten.

```
offline -c catalog [-d offline_dir]
```

Do a `perldoc VENDROOT/bin/offline` for full documentation.

9.5. Updating Individual Records

If it takes a long time to build a very large DBM database, consider using the `bin/update` script to change just one field in a record, or to add from a corrections list.

The following updates the products database `price` field for item 19-202 with the new value 25.00

```
update -c catalog -f price 25.00
```

More than one field can be updated on a single command line.

```
update -c catalog -f price -f comment 25.00 "That pitchfork couple"
```

The following takes input from `file`, which must be formatted exactly like the original database, and adds/corrects any records contained therein.

```
update -c catalog -i file
```

Invoke the command without any arguments for a usage message describing the options.

9.6. Expiring Sessions

If Interchange is using DBM capability to store the sessions, periodically expire old sessions to keep the session database file from growing too large.

```
expire -c catalog
```

There is also an `expireall` script which reads all catalog entries in `interchange.cfg` and runs `expire` on them. The `expire` script accepts a `-r` option which tells it to recover lost disk space.

On a UNIX server, add a crontab entry such as the following:

```
# once a day at 4:40 am
40 4 * * *    perl /home/interchange/bin/expireall -r
```

Interchange will wait until the current transaction is finished before expiring, so this can be done at any time without disabling web access. Any search paging files for the affected session (kept in `ScratchDir`) will be removed as well.

With Windows or other operating systems which don't fork(), stop the server before running `expire`. This will prevent corruption of the database.

If not running DBM sessions, use a Perl script to delete all files not modified in the last one or two days. The following will work if given an argument of a session directory or session files:

```
#!perl
# expire_sessions.pl -- delete files 2 days old or older

my @files;
my $dir;
foreach $dir (@ARGV) {
    # just push files on the list
    if (-f $dir) { push @files, $_; next; }

    next unless -d $dir;

    # get all the file names in the directory
    opendir DIR, $dir or die "opendir $dir: $!\n";
    push @files, ( map { "$dir/$_" } grep(! /^\.\.?$/, readdir DIR) ) ;
}

for (@files) {
    unless (-f $_) {
        warn "skipping $_, not a file.\n";
        next;
    }
    next unless -M $_ >= 2;
    unlink $_ or die "unlink $_: $!\n";
}
```

It would be run with a command invocation like:

```
perl expire_sessions.pl /home/you/catalogs/simple/session
```

Multiple directory names are acceptable, if there is more than one catalog.

This script can be adjusted as necessary. Refinements might include reading the file to "eval" the session reference and expire only customers who are not members.

10. DEBUGGING

No debug output is provided by default. The source files contain commented−out '::logDebug(SOMETHING)' statements which can be edited to activate them. Set the value of `DebugFile` to a file that will be written to:

```
DebugFile /tmp/mvdebug
```

11. SETUP FOR HTTP SERVERS

Interchange requires a web server that is already installed on a system. It does have an internal server which can be used for administration, testing, and maintenance, but this will not be useful or desireable in a production environment.

As detailed previously, Interchange is always running in the background as a daemon, or resident program. It monitors either a UNIX−domain file−based socket or a series of INET−domain sockets. The small CGI link program, called in the demo `simple`, is run to connect to one of those sockets and provide the link to a browser.

Note: Since Apache and other CERN/NCSA−derived servers are the most popular, these instructions will focus on them. If using another type of web server, some translation of terms may be necessary. For instance, on MS Personal Web server, the standard `ScriptAlias` is `/scripts`.

A `ScriptAlias` or other CGI execution capability is needed to use the link program. (The default `ScriptAlias` for many web servers is `/cgi−bin`.) If `ExecCGI` is set for all directories, then any program ending in a particular file suffix (usually `.cgi`) will be seen as a CGI program.

Interchange, by convention, names the link program the same name as the catalog ID, though this is not required. In the distribution demo, this would yield a program name or SCRIPT_PATH of `/cgi−bin/simple` or `/simple.cgi`. This SCRIPT_PATH can be used to determine which Interchange catalog will be used when the link program is accessed.

