Advanced Interchange Topics

Table of Contents

1. Advanced Interchange Topics

- Maintaining production Interchange servers
- Interchange Administration Tool Development
- Making catalog skeletons for use with makecat
- Building custom link programs

2. Maintaining Interchange

Some utilities are supplied in the VendRoot/bin directory:

```
compile_link Compiles an Interchange vlink or tlink CGI link
configdump   Dumps the configuration directives for a particular catalog
dump         Dumps the session file for a particular catalog
expire       Expires sessions for a particular catalog
expireall    Expires all catalogs
makecat      Make catalog
```

Some example scripts for other functions are in the `eg/` directory of the software distribution.

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.

2.1. 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
```

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.

It is usually best to issue a restart instead. It doesn't hurt to do a restart if you're actually starting the first time. And, if a server is already running (from this VENDROOT), a new start attempt will fail. To restart the server:

```
VENDROOT/bin/interchange −restart
```

The `−r` option 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 `VENDROOT/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 `VENDROOT/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
```

2.2. 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 web 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.

Use the −i and −u flags if you only want to use one communication method:

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

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

2.3. 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.

2.4. 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.

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.

2.5. 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
```

3. Interchange Components

Interchange components are merely portions of HTML/ITL that are included into pages within the site depending on options set in the Admin UI. The default component set includes the following:

```
best_horizontal
best_vertical
cart
cart_display
cart_tiny
category_vertical
cross_horizontal
cross_vertical
promo_horizontal
promo_vertical
random_horizontal
random_vertical
upsell_horizontal
upsell_vertical
```

3.1. Content −> Page Edit

The Interchange Admin UI offers a page editor function that allows component definitions and options to be modified for each page within the catalog.

3.1.1. Template

The choices for the Template drop−down are read from template definition files located in the CATROOT/template directory. These files store the name and description of the template, as well as components and options for the particular template.

To create a new template for use in the admin, it is best to copy an existing template definition to a new file name and edit it's contents to suit. Once the catalog is reconfigured, the new choice will be visible within the Content Page Editor admin function.

Each template option is looped through and a scratch is set using its same name and value.

ITL is used near the bottom of this file to set each option to default values before the page is displayed.

```
[set page_title][set]
[set page_banner][set]
[set members_only][set]
[set component_before][set]
[set component_after][set]
[set bgcolor]#FFFFFF[/set]
```

3.1.2. Page Title

Sets the title of the page which is synonymous with the html <title></title> code.

The following code within the template definition file is used to display this option within in the content editor:

page_title: description: Page title

This code dynamically adds the title to the page:

<title>[scratch page_title]</title>

3.1.3. Page Banner

Sets a textual title for each page which is called by [either][scratch page_banner][or][scratch page_title][/either] This results in the Page Banner being displayed if defined. Otherwise, the Page Title is used.

3.1.4. Members Only

The members only function is handled by the following code within each template file:

```
[if scratch members_only]
    [set members_only][/set]
    [if !session logged_in]
    [set mv_successpage]@@MV_PAGE@@[/set]
    [bounce page=login]
    [/if]
[/if]
```

This code says if members only is set to yes and the visitor is logged in, display the page. Otherwise, redirect the visitor to the login page.

3.1.5. Break 1

This code causes a separation in the Content Editor between the next set of options. (A blue line)

3.1.6. Horizontal Before Component

This allows the inclusion of a defined component to be displayed before, or above, the page's content. It is called with the following code within the LEFTRIGHT_TOP template:

```
[if scratch component_before]
[include file="templates/components/[scratch component_before]"]
[set component_before][/set]
[/if]
```

3.1.7. Horizontal After Component

This function allows the inclusion of a defined component to be displayed after or below the pages's content. It's called with the following code within the LEFTRIGHT_BOTTOM template:

```
[if scratch component_after]
[include file="templates/components/[scratch component_after]"]
[set component_after][/set]
[/if]
```

3.1.8. Horizontal Item Width

This setting allows you to choose how many items the horizontal components display. For example, the horizontal best sellers component uses this setting to randomly select the best sellers. Notice the default to 2 if nothing is defined.

