Perl Development Best Practices

For Blackhills Corporation

[Introduction and Objective 4](#_Toc443054943)

[References 4](#_Toc443054944)

[Server Environment 4](#_Toc443054945)

[Apache Webserver Configuration 4](#_Toc443054946)

[CGI Application Source Code Organization 4](#_Toc443054947)

[Perl Libraries 4](#_Toc443054948)

[Development Libraries 4](#_Toc443054949)

[System Auxiliary CPAN Library 5](#_Toc443054950)

[Custom BHC Developer Library 5](#_Toc443054951)

[CGI Common Library 5](#_Toc443054952)

[Develop and Test With Various Libraries 5](#_Toc443054953)

[Bootstrap.pm 6](#_Toc443054954)

[DirAccess.pm 6](#_Toc443054955)

[DIRACCESS.pm 6](#_Toc443054956)

[Security 6](#_Toc443054957)

[LDAP.pm 6](#_Toc443054958)

[Display Logic 6](#_Toc443054959)

[View.pm 6](#_Toc443054960)

[Application Specific Libraries 6](#_Toc443054961)

[Web Presentation Standards 7](#_Toc443054962)

[Templates 7](#_Toc443054963)

[User Notifications 7](#_Toc443054964)

[Perl Notices Hash to HTML 7](#_Toc443054965)

[Form Fields and Dropdown Selectors 7](#_Toc443054966)

[Data Interfaces 8](#_Toc443054967)

[Database Access 8](#_Toc443054968)

[DBI Queries 8](#_Toc443054969)

[fetchrow\_array() - BHC Usual Pattern 8](#_Toc443054970)

[bind\_col 8](#_Toc443054971)

[Basic Column Bind Example 8](#_Toc443054972)

[Another Column Bind Example 9](#_Toc443054973)

[bind\_columns() 9](#_Toc443054974)

[Perl to Database Patterns 9](#_Toc443054975)

[Perl String to SQL Bit 9](#_Toc443054976)

[User Data Files 9](#_Toc443054977)

[Logging 10](#_Toc443054978)

[Apache Logs 10](#_Toc443054979)

[Security 10](#_Toc443054980)

[LDAP 10](#_Toc443054981)

[LDAP Testing 10](#_Toc443054982)

[Linux LDAP Search 10](#_Toc443054983)

[Custom Perl LDAP Search 10](#_Toc443054984)

[LDAP Perl Development 11](#_Toc443054985)

[LDAP Perl Use Case 11](#_Toc443054986)

[Perl Coding Style 11](#_Toc443054987)

[Sample Code 12](#_Toc443054988)

[Programming Tools 12](#_Toc443054989)

[Third Party Tools 12](#_Toc443054990)

[In-house Developed Tools 13](#_Toc443054991)

[Apache Logfile Analysis 13](#_Toc443054992)

[SQL Development 13](#_Toc443054993)

[SQL Server Management Studio 13](#_Toc443054994)

[Create SQL Export/Import Script of a Table 13](#_Toc443054995)

[Perl SQL Analysis 14](#_Toc443054996)

[Code Clean-up and Refactor Tasks 14](#_Toc443054997)

[Source Files Use CR Line Endings 15](#_Toc443054998)

[Perltidy 15](#_Toc443054999)

[Use the Standard lib/BHC Perl Libraries 16](#_Toc443055000)

[Eradicate Runtime Errors 16](#_Toc443055001)

[Separate User Data from Source Code Directories 16](#_Toc443055002)

[Code Debugging 16](#_Toc443055003)

[Use of IE F12 Developer Tools 16](#_Toc443055004)

[Code Testing 16](#_Toc443055005)

[Unit Testing on Command Line 16](#_Toc443055006)

[Code Deployment 17](#_Toc443055007)

[Incremental File Deployment During Development 17](#_Toc443055008)

[Package Deployment 18](#_Toc443055009)

[Create a Deployment Package 18](#_Toc443055010)

[Determining SVN Revisions for Deployment Package 18](#_Toc443055011)

[Verify the Deployment 20](#_Toc443055012)

[Tar the Deployment Directory 21](#_Toc443055013)

[Untar the Deployment Directory 21](#_Toc443055014)

[Documentation 22](#_Toc443055015)

[Document Locations 22](#_Toc443055016)

[Types of Documents 22](#_Toc443055017)

[System and Application Overview 22](#_Toc443055018)

[Entity Relationship Diagram (ERD) 22](#_Toc443055019)

[Code Comments 22](#_Toc443055020)