11.1. UNIX−Domain Sockets

This is a socket which is not reachable from the Internet directly, but which must come from a request on the server. The link program `vlink` is the provided facility for such communication with Interchange. This is the most secure way to run a catalog, for there is no way for systems on the Internet to interact with Interchange except through its link program.

The most important issue with UNIX−domain sockets on Interchange is the permissions with which the CGI program and the Interchange server run. To improve security, Interchange normally runs with the socket file having 0600 permissions (rw−−−−−−−), which mandates that the CGI program and the server run as the same user ID. This means that the `vlink` program must be SUID to the same user ID as the server executes under. (Or that CGIWRAP is used on a single catalog system).

With Interchange's multiple catalog capability, the permissions situation gets a bit tricky. Interchange comes with a program, `makecat`, which configures catalogs for a multiple catalog system. It should properly set up ownership and permissions for multiple users if run as the superuser.

11.2. INET−Domain Sockets

These are sockets which are reachable from the Internet directly. The link program `tlink` is the provided facility for such communication with Interchange. Other browsers can talk to the socket directly if mapped to a catalog with the global `TcpMap` directive. To improve security, Interchange usually checks that the request comes from one of a limited number of systems, defined in the global `TcpHost` directive. (This check is not

made for the internal HTTP server.)

11.3. Internal HTTP Server

If the socket is contacted directly (only for INET−domain sockets), Interchange will perform the HTTP server function itself, talking directly to the browser. It can monitor any number of ports and map them to a particular catalog. By default, it only maps the special catalog mv_admin, which performs administrative functions. The default port is 7786 (ASCII for M an V), which is the default compiled into the distribution tlink program. This port can be changed via the `TcpMap` directive.

To prevent catalogs that do not wish access to be made in this way from being served from the internal server, Interchange has a fixed SCRIPT_PATH of `/catalogname` (/simple for the distribution demo) which needs to be placed as an alias in the Catalog directive to enable access. See `TcpMap` for more details.

11.4. Setting Up VLINK and TLINK

The `vlink` and `tlink` programs, compiled from `vlink.c` and `tlink.c`, are small C programs which contact and interface to a running Interchange daemon. The VLINK executable is normally made setuid to the user account which runs Interchange, so that the UNIX−domain socket file can be set to secure permissions (user read−write only). It is normally not necessary for the user to do anything. They will be compiled by the configuration program. If the Interchange daemon is not running, either of the programs will display a message indicating that the server is not available. The following defines in the produced `config.h` should be set:

LINK_FILE

Set this to the name of the socket file that will be used for configuration, usually "/usr/local/lib/interchange/etc/socket" or the "etc/socket" under the directory chosen for the VendRoot.

LINK_HOST

Set this to the IP number of the host which should be contacted. The default of 127.0.0.1 (the local machine) is probably best for many installations.

LINK_PORT

Set this to the TCP port number that the Interchange server will monitor. The default is 7786 (the ASCII codes for 'M' and 'V') and does not normally need to be changed.

LINK_TIMEOUT

Set this to the number of seconds `vlink` or `tlink` should wait before announcing that the Interchange server is not running. The default of 45 is probably a reasonable value.

11.5. Compiling VLINK and TLINK

There is a `compile_link` program which will assist with this. Do:

```
perldoc VENDROOT/bin/compile_link
```

for its documentation.

11.6. Manually Compiling VLINK and TLINK

Change directories to the `src` directory, then run the GNU configure script:

```
cd src
./configure
```

There will be some output displayed as the configure script checks the system. Then, compile the programs:

```
perl compile.pl
```

To compile manually:

```
cc vlink.c -o vlink
cc tlink.c -o tlink
```

On manual compiles, ensure that the C compiler will be invoked properly with this little ditty:

```
perl -e 'do "syscfg"; system("$CC $LIBS $CFLAGS $DEFS -o tlink tlink.c");'
perl -e 'do "syscfg"; system("$CC $LIBS $CFLAGS $DEFS -o vlink vlink.c");'
```

On some systems, the executable can be made smaller with the strip program, if available. It is not required.

```
strip vlink
strip tlink
```

If Interchange is to run under a different user account than the individual configuring the program, make that user the owner of `vlink`. Do not make `vlink` owned by root, because making `vlink` SETUID root is an huge and unnecessary security risk. It should also not normally run as the default Web user (often `nobody` or `http`)).

```
chown interchange vlink
```

Move the `vlink` executable to the cgi-bin directory:

```
mv vlink /the/cgi-bin/directory
```

Make `vlink` SETUID:

```
chmod u+s /the/cgi-bin/directory/vlink
```

Most systems unset the SUID bit when moving the file, so change it after moving.