```
random="[either][scratch component_hsize][or]2[/either]"
```

3.1.9. Special Tag

This setting is only viable when a promotion is used for a horizontal component. It tells the promotional component which rows to evaluate in the merchandising table for display within the component. This setting normally corelates to the featured column of the merchandising table as follows:

```
[query arrayref=main
          sql="
               SELECT sku,timed_promotion,start_date,finish_date
               FROM merchandising
               WHERE featured = '[scratch hpromo_type]'
               "][/query]
```

3.1.10. Before/After Banner

Allows a title for the horizontal components to be defined to displayed in a header above the component's items. It is called with the [scratch hbanner] tag.

3.1.11. Break 2

This code causes a separation in the Content Editor between the next set of options. (A blue line)

3.1.12. Vertical Component

Defines a component to be displayed along the right side of the LEFTRIGHT_BOTTOM template. It is called from the template with the following code:

```
[include file="templates/components/[scratch component_right]"]
```

3.1.13. Vertical Items Height

Sets the number of items to display within the vertical component. Called with the following code:

```
random="[either][scratch component_vsize][or]3[/either]"
```

3.1.14. Right Banner

Allows a title to be set for a vertical component which is displayed as a header above the items in the vertical component. It's called with the [scratch vbanner] tag.

3.1.15. Special Tag

Essentially the same as the Special Tag setting described in item number 9 above.

3.1.16. Background Color

Allows the background color of the page to be selected. The choices are stored within the page or template definition as in:

```
bgcolor:
        options: #FFFFFF=White,pink=Pink
        widget: select
        description: Background color
```

3.1.17. Content

Allows the page code to be downloaded, uploaded and viewed/edited. Only the code between <!−− BEGIN CONTENT −−> and <!−− END CONTENT −−> is displayed or can be edited here.

3.1.18. Preview, Save, and Cancel buttons

Allows the changes to the edited page to be previewed in a pop−up browser window, or saved. Cancel returns you to the page editor selection page.

3.1.19. Save template in page

Template settings are stored in the template definitions by default. This allows a common set of choices for template settings for all pages. If specific setting options are desired for a page, the template can be saved within the page so that it may have individual options.

The in−page template definition must be surrounded by [comment] [/comment].

3.2. Custom Admin UI Options

Other options may be added to the template by defining them in the default definition file, or using in−page definitions.

When the following lines are added to the template definition, the new option is added to the Admin UI.

```
option_name:
options: 1,2*,3
widget: select
description: Option Description
help: Other Details
```

Each time the template is used, an option_name scratch variable is created. (Called with: [scratch option_name].) This scratch value will be equal to what's selected here in the admin tool.

The possible widgets include:

```
break − produces the blue line separator.
```

```
radio − produces radio button type selections.
select − standard drop−down selector.
move_combo − select drop down with options and text input for new option.
```

4. Administrative Pages

With Interchange's `GlobalSub` capability, very complex add−on schemes can be implemented with Perl subroutines. And with the new writable database, pages that modify the catalog data can be made. See `MasterHost`, `RemoteUser`, and `Password`.

In addition, any Interchange page subdirectory can be protected from access by anyone except the administrator if a file called '.access' is present and non−zero in size.

4.1. Controlling Access to Certain Pages

If the directory containing the page has a file `.access` and that file's size is zero bytes, access can be gated in one of several ways.

1. If the file `.access_gate` is present, it will be read and scanned for page−based access. The file has the form:

```
page: condition
*: condition
```

The `page` is the file name of the file to be controlled (the .html extension is optional). The `condition` is either a literal `Yes/No` or Interchange tags which would produce a `Yes` or `No` (1/0 work just fine, as do true/false).
The entry for `*` sets the default action if the page name is not found. If pages will be allowed by default, set it to `1` or `Yes`. If pages are to be denied by default in this directory, leave blank or set to `No`. Here is an example, for the directory `controlled`, having the following files:

```
−rw−rw−r−−   1 mike     mike            0 Jan  8 14:19 .access
−rw−rw−r−−   1 mike     mike          185 Jan  8 16:00 .access_gate
−rw−rw−r−−   1 mike     mike          121 Jan  8 14:59 any.html
−rw−rw−r−−   1 mike     mike          104 Jan  8 14:19 bar.html
−rw−rw−r−−   1 mike     mike          104 Jan  8 14:19 baz.html
−rw−rw−r−−   1 mike     mike          104 Jan  8 14:19 foo.html
```

The contents of `.access_gate`:

```
foo.html: [if session username eq 'flycat']
            Yes
          [/if]
bar:      [if session username eq 'flycat']
          [or scratch allow_bar]
            Yes
          [/if]
baz:      yes
*:        [data session logged_in]
```

The page `controlled/foo` is only allowed for the logged−in user **flycat**.
The page `controlled/bar` is allowed for the logged−in user **flycat**, or if the scratch variable `allow_bar` is set to a non−blank, non−zero value.
The page `controlled/baz` is always allowed for display.
The page `controlled/any` (or any other page in the directory not named in `.access_gate`) will be allowed for any user logged in via *UserDB*. NOTE: The `.access_gate` scheme is available for database

access checking if the database is defined as an `AdminDatabase`. The `.access_gate` file is located in `ProductDir`.

1. If the Variable `MV_USERDB_REMOTE_USER` is set (non−zero and non−blank), any user logged in via the UserDB feature will receive access to all pages in the directory. NOTE: If there is a `.access_gate` file, it overrides this.
2. If the variables `MV_USERDB_ACL_TABLE` is set to a valid database identifier, the UserDB module can control access with simple ACL logic. See USER DATABASE. NOTE: If there is a `.access_gate` file, it overrides this. Also, if `MV_USERDB_REMOTE_USER` is set, this capability is not available.

4.2. display tag and mv_metadata

Interchange can store meta information for selected columns of tables in a site's database. This meta information is used when the user interacts with the database. For example, the meta informaton for a `Hide Item` field might specify that a checkbox be displayed when the user edits that field, since the only reasonable values are `on` and `off`. Or, the meta information might specify a filter on data entered for a `Filename` field which makes sure that the characters entered are safe for use in a filename.

`Widget type` specifies the `HTML INPUT` tag type to use when displaying the field in, say, the item editor.

`Width` and `Height` only apply to some of the `Widget type` options, for instance the `Textarea` widget.

`Label` is displayed instead of the internal column name. For example, the `category` column of the `products` table might have a label of `Product Category`.

`Help` is displayed below the column label, and helps describe the purpose of the field to the user.

`Help url` can be used to link to a page giving more information on the field.

`Lookup` can be used when a field is acting like a foreign key into another table. In that case, use some sort of select box as the widget type, and if referencing multiple rows in the destination table, use a multi select box, with `colons_to_null` as the `pre_filter`, and `::` as the `lookup_exclude`.

`Filter` and `pre_filter` can be used to filter data destined for that field or data read from that field, respectively.

Repeat?: The Interchange back office UI uses the mv_metadata table to discover formatting information for field, table, and view display. The information is kept in fields in the mv_metadata table, and is used to select the display, the HTML input type, the size (height and width where appropriate), label, help text, additional help URL, and default value display.

It works in conjunction with the [display ...] usertag defined in the Interchange UI as well as in specific pages in the UI. The [display] tag has this syntax:

[display table=tablename column=fieldname key=key arbitrary=tag filter=op ...]

In the simplest use, the formatting information for a table form field is called with:

[display table=products column=category key="os28007"]

The mv_metadata table is scanned for the following keys:

products::category::os28007
products::category

If a row is found with one of those keys, then the information in the row is used to set the display widget. If no row is found, an INPUT TYPE=TEXT widget is displayed. If the data is all digits, a size of 8 is used, otherwise the size is 60.

If the following row were found (not all fields shown, would be tab−separated in the actual data):

```
code                type   width height label     options
products::category  text   20           Category
```

Then this would be output:

```
<INPUT TYPE=text SIZE=20 VALUE="*category*">
```

If the following row were found:

```
code                type   width height label     options
products::category  select              Category  =none, product=Hardware
```

Then the following would be output:

```
<SELECT NAME=category>
<OPTION VALUE="">none
<OPTION VALUE="product">Hardware
</SELECT>
```

The standard widget types are:

text

The default. Uses the fields:

```
width   size of input box
```

textarea

Format a <TEXTAREA> </TEXTAREA> pair. Uses the fields:

```
width   COLS for textarea
height  ROWS for textarea
```

select

Format a <SELECT> </SELECT> pair with appropriate options. Uses the fields:

```
height          SIZE for select
default         Default for SELECTED
options         Options for a fixed list (or prepended to a lookup)
lookup          signals a lookup (used as field name if "field" empty)
field           field to look up possible values in
```

```
db              table to look up in if not current table
lookup_exclude  regular expression to exclude certain values from lookup
```

5. Usertag Reference

Admin Tool−specific usertags.

6. Admin Tool Database Tables

6.1. icmenu.txt

Used for back−office administration UI menus and wizards.