[Outline of Algorithmic Logic 22](#_Toc443055021)

[Authoring Documents 23](#_Toc443055022)

[Appendix 24](#_Toc443055023)

[Document Revisions 24](#_Toc443055024)

[Recommended Reading 24](#_Toc443055025)

[Perl Programming 24](#_Toc443055026)

[Object Oriented Programming 24](#_Toc443055027)

[Sample Bash Scripts 24](#_Toc443055028)

[Bash Startup Scripts 24](#_Toc443055029)

[.bashrc 24](#_Toc443055030)

[.bash\_aliases 25](#_Toc443055031)

[.bash\_aliases\_local 25](#_Toc443055032)

[CPAN Configuration 25](#_Toc443055033)

[Perlinfo.cgi Output 27](#_Toc443055034)

[Development Environment 27](#_Toc443055035)

[Production Environment 28](#_Toc443055036)

# Introduction and Objective

This document explains the best practices for the Perl software development at the Blackhills Corporation (BHC).

The objective is to assist in getting multiple software developers to agree on a common development methodology.

This is a shared document, with multiple authors. Refer to the appendix document change log for a change history.

# References

Also refer to:

* *Operational Support Perl Applications Analysis* document
* *Subversion Methodology* document

# Server Environment

## Apache Webserver Configuration

The webserver uses a standard Apache 2.0 configuration, including modperl.

The modperl module is configured with the file:

* /etc/httpd/conf.d/perl.conf

PerlPostConfigRequire /var/www/dakossdev/perl-bin/startup.pl

SetEnv PERL5LIB /var/www/dakossdev/cgi-bin/lib

PerlSwitches -I/var/www/dakossdev/cgi-bin/lib

The modperl startup.pl file allows for programmatic configuration and sets the CGI application common library path (like the PERL5LIB environment variable for command line testing) (but that doesn’t work; but SetEnv or PerlSwitches does!).

## CGI Application Source Code Organization

/var/www/dakossdev/

|  |  |
| --- | --- |
| cgi-bin | application CGI code |
| perl-bin | Apache-level modules (only startup.pl is used?) |
| test | framework and application tests |
| www | application templates |

# Perl Libraries

Library usage details are included in each module’s file header.

## Development Libraries

Refer to the *Bash Script Appendix* sections for details on use of the Perl development libraries.

### System Auxiliary CPAN Library

A Perl system library, that contains CPAN modules, resides in the following directory:

* /opt/bhc/perl/lib/perl5/

This directory is in addition (auxiliary) to the formal Perl system libraries (perl –V):

* /usr/local/lib64/perl5
* /usr/local/share/perl5
* /usr/lib64/perl5/vendor\_perl
* /usr/share/perl5/vendor\_perl
* /usr/lib64/perl5
* /usr/share/perl5

Developers are free to install CPAN modules in the auxiliary library. Some example auxiliary modules include:

* CPAN
* ExtUtils
* JSON
* Parse
* Perl
* Test
* XML
* YAML

### Custom BHC Developer Library

A Perl library exists to contain custom BHC developer modules, that are used for software development:

* apps-perl/cgi-bin/branches/joe-dev/lib

## CGI Common Library

The common company Perl libraries, accessible by the CGI programs, are located in the following directory:

* /var/www/dakossdev/cgi-bin/lib/

Of course, the library is maintained in the repository:

* apps-perl/cgi-bin/branches/joe-dev/cgi-bin/lib

The CGI application modules do not need to include a *use lib* statement, since the Apache perl.conf file does it. Then for example, company modules are simply included with:

use BHC::Security::Ldap;

### Develop and Test With Various Libraries

Yet, new Perl library code (and application modules) can then be developed and tested, on the command line, with:

$ export PERL5LIB=/home/joe-dev/dev-dir/lib

$ perl ./my-test-program.pl

Or, use the current (development/production) library code:

$ export PERL5LIB=/var/www/dakossdev/cgi-bin/lib

$ perl ./my-test-program.pl

Refer to the .bash\_alias\_local settings in the *Appendix* sections for a better implementation of export variables.

#### Bootstrap.pm

use BHC/Bootstrap.pm;

This package contains essential data and functions for all BHC code. The Perl package defines constants, which can include application administrator names and user groups.

#### DirAccess.pm

use BHC/DirAccess.pm;

This package contains a library of authentication and authorization functions. The functions may be replaced with more robust functions, which reside in the lib/BHC subdirectories.

This package replaces a similar package named DIRACCESS.pm.