The SCRIPT_NAME, as produced by the HTTP server, must match the name of the program. (As usual, let the makecat program do the work.)

12. FREQUENTLY ASKED QUESTIONS

12.1. 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.

12.2. VLINK or TLINK Compile Problems

(None of this is valid for Windows. Use the precompiled `tlink.exe or tlink.pl`.)

The latest version of `vlink.c` and `tlink.c` have been compiled on the following systems:

```
AIX 4.1
BSD2.0 (Pentium/x86)
Digital Unix (OSF/Alpha)
FreeBSD 2.x and 3.x
IRIX 5.3, IRIX 6.1
Linux
SCO OpenServer 5.x
Solaris 2.x (Sun compiler and GCC)
Solaris 7 (Sun compiler and GCC)
SunOS 4.1.4
Red Hat 6.2, 7.0
```

Some problems may occur. In general, ignore warnings about pointers.

Make sure that you have run the configure program in the src directory. If you use Interchange's makecat program, it will try to compile an appropriate link at that time, and will substitute tlink.pl if that doesn't work.

You can compile manually with the proper settings with this series of commands:

```
cd src
```

```
./configure
perl -e 'do "syscfg"; system ("$CC $CFLAGS $DEFS $LIBS -o tlink tlink.c")'
perl -e 'do "syscfg"; system ("$CC $CFLAGS $DEFS $LIBS -o vlink vlink.c")'
```

There is also a `compile_link` program which has docmentation embedded and which will compile an approprate link. If you cannot compile, try using the `tlink.pl` script which should work on most any system.

12.3. 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 `ichange` or `interchange` 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
```

12.4. 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/start
```

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

```
Linux, SunOS, 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
```

Permissions are easy. If you, for some reason, can't run VLINK as an SUID program (SUID to your user ID, not to root), you will need to make permissions more permissive. See the Interchange documentation for how to use the ReadPermission and WritePermission directives to make the session files and sockets 'group' or 'world' readable and writable. If you can make your UID, or the one you have chosen to own Interchange, the same group as your HTTP user, you can make the files group read-write. If you can't, you will have to set the dreaded 'world' permissions.

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.

12.5. 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.

12.6. Your demo has a very large price for the first item, and when I change the quantity the price drops precipitously.

This was an attempt at humor in the older `simple` demo catalog. Check out the Quantity Pricing feature of Interchange. Try looking at:

```
http://interchange.akopia.com/cgi-bin/simple/quantity?;00-0011
```

12.7. Can't locate lib.pm in @INC. BEGIN failed--compilation aborted.

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.

12.8. Can I run Interchange on my Macintosh?

Interchange will not run on a Macintosh running System 7 or other "Finder" operating system.

After some manual effort to install the proper Perl modules, Interchange has been run on the Macintosh OS/X server (DP 3). Given the degree of effort involved this is not considered a tested configuration.

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. But, you must run it on UNIX or another Mac operating system. Interchange does run on MkLinux, Linux PPC, and OS/X.

12.9. Shopping cart is dropped when using SSL.

This is usually 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).

12.10. 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.

12.11. 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.

In any case, it has nothing to do with Interchange when you are running on UNIX. Interchange does not include any compiled libraries.

12.12. Interchange doesn't work with RedHat 5.0 (glibc).

This is a defective Perl distributed by RedHat. The problem is that they don't take GLIBC into account. It will pass make test 100 percent, but strftime dumps core on a segfault. You shouldn't run RedHat 5.0 anyway. It has drastic security problems. Update to a later version and build a new Perl.