```
code
            Arbitrary primary key
mgroup
            Menu group (for grouping searches)
msort
            Sort order within the group
next_line
            Set to 1 if submenu
indicator
exclude_on
depends_on
page
form
name
super
inactive
special
help_name
    img_dn
    img_up
    img_sel
    img_icon
    url
```

6.2. mv_metadata.asc

```
code
    Table::Column to be operated on.
    Database table
type
    Widget type (Select the basic display type for the field)
        textarea = Textarea
        text = Text Entry (default)
        select = Select Box
        yesno = Yes/No (Yes=1)
        noyes = No/Yes (No=1)
        multiple = Multiple Select
        combo = Combo Select
        reverse_combo = Reverse Combo
        move_combo = Combo move
        display = Text of option
        hidden_text = Hidden(show text)
        radio = Radio box
        radio_nbsp = Radio (nbsp)
        checkbox = Checkbox
        check_nbsp = Checkbox (nbsp)
        imagedir = Image listing
        imagehelper = Image upload
        date = Date selector
        value = Value
        option_format = Option formatter
        show = Show all options
```

```
width
    Width (SIZE for TEXT, COLS for TEXTAREA, Label limit for SELECT)
height
    Height (SIZE for SELECT, ROWS for TEXTAREA)
field
    Field for lookup (can be two comma separated fields, in which case
    second is used as the label text.  Both must be in the same table.)
db
name
    Variable name (normally left empty, changes variable name to send in
    form)
outboard
    Select directory for image listing widget
options
    options in the format <blockquote>value=label*</blockquote>
attribute
    Column name (Do not set this.)
label
help
    Help (displays at top of page)
lookup
    Lookup select (Whether lookup is performed to get options for a select
    type.  If nothing is in the field, then used as the name of the field
    to lookup in.  Use lookup table if you want to look up in a different
    table.
filter
    Filters (Filters which can transform or constrain your data.  Some
    widgets require filters.)
help_url
    Help URL (links below help text)
    A URL which will provide more help
pre_filter
lookup_exclude
    ADVANCED: regular expression that excludes certain keys from the lookup
prepend
append
    Append HTML (HTML to be appended to the widget.  Will substitute in the
    macros _UI_TABLE_, _UI_COLUMN_, _UI_KEY_, and _UI_VALUE_, and will
    resolve relative links with absolute links.)
display_filter
```

7. makecat − Set Up a Catalog from a Template

After Interchange is installed, you need to set up at least one catalog. Interchange will not function properly until a catalog is created.

The supplied `makecat` script, which is in the Interchange program directory `bin`, is designed to set up a catalog based on the user's server configuration. It interrogates the user for parameters like which directories to use, a URL to base the catalog in, HTTP server definitions, and file ownership. It gives relevant examples of the entries it expects to receive.

Note: A catalog can only be created once. All further configuration is done by editing the files within the catalog directory.

The `makecat` script requires a catalog skeleton to work from. The Foundation demo is distributed with Interchange. See the icfoundation document for information on building the Foundation demo store. Other demo catalogs are available at `http://interchange.redhat.com/`.

It is not normally necessary for you to understand how to build catalog skeletons for use with makecat, but the following information will help you if you should ever need to.

7.1. Catalog Skeletons

A catalog skeleton contains an image of a configured catalog. The best way to see what the makecat program does is to configure the simple demo and then run a recursive `diff` on the template and configured catalog directories:

```
cd /usr/local/interchange
diff −r construct catalogs/construct
```

The files are mostly identical, except that certain macro strings have been replaced with the answers given to the script. For example, if `www.mydomain.com` was answered at the prompt for a server name, this difference would appear in the catalog.cfg file:

```
# template
Variable SERVER_NAME  __MVC_SERVERNAME__

# configured catalog
Variable SERVER_NAME  www.mydomain.com
```

The macro string __MVC_SERVERNAME__ was substituted with the answer to the question about server name. In the same way, other variables are substituted, and include:

```
MVC_BASEDIR      MVC_IMAGEDIR
MVC_CATROOT      MVC_IMAGEURL
MVC_CATUSER      MVC_MAILORDERTO
MVC_CGIBASE      MVC_MINIVENDGROUP
MVC_CGIDIR       MVC_MINIVENDUSER
MVC_CGIURL       MVC_SAMPLEHTML
MVC_DEMOTYPE     MVC_SAMPLEURL
MVC_DOCUMENTROOT MVC_VENDROOT
MVC_ENCRYPTOR
```

Note: Not all of these variables are present in the "construct" template, and more may be defined. In fact, any environment variable that is set and begins with MVC_ will be substituted for by the `makecat` script. For example, to set up a configurable parameter to customize the COMPANY variable in catalog.cfg, run a pre−qualifying script that set the environment variable MVC_COMPANY and then place in the catalog.cfg file:

Variable COMPANY __MVC_COMPANY__

All files within a template directory are substituted for macros, not just the catalog.cfg file. There are two special directories named `html` and `images`. These will be recursively copied to the directories defined as SampleHTML and ImageDir.

Note: The template directory is located in the Interchange software directory, i.e., where `interchange.cfg` resides. Avoid editing files in the template directory. To create a new template, it is recommended that it should be named something besides 'construct' and a copy of the `construct` demo directory be used as a starting point. Templates are normally placed in the Interchange base directory, but can be located anywhere. The script will prompt for the location if it cannot find a template.

In addition to the standard parameters prompted for by Interchange, and the standard catalog creation procedure, four other files in the `config` directory of the template may be defined:

```
additional_fields −− file with more parameters for macro substitution
additional_help   −− extended description for the additional_fields
precopy_commands  −− commands passed to the system prior to catalog copy
postcopy_commands −− commands passed to the system after catalog copy
```

All files are paragraph−based. In other words, a blank line (with no spaces) terminates the individual setting.

The additional_fields file contains:

```
PARAM
The prompt. Set PARAM to?
The default value of PARAM
```

This would cause a question during makecat:

```
The prompt. Set PARAM to?.....[The default value of PARAM]
```

If the additional_help file is present, additional instructions for PARAM may be provided.

```
PARAM

These are additional instructions for PARAM, and they
may span multiple lines up to the first blank line.
```

The prompt would now be:

```
These are additional instructions for PARAM, and they
may span multiple lines up to the first blank line.

The prompt. Set PARAM to?.....[The default value of PARAM]
```

If the file config/precopy_commands exists, it will be read as a command followed by the prompt/help value.

```
mysqladmin create __MVC_CATALOGNAME__
We need to create an SQL database for your Interchange
database tables.
```

This will cause the prompt:

```
We need to create an SQL database for your Interchange
database tables.

Run command "mysqladmin create simple"?
```

If the response is "y" or "yes," the command will be run by passing it through the Perl system() function. As with any of the additional configuration files, MVC_PARAM macro substitution is performed on the command and help. Proper permissions for the command are required.

The file config/postcopy_commands is exactly the same as <precopy_commands>, except the prompt occurs after the catalog files are copied and macro substitution is performed on all files.

There may also be SubCatalog directives:

```
SubCatalog easy simple /home/catalogs/simple /cgi-bin/easy
```

easy

The name of the subcatalog, which also controls the name of the subcatalog configuration file. In this case, it is `easy.cfg`.

simple

The name of the base configuration that will be the basis for the catalog. Parameters in the easy.cfg file that are different will override those in the catalog.cfg file for the base configuration.

The remaining parameters are similar to the Catalog directive.

Additional interchange.cfg parameters set up administrative parameters that are catalog wide. See the server configuration file for details on each of these.

Each catalog can be completely independent with different databases, or catalogs can share pages, databases, and session files. This means that several catalogs can share the same information, allowing "virtual malls."

7.2. Manual Installation of Catalogs

An Interchange installation is complex, and requires quite a few distinct steps. Normally you will want to use the interactive catalog builder, `makecat`, described above. It makes the process much easier. Please see the `iccattut` document for a full tutorial on building a catalog by hand.

8. Link Programs

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 is the most popular web server, these instructions will focus on it. If using another type of web server, some translation of terms may be necessary.

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.

8.1. 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.

8.2. 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.)

8.3. 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, 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.

8.4. 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 decimal 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.

8.5. Compiling VLINK and TLINK

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

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

for its documentation.

8.6. 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.)

8.7. VLINK or TLINK Compile Problems

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