#### DIRACCESS.pm

This uppercased version of the package is original, legacy, obsolete, and buggy. Do not use this package. In fact, there are several copies of the same package in various Perl path locations… good gravy, seriously?!

The problem with the uppercased version of module is that, in the past, someone had the brilliance to install several differing versions of the file (in /usr/perl and /usr/local/perl and cgi-bin), with differing versions between the development and production environments and the repository! That is a worst practice. There is now a single, maintained version: cgi-bin/lib/BHC/DirAccess.pm.

### Security

#### LDAP.pm

The LDAP library module contains LDAP authentication functions. Refer to the *Security* section below for more details.

### Display Logic

#### View.pm

## Application Specific Libraries

Frequently, there is common code that applies to a specific application. In that case, a library module should be created that resides in the application directory.

For example, a Perl function library that is dedicated to the *Pole Inventory* application:

cgi-bin/bhp/PolesSw/LibPoleInventory.pm

package LibPoleInventory;

sub queryPoleInventoryDivisions() - a query is pretty much the same as get

sub getCustomerNames() - could be named queryCustomerNames()

sub getDatabaseConnectionPoles() - should be refactored to a common library, singleton method

sub getDatabaseConnectionGis() - should be refactored to a common library, singleton method

cgi-bin/bhp/Transformers/LibTransformers.pm

# Web Presentation Standards

## Templates

Use Text::Template;

## User Notifications

### Perl Notices Hash to HTML

The Perl code maintains a hash of user notices.

package BHC::Web::View;

push @{$userNotices{"info.data"}}, "Connecting to database: SQL\_DATA table: PoleInventory";

push @{$userNotices{"error.data"}}, "Query returned no matches.";

$templateVars{"tmplNoticeRows"} = formatUserNoticesToHtml( \%userNotices );

The user notices are then displayed to the user via the Text::Template code:

<hr/>

Application Notices

<table border="1" width="100%">

<tr>

<td align="center"><strong>Type</strong></center></td>

<td align="center"><strong>Description</strong></center></td>

</tr>

{

for( $j=0; $j<@tmplNoticeRows; $j++ ){

$OUT .= "$tmplNoticeRows[$j]";

}

}

</table>

## Form Fields and Dropdown Selectors

Handling user menu selections can be coded in a fairly simple manner using a combination of HTML, Javascript, and Perl.

A good example of passing form field values and menu dropdown selections can be found in the *Transformer Management System* (TMS) *user’s transformer search* HTML frame code:

* cgi-bin/bhp/Transformers/transnav.cgi
* www/bhp/Transformers/transnav.tmp

The TMS search implementation is a refactor of *how-not-to-code* an over-complication of a user interface. Yet, it continues to use:

* HTML frames (see menu.tmp for frame layout),
* HTML forms and (basic) CSS styling,
* Multiple HTML forms that call actions to different Perl scripts,
* Naming of HTML forms for reference by Javascript (rather than forms[i] references),
* Perl Text::Template variables to control HTML presentation that is based on user authorization (Perl) code,

if ( eval { LibTransformers::isAuthorizedToModule( { "authToSystem" => $authToSystem } ) } ) {

$templateVars{"menuAuth"} = $authToSystem;

* Javascript on\_load() method to set default HTML form values,
* Javascript to set DOM values on calls to the HTML form onSubmit() method,
* Javascript to perform the form submit() method,
* Passing of HTML form values, as set by the user, back to Perl on form submission, and
* Use of conditional Perl code that calls a method based on a variable passed from an HTML form value

if ( $method eq "saveUserDefaultDivision" )

# Data Interfaces

## Database Access

Also refer to the *SQL Development* section.

### DBI Queries

my $query = "SELECT column1, column2

FROM table

WHERE <condition>

ORDER BY <column>";

my $handle = $dbhGis->prepare($query) or die "Failed query: $!";

if ( ! defined $handle->execute() ) {

# returns undef on error, otherwise, rows affected on non-select (caveats), or true for selects

return ( -1, "Failed query in functionName(): $handle->errstr", undef);

}

$num = $sth->{NUM\_OF\_FIELDS}; # for selects

#### fetchrow\_array() - BHC Usual Pattern

while ( my ($column1, $column2) = $handle->fetchrow\_array() ) {

# process query data

}

#### bind\_col

(<http://search.cpan.org/~timb/DBI/DBI.pm#bind_col>)

##### Basic Column Bind Example

my $rcDbi; # DBI return codes

my $jobTicketIndex;