13. ISP PROBLEMS

Great strides have been made since Interchange's early days. 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.

13.1. No shell access allowed on my ISP.

This used to mean that you were totally out of luck. You probably still are, but it is at least possible to run Interchange on a virtual server such as HWY.NET. You should be able to use the tlink.pl program and INET mode to start using Interchange with quite a bit of work. You will need to hand−edit all of the configuration files, and use tlink.pl with the appropriate settings. Long term, you should put a compiled tlink in place.

You will also have to start the Interchange daemon via a CGI program that you write. Bottom line: don't try this unless most people call you a wizard. The Interchange web site has pointers to ISPs who happily host Interchange.

13.2. No C compiler to compile VLINK or SVEND.

Since VLINK needs certain values set in the source before compilation, a pre−compiled version will not work. It is recommended that you get the latest version of Interchange and run in INET mode along with the tlink.pl Perl−based link program. A pre−compiled tlink.c might also work for you, if you can use the defaults of 'localhost' and port 7786.

13.3. Perl version too low.

Interchange requires Perl 5.005 or higher. Perl 5.6.0 or higher is strongly recommended.

13.4. 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.

13.5. 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.

13.6. My provider runs Windows NT.

Interchange is unlikely to work on Windows NT ISPs in the near future. If you have a dedicated NT machine, there is a version of Interchange for Windows NT. It may work for you, but there are differences in operation from the UNIX version (mostly due to NT's lack of fork() or and lack of user−space threading support in Perl).

13.7. 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.

14. SETUP QUESTIONS

Interchange is quite powerful, but also quite complex. There are many possibilities opened up with its conditional HTML, but also many pitfalls. Read the documentation thoroughly, then inspect the `basic` and `simple` demo pages for help. Pay particular attention to:

```
browse*.html
flypage.html
results.html
ord/basket.html
ord/checkout.html
```

These pages use many of the powerful features of Interchange, and studying them will be instructive.

14.1. 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.

14.2. 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. 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.

14.3. 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.

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

Interchange has product modifiers, or attributes, which can be carried around with the product. The [item−accessories attribute] tag will automatically place a suitable select widget on an HTML form, and "remember" what should be selected. See the Interchange documentation for Item Attributes.

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.) This makes size/color handling much easier for multiple items. They can be stacked, which means that you can add multiple instances of a size or color to an individual product within a shopping cart entry line, but this is too much trouble.

The attribute information is placed on the form with form fields, like:

```
<INPUT TYPE=XXXXX NAME="[modifier−name size]" VALUE="[item−modifier size]">
```

The embedded Perl capability of Interchange means that you can discover the number of items on order of each type, and build quite detailed input forms. The Interchange demos have some examples which should help you. Try ordering the T−Shirt and visiting its flypage.

14.5. 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:
>
```

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 GNUPGPATH) to orient the program correctly.

Trust problems

If the key you are sending has not been signed by another (trusted) user, then you will get error returns from some flavors of PGP, which causes the problem. Sometimes the fix is to route the error output to /dev/null (at least on UNIX) by changing the EncryptProgram directive in catalog.cfg:

```
EncryptProgram  /usr/bin/pgp −fat −r sales@your.com 2>/dev/null
```

For GNU Privacy Guard, GPG, you would use something like:

```
EncryptProgram /usr/local/bin/gpg -fa --always-trust -r sales@your.com
```

On Windows or other operating systems, the only fix is to sign the key or otherwise prevent the error from occurring.

14.6. 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
```