```
AIX 4.1
BSD2.0 (Pentium/x86)
```

```
Debian GNU/Linux
Digital Unix (OSF/Alpha)
FreeBSD 2.x, 3.x, 4.x
IRIX 5.3, IRIX 6.1
OpenBSD 2.7
Red Hat Linux 6.2, 7.0, 7.1
SCO OpenServer 5.x
Solaris 2.x (Sun compiler and GCC)
Solaris 7 (Sun compiler and GCC)
SunOS 4.1.4
```

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, written in Perl instead of C, which should work on most any system. Since vlink needs to have values set before compilation, a pre−compiled version will not work unless it has the exact values you need on your system. If you can use the defaults of 'localhost' and port 7786, you may be in luck.

9. Installing Perl Modules without Root Access

Installing Interchange without root access is no problem. However, installing Perl modules without root access is a little trickier.

You must build your makefile to work in your home dir. Something like:

```
PREFIX=~/usr/local \
INSTALLPRIVLIB=~/usr/local/lib/perl5 \
INSTALLSCRIPT=~/usr/local/bin \
INSTALLSITELIB=~/usr/local/lib/perl5/site_perl \
INSTALLBIN=~/usr/local/bin \
INSTALLMAN1DIR=~/usr/local/lib/perl5/man \
INSTALLMAN3DIR=~/usr/local/lib/perl5/man/man3
```

Put this in a file, say 'installopts', and use it for the Makefile.PL.

```
perl Makefile.PL `cat installopts`
```

Then, forget ./config. Just do:

```
make
make test
make install
```

Some of the tests may fail, but that's probably ok.

Also make sure to install Bundle::Interchange, which will need the same config data as you put into 'installopts'.

10. Installation Troubleshooting

Interchange uses the services of other complex programs, such as Perl, Web servers, and relational databases, to work. Therefore, when there is a problem, check these programs before checking Interchange. Many more basic installation problems have to do with those than with Interchange itself.

If an error message is received about not being able to find libraries, or a core dump has occurred, or a segment fault message, it is always an improperly built or configured Perl. Contact the system administrator or install a new Perl.

The `makecat` program is intended to be used to create the starting point for the catalog. If the demo does not work the first time, keep trying. If it still does not work, try running in INET mode.

Check the two error log files: `error.log` in the Interchange home directory (where interchange.cfg resides) and `error.log` in the catalog directory (where catalog.cfg resides; there can be many of these). Many problems can be diagnosed quickly if these error logs are consulted.

Check the README file, the FAQ, and mail list archive at the official Interchange web site for information:

```
http://interchange.redhat.com/
```

Double check the following items:

1. Using UNIX sockets?
   - Check that the `vlink` program is SUID, or the appropriate changes have been made in the SocketPerms directive. Unless the files are world−writable, the vlink program and the Interchange server must run as the same user ID! If running CGI−WRAP or SUEXEC, the `vlink` program must not be SUID.
   - If having trouble with the vlink program (named `construct` in the demo configuration), try re−running `makecat` and using INET mode instead. (Or copy the `tlink` INET mode link program over `vlink`). This should work unchanged for many systems.
   - If using an ISP or have a non−standard network configuration, some changes to interchange.cfg are necessary. For `tlink` to work, the proper host name(s) must be configured into the TcpHost directive in interchange.cfg. The program selects port 7786 by default (the ASCII codes for "M" and "V"). If another port is used, it must be set to the same number in both the tlink program (by running `compile_link`) and the `minivend.cfg` file. The `tlink` program does not need to be SUID.
2. Proper file permissions?
   - The Interchange server should not run as the user `nobody`! The program files can be owned by anyone, but any databases, ASCII database source files, error logs, and the directory that holds them must be writable by the proper user ID, that is the one that is executing the MiniVend program.
   - The best way to operate in multi−user, multiple catalog setups is to create a special `interch` user, then put that user in the group that contains each catalog user. If a group is defined for each individual user, this provides the best security. All associated files can be in 660 or 770 mode. There should be no problems with permissions and no problems with security.
3. Is the `vlink` program being executed on a machine that has the socket file `etc/socket` on a directly attached disk?
   - UNIX−domain sockets will not work on NFS−mounted file systems! This means that the server `minivend` and the CGI program `vlink` must be executing on the same machine.

- The `tlink` program does not have this problem, but it must have the proper host name(s) and TCP ports set in the TcpHost and TcpPort directives in `interchange.cfg`. Also, be careful of security if sensitive information, like customer credit card numbers, is being placed on a network wire.