$rcDbi = $sth->execute(); # undef on error

die "Failed query for next job ticket ID" if ( ! defined $rcDbi );

$rcDbi = $sth->bind\_columns( \$jobTicketIndex );

die "Failed to bind columns for next job ticket ID" if ( ! defined $rcDbi );

$rcDbi = $sth->fetch;

# no rows or error, set as first job ticket for the day, otherwise increment

( ! defined $rcDbi || ! defined $jobTicketIndex ) ? $jobTicketIndex = 1 : $jobTicketIndex++;

##### Another Column Bind Example

$dbh->{RaiseError} = 1; # do this, or check every call for errors

eval {

$sth = $dbh->prepare( q{ SELECT region, sales FROM sales\_by\_region } );

$sth->execute;

my ($region, $sales);

# Bind Perl variables to columns:

$rv = $sth->bind\_columns( \$region, \$sales );

# you can also use Perl's \(...) syntax (see perlref docs):

# $sth->bind\_columns( \($region, $sales) );

# Column binding is the most efficient way to fetch data

# same as $sth->fetchrow\_arrayref, no more rows returns undef

# or error also returns undef!!, so then check $sth->err or $@

while ( $sth->fetch ) {

print "$region: $sales\n";

}

};

if ($@) {

# $sth->err and $DBI::err will be true if error was from DBI

warn $@; # print the error

... # do whatever you need to deal with the error

}

#### bind\_columns()

This is a slick query into a hash. Refer to the <http://search.cpan.org/~timb/DBI/DBI.pm#bind_columns> documentation.

$sth->execute;

my %row;

$sth->bind\_columns( \( @row{ @{$sth->{NAME\_lc} } } ));

while ($sth->fetch) {

print "$row{region}: $row{sales}\n";

}

## Perl to Database Patterns

### Perl String to SQL Bit

## User Data Files

There are user files that get uploaded to the system, by the users, through the web interface. Unfortunately, those user files reside in subdirectories of the CGI source code… no, seriously, it’s true!

## Logging

Not much logging, other than Apache logs.

### Apache Logs

The apache logs are archived, on the Production server, in the directory:

* /opt/bhc/archive/log/httpd

# Security

## LDAP

A newer LDAP library is available:

* lib/BHC/Security/LDAP.pm

The *Transformers application* uses the LDAP authentication (and does not use the custom framework authentication).

### LDAP Testing

#### Linux LDAP Search

The ldapsearch Linux command is useful in querying the LDAP service. The default LDAP server and base is configured on the bhdcapwsd01 development server (in the /etc/openldap/ldap.conf file).

$ ldapsearch [-LLL] "(sn=sheffel)" description name givenName displayName

$ ldapsearch [-A] "(sn=sheffel)"

$ ldapsearch -LLL "(sn=sullivan)" displayName title description sAMAccountName manager memberOf

$ ldapsearch -LLL -b "OU=Production,OU=FileNet,OU=Groups,OU=BHCORP,DC=bhcorp,DC=ad" # many production FileNet cn's

$ ldapsearch -LLL -b "OU=Groups,OU=BHCORP,DC=bhcorp,DC=ad" "objectClass=group" dn objectClass description objectCategory # all groups?, exceeds query limit of 5000

$ ldapsearch "(cn=app\_xmr\_admin)"

#### Custom Perl LDAP Search

See script headers for more example usages.

* ldap-search-any.pl -> ../../dev/transformer/CR223042-ldap/test-ldap-search-any.pl

$ ldap-search-any.pl <searchBase> <searchFilter> <searchAttrs> [<showAttrs>]

$ ldap-search-any.pl # no arguments shows options and sample run

$ ldap-search-any.pl "" "(cn=app\_xmr\_admin)" # find group, show all attributes

* ldap-search-user.pl -> ../../dev/transformer/CR223042-ldap/test-ldap-search-user.pl

$ ldap-search-user.pl

$ search-userids-by-fullnames.sh

$ show-ldap-info-by-userid.sh

### LDAP Perl Development

cd $SVN\_BRANCH/dev/transformer/CR223042-ldap/

./test-ldap-user-memberships.pl jsheffel

./test-ldap-user-memberships.pl mschad

./test-ldap-check-memberships.pl jsheffel App\_VC\_Ubuntu APP\_SVN\_Perl some\_bogus\_group App\_SVN\_CLick

./test-ldap-check-memberships.pl mschad App\_VC\_Ubuntu APP\_SVN\_Perl some\_bogus\_group App\_SVN\_CLick