On Windows, it might be:

```
c:\mycommands\pgp -m %s
```

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

14.7. 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]
```

14.8. 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. Though Akopia likes to get reports of malfunctions in the demo, they feel absolutely no urgency about fixing or responding to such reports.

14.9. Difficulty interacting with browser: Request method for form submission is not POST

There are many possible causes for this error. The main ones are:

- This most common case is where the user bookmarks a shopping cart after Recalc or bookmarks a checkout page. Those pages are usually the results of form submissions, and are not really a candidate for a bookmark. You should make sure you provide links to both of those pages on the interact special page (in special/interact if you have started with the demo).
- User hits BACK with a very small cache. There are quite a few users who have decided 10Kbytes is enough of a cache, and that isn't adequate. (Actually, they probably assumed the number to enter was in megabytes.)
- One possible cause is that you must define METHOD=POST in all of your forms, not METHOD=post. The lower case form is not recognized in the HTTP spec.
- If you are using SSL and don't have a later HTTP server (released after roughly September 1997), there is a bug in earlier versions Microsoft Internet Explorer's version of SSL3. Stronghold 2.1.1 works around this bug. So, if you use Stronghold, you should update.

Interchange also allows you to have forms with METHOD=GET. To use, set the session ID in your form:

```
<INPUT TYPE=hidden NAME=mv_session_id VALUE="[data session id]">
```

You might then try METHOD=GET on forms that have problems. This may prevent errors due to one of the causes above.

14.10. 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:

```
<A HREF="http://yourcatalog.com/cgi-in/yourcat/sp_offer?mv_pc=Source1">
    Special offer!&lt;/A>
```

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

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

14.11. 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.

14.12. 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.

14.13. Why isn't the error more enlightening?

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

14.14. We need some more information! Problem: unspecified error

This is a fault in the order profile, and can be due to any of the following:

- Problems with encryption.
- Inadvertently setting the same variable twice on the form, causing problems due to the embedded null.
- Changing the checkout profile and not testing properly under all conditions. The order profile must not have non−blank lines that don't contain valid check procedures (excluding comments). If you are using a Perl routine in the profile, as above, you must take care that the routine returns a valid check routine.

14.15. 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.

14.16. 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.

14.17. 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."

15. 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.

15.1. 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.

15.2. 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 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.

15.3. The browse.html page (from the demo) is SLOW+!

That is because it is only useful for small product lines. It pulls out all keys from the entire database and iterates over them, which is not reasonable for more then 50 products or so (depending on your server speed). Look at the browse1.html page instead.

A better approach is to use a search and browse by category, an example of which is given in the demo.

15.4. I am using SQL, and Interchange is so 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.

15.5. My session files change to owner root every day!

You have the expireall `−r` entry in the root crontab, and it should either be in the Interchange user crontab or run as:

```
44 4 * * * su −c "/INTERCHANGE_ROOT/bin/expireall −r" INTERCHANGE_USERNAME
```

WHEN IN DOUBT, RE−START 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.

16. AVAILABILITY AND COMPATIBILITY

Interchange 4.6 has been tested on Solaris 2, Linux, FreeBSD, and Digital UNIX/Tru64. Its predecessor, Minivend 3, also reportedly runs on SCO, AIX, and other Unix ports. It should work on any UNIX or Win32 OS with Perl 5.005 or higher.

IMPORTANT NOTE FOR BSD USERS: Interchange has been known to segfault when reading sockets on Perl. This occurs more frequently on a busy system. The workaround is to set in interchange.cfg:

```
MaxServers   0
Housekeeping 1
```

Windows does not support fork(), so the server runs in the foreground. There are other differences, and there may be portablity problems. Interchange is not yet tested to run on Windows; the canonical installation is UNIX.

17. DEMO APPLICATIONS

There are two demo applications included. One is not so aptly named "simple," while the other other is named "barry."

To install the demo, go to the directory where Interchange is installed (default is "mvend" in the home directory or /usr/local/interchange for root installations) and run:

```
bin/makecat barry
```

or

```
bin/makecat simple
```

or

```
bin/makecat any_id_you_pick
```

Follow the prompts and after restarting the Interchange server, the demo catalog should be accessable.

18. KNOWN PROBLEMS

BSDI, FreeBSD, Net BSD: make sure the following are in effect in interchange.cfg:

```
MaxServers 0
Housekeeping 1
```

Expect lots of crashes if this isn't done.

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.

Windows: Interchange will not work properly with Windows. No strategy has been yet developed for providing reliable operation on Windows.

- Akopia and Interchange are registered trademarks of Akopia, Inc. All other product names are trademarks or registered trademarks of their respective manufacturers. This version of the document supersedes any and all previous versions.