./test-ldap-check-memberships.pl jsheffel App\_poweron\_admins App\_poweron\_switch\_users App\_poweron\_users App\_poweron\_webcall

./test-ldap-check-memberships.pl mschad App\_poweron\_admins App\_poweron\_switch\_users App\_poweron\_users App\_poweron\_webcall

### LDAP Perl Use Case

The Transformer Management System makes good use of the LDAP library, as the following code excerpt shows:

# DeleteTrans.cgi – module that deletes a transformer, so requires high privileges

use BHC::Bootstrap;

use LibTransformers;

my %scriptEnv = BHC::Bootstrap::getScriptEnv();

my $userId = $ENV{'REMOTE\_USER'} // "UnknownUser";

my ($errorText, $authToSystem, $authToDivision, $arrayRefMemberships, $hashRefNotices) = LibTransformers::getAuthorizations( {"accountName" => $userId} );

if ( eval { LibTransformers::isAuthorizedToModule( { "authToSystem" => $authToSystem } ) } ) {

&main();

} else {

# user unauthorized

# LibTransformers.pm

package LibTransformers;

use BHC::Security::Ldap;

sub getAuthorizations {

my %paramHash = %{ shift @\_ };

my $accountName = $paramHash{accountName};

my ($errorText, $arrayRefMemberships, $hashRefNotices) = ldapCheckMemberships( {"accountName" => $accountName, "checkGroups" => \@ldapGroupsXmr} );

# Perl Coding Style

The following code styles are recommended:

1. File and function comment headers
2. Camel-case variable names that describe their use (hence self-commenting)
3. Code comments to reveal the intent of an esoteric line or obscure block; code should be self-commenting
4. Indentation using 4 spaces, not tabs; code editors can be configured for this
5. Control constructs separated by single spaces; e.g. if ( ! defined $condition ) {
6. Function parameters and array/hash definitions not surrounded by spaces, but comma/space separated

## Sample Code

The following is a sample Perl library module.

#!/usr/bin/perl -w

# Common library routines for specific application.

#

# NOTES:

# ) Refer to \_\_END\_\_ for additional notes.

package LibPoleInventory;

# ------------------------------------------------------------------------------------------------------

# function queryPoleInventoryDivisions( $dbhPoleInventory )

# Get division ID to city and state relations; see \_\_END\_\_ notes

# Database Table: [PoleInventory].[linuxrpt].[Divisions]

# DivId Division State

# 1 RapidCity SD

#

# inputs:

# $dbhPoleInventory - database handle to PoleInventory database

#

# returns:

# $rc - return code: -1 = empty table, 0 = success, 1 = recoverable error

# $errorMessage - text error message (on errors)

# \%tableHash - pointer to hash of arrays: {$DivId} = [$divisionState, $divisionName]

# -------------------------------------------------------------------------------------------------------

sub queryPoleInventoryDivisions {

my $dbhPoleInventory = shift;

my %tableHash = ();

my $errorMessage = undef;

if ( ! defined $dbhPoleInventory ) {

$errorMessage = "Bad call to queryPoleInventoryDivisions(), requires database handle parameter";

return (-1, $errorMessage, undef);

}

# query divisions to build return hash [logic]

my $query = "SELECT DivId, State, Division FROM linuxrpt.Divisions";

my $handle = $dbhPoleInventory->prepare($query) or die "Failed query: $!";

if ( ! defined $handle->execute() ) {

return (-1, "Failed query in queryPoleInventoryDivisions(): " . $handle->errstr, undef);

}

# more code here

}

# Programming Tools

The Development (bhdcapwsd01) and V10004 servers host several useful (Linux) programming tools.

## Third Party Tools

* Subversion
* /opt/bhc/bin/cloc (v10004:/usr/local/bin/cloc -> /opt/cloc/cloc-1.62/cloc)
* /opt/bhc/bin/perltidy (v10004:/usr/local/bin/perltidy -> /opt/perltidy/Perl-Tidy-20140711/perltidy)

Also, there are some tools on the production server:

* /opt/bhc/bin

## In-house Developed Tools

Refer to the $SVN\_BRANCH/opt/bin directory that contains custom in-house scripts.

## Apache Logfile Analysis

To analyze the *Job Tickets system* page hits for the last 2 months:

bhdcapwsp01$ cd /opt/bhc/archive/log/httpd/

bhdcapwsp01$ cat ssl\_access\_log-201510?? ssl\_access\_log-201511?? \

| grep /wos/ \

| format-ssl-access-log.pl

--filter401 \ # filter out 401 unauthorized, ie. password prompting

--filterNonCgi \ # filter out file uploads, etc.

--stripParams \ # strip off HTTP request <uri>?<parameters>

| sort | uniq -c | sort -rn

## SQL Development

### SQL Server Management Studio

#### Create SQL Export/Import Script of a Table

http://stackoverflow.com/questions/2321052/get-script-of-sql-server-data

You can generate a script that will build whichever tables you wish from a database as well as insert the data in those tables (as far as I know you have to export all of the data in the selected tables however).

To do this follow these steps:

1. Right-click on your database and select Tasks > Generate Scripts
2. In the Generate and Publish Scripts wizard, select the "Select specific database objects" option
3. Expand the "Tables" tree and select all of the tables you wish to export the scheme and data for, then click Next
4. In the next screen choose how you wish to save the script (the Output Type must remain set as "Save scripts to a specific location"), then
   1. Click the Advanced button in the top right corner
5. In the newly opened window, under the General section is a setting called "Types of data to script", set this to "Scheme and data" and click OK
6. Click Next, review the export summary and click Next again. This will generate the script to your selected destination.

To restore your database, simply create a new database and change the first line of your generated script to USE [Your.New.Database.Name], then execute.

Your new database will now have all of the tables and data you selected from the original database.

### Perl SQL Analysis

An in-house developed Perl source code static analyzer is available, that shows which database tables are queried and the individual queries by module filename and line number. The analyzer is very useful for determining which tables a system or application queries, or which table columns are accessed.

The custom analyzer is somewhat restricted, in that it may rely on certain source code patterns to parse the SQL code. For example, extracting the MSSQL database names relies on each (Perl) module source code to call the database connection using the string "sql\_data", as in: db\_connect( "SQL\_DATA", "Job\_Ticket\_RC" ). Source code for Oracle database interfaces are not analyzed (even though the SQL query statement will be parsed and included in the output).

First, create or update the system’s database list:

$ cd $SVN\_BRANCH/dev/analysis/perl/perl-sql

$ ./create-system-database-list.sh wos # create or update a system's database list

$ cat /opt/bhc/analysis/perl/sql/system/wos/databases.txt

Then, create or update the table list:

# Reads the database.txt file, as created above, and queries for a list of the database

# This command can be executed from any directory.

./list-system-database-tables.pl wos > /opt/bhc/analysis/perl/sql/system/wos/tables.txt

Then, run a system's SQL analysis:

$ cd $SVN\_BRANCH/cgi-bin/bhp/wos/

wos$ ls \*.cgi | analyze-perl-sql.pl --readFileList --tableInfoFile /opt/bhc/analysis/perl/sql/system/wos/tables.txt

> /opt/bhc/analysis/perl/sql/system/wos/sql-analysis-wos-151103-01.out

2>/opt/bhc/analysis/perl/sql/system/wos/sql-analysis-wos-warnings-151103-01.out

An example of the output is:

amibin.cgi:143:job\_ticket\_rc:linuxrpt.customers: "SELECT [Last], [First], [Service City] FROM linuxrpt.Customers WHERE [Customer ID] = $acctNumber"

woslist.cgi:601:job\_ticket\_rc:linuxrpt.invoices: "SELECT jtnum, marnum FROM linuxrpt.invoices WHERE jtnum= '$callID' ORDER BY jtnum"

woslist.cgi:627:job\_ticket\_rc:linuxrpt.work\_order: "SELECT status FROM linuxrpt.Work\_Order WHERE Call\_Id= '$callID'"

The output shows: module name, line number, database name, table name, and query.

# Code Clean-up and Refactor Tasks

Also refer to the *Perl Applications Analysis* document for additional (duplicate) tasks.

## Source Files Use CR Line Endings

All files in the repository should have the CR end-of-line terminators; and not the DOS CRLF line terminations. If various developers edit files and commit files that have CRLF terminations, then maintaining the repository becomes very difficult (if not impossible, by rendering diffs and blames useless). Refer the *Subversion Methodology* document for proper configuration of the Subversion application before committing any repository files!

Code editor settings may also affect line endings. Notepad++ maintains the line endings that are already in a file. Therefore, the line endings for a properly formatted file do not need to be specified, but when a new file is created, then use the following Notepad++ action to convert to Unix CR line endings:

* Edit -> EOL Conversion -> UNIX Format

Using the Vim editor will signify the file type:

* "test/data/test-DOS-eol-file.txt" [dos] 26L, 1179C

Vim can convert the file by editing the file, issuing the set command, and saving the file:

: set ff=unix

On Linux, files containing DOS CRLF can be identified:

* $SVN\_BRANCH/cgi-bin/trunk/opt/bin/isDosFile.pl [--verbose] <filename>

To test a directory hierarchy of files:

$ for srcFile in $(find . \( -name "\*.cgi" -o -name "\*.pm" -o -name "\*.pl" -o -name "\*.txt" \) -a -type f -print); do

if [ $( opt/bin/isDosFile.pl $srcFile ) -eq 1 ]; then

echo DOS file: $srcFile;

fi;

done

Or:

$ find cgi-bin/ \( -name "\*.cgi" -o -name "\*.pm" -o -name "\*.pl" -o -name "\*.txt" \) -a -type f –exec isDosFile –verbose {} \;

To convert DOS files that are already committed to the repository, use the following steps on a Linux-based repository:

$ cd ~/code/repo/apps-perl/cgi-bin/branches/jsheffel-dev

$ for dosFile in $(find . -path '\*svn' -prune -o -type f -exec file {} \; | grep CRLF | awk -F: '{print $1}'); do

dos2unix $dosFile

done | tee ~/tmp/perl-repo-DOS-file-convert-150707-01.out 2>&1

$ svn status # should list the converted files

Follow the Subversion method for committing the modified files.

## Perltidy

For Perl source that has not yet been prettified, first process each file with the perltidy utility.

$ perltidy --backup-and-modify-in-place --maximum-line-length=120 <file.pl>

A backup (.bak) file of the original file will be created. Any errors will be reported (see .ERR file).

Unfortunately, the resulting tidied source code is not guaranteed to execute without error. So, more extensive testing may need to be performed.

## Use the Standard lib/BHC Perl Libraries

## Eradicate Runtime Errors

## Separate User Data from Source Code Directories

# Code Debugging

## Use of IE F12 Developer Tools

The Internet Explorer Developer Tools, which is activated from the browser using the F12 key, can be used for debugging CGI scripts. This is especially true when the BHP::Bootstrap method is called at the beginning of the CGI script. The Bootstrap method sends a cookie (or cookies) to the browser, which contains the (HTTP post and get) parameters that were sent to the CGI script when called. For example,

Key Value

Bhcparams1 address0="1515 Wynkoop St." certDate0=03/01/2021 comments="160204 jsheffel Add test entry" cylinderDrop=RC004 cylinderId0=RC004 district=All districtDrop=RapidCity empId0=jsheffel method=save ouncesAfterCheck0=1 ouncesBeforeCheck0=1 recoveryDate0=02/04/2016 refrigType0=R-407C source0="Denver Office Test" stateDrop=SD techDrop=ddejong transDate0= transDate\_246= transDate\_247= transDate\_526= transDate\_572= transDate\_9= transTo0= wgtAfter0=1000 wgtBefore0=978

The cookie is only set for developers that are defined in the Bootstrap.pm module, and not for any other users (as a security feature).

This cookie of parameters can then simply be pasted to a script command line call, for interactively debugging Perl:

$ SERVER\_NAME=bhdcapwsd01 REMOTE\_USER=jsheffel perl -d RecoveryLog.cgi address0="1515 Wynkoop St." certDate0=03/01/2021 comments="160204 jsheffel Add test entry" cylinderDrop=RC004 cylinderId0=RC004 district=All districtDrop=RapidCity empId0=jsheffel method=save ouncesAfterCheck0=1 ouncesBeforeCheck0=1 recoveryDate0=02/04/2016 refrigType0=R-407C source0="Denver Office Test" stateDrop=SD techDrop=ddejong transDate0= transDate\_246= transDate\_247= transDate\_526= transDate\_572= transDate\_9= transTo0= wgtAfter0=1000 wgtBefore0=978

Refer to the *Code Testing* section for additional details on command line usage.

# Code Testing

## Unit Testing on Command Line

Unit testing can be performed on the (Linux) command line.

export PERL5LIB=$SVN\_BRANCH/cgi-bin/lib # use local Perl library; refer to .bash\_aliases\_local appendix

# unit testing by calling cgi file and redirecting output and timed results

time \

SERVER\_NAME=bhdcapwsd01.bhcorp.ad REMOTE\_USER=jsheffel \

perl appModule.cgi paramA=1 paramB= paramC=”some text” > module.out 2> module.err

echo RESULT: $?

A good example of a scripted unit test suite can be found at:

* $SVN\_BRANCH/test/system/Transformers/run-test-suite-transformers.sh

Calling a module without command line parameters (which equates to no query or post parameters), only tests a module’s default program path. A more elaborate calling scheme would need to be developed to test various script flows of control (by calling with command line parameters as script params, as shown above).

# Code Deployment

## Incremental File Deployment During Development

$ deploy-repo-files.pl –-help

$ svn status

M cgi-bin/bhe/Refrigerant/DisburseLog.cgi

? cgi-bin/bhe/Refrigerant/archive

M cgi-bin/bhp/ElectDc/ClosedBin.cgi

M cgi-bin/bhp/ElectDc/OpenBin.cgi

? cgi-bin/bhp/Transformers/TODO

M opt/bin/isDosFile.pl

M www/bhp/ElectDc/DcApp.tmp

M www/bhp/ElectDc/EstLetter.tmp

M www/bhp/wos/job\_ticket\_print.tmp

M www/bhp/wos/jobticket.tmp

Deploy modified files of interest from the top-level SVN directory:

$ deploy-repo-files.pl --interactive www/bhp/ElectDc/

Deploy: www/bhp/ElectDc/DcApp.tmp -> www/bhp/ElectDc/DcApp.tmp [y|n]: y

Copied: www/bhp/ElectDc/DcApp.tmp to /var/www/dakossdev/www/bhp/ElectDc

-rw-rw-r--. 1 mrsadmin dakossadmin 13401 Jan 29 10:08 /var/www/dakossdev/www/bhp/ElectDc/DcApp.tmp

Deploy: www/bhp/ElectDc/EstLetter.tmp -> www/bhp/ElectDc/EstLetter.tmp [y|n]: y

Copied: www/bhp/ElectDc/EstLetter.tmp to /var/www/dakossdev/www/bhp/ElectDc

-rw-rw-r--. 1 mrsadmin dakossadmin 5379 Jan 29 10:08 /var/www/dakossdev/www/bhp/ElectDc/EstLetter.tmp

Or, descend into lower subdirectories to deploy all modified files in that subdirectory:

$ cd www/bhp/wos/

$ deploy-repo-files.pl

Copied: jobticket.tmp to /var/www/dakossdev/www/bhp/wos

-rw-rw-r--. 1 mrsadmin dakossadmin 3519 Jan 29 10:12 /var/www/dakossdev/www/bhp/wos/jobticket.tmp

Copied: job\_ticket\_print.tmp to /var/www/dakossdev/www/bhp/wos

-rw-rw-r--. 1 mrsadmin dakossadmin 3710 Jan 29 10:12 /var/www/dakossdev/www/bhp/wos/job\_ticket\_print.tmp

Now, incremental tests or integration tests can be executed.

## Package Deployment

A command line deployment methodology exists.

Typically, a package is created for the deployment of a set of files that are associated with a change request (e.g. CR123456). After the file updates are tested in the Development environment, a package is created. The package deployment can then be tested in the Development environment (even though the updated files may already reside in the deployment target directory, due to recent development and testing). Then, the deployment package is copied to the Production server, and deployed per the scheduled change request plan.

### Create a Deployment Package

$ svn commit -m "jobticket: correct phone number (CR290039)" www/bhp/wos/

Authentication realm: <https://svnweb:443> BHE Subversion Server - Apps (Perl)

Password for 'jsheffel': \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Sending www/bhp/wos/job\_ticket\_print.tmp

Sending www/bhp/wos/jobticket.tmp

Transmitting file data ..

Committed revision 306.

[jsheffel@bhdcapwsd01 jsheffel-dev]$ svn commit -m "edca: update bank option text; fix warnings; (CR290039)" cgi-bin/bhp/ElectDc/ www/bhp/ElectDc/

Authentication realm: <https://svnweb:443> BHE Subversion Server - Apps (Perl)

Password for 'jsheffel': \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Sending cgi-bin/bhp/ElectDc/ClosedBin.cgi

Sending cgi-bin/bhp/ElectDc/OpenBin.cgi

Sending www/bhp/ElectDc/DcApp.tmp

Sending www/bhp/ElectDc/EstLetter.tmp

Transmitting file data ....

Committed revision 307.

[jsheffel@bhdcapwsd01 jsheffel-dev]$ svn update

Updating '.':

Authentication realm: <https://svnweb:443> BHE Subversion Server - Apps (Perl)

Password for 'jsheffel': \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

At revision 307.

### Determining SVN Revisions for Deployment Package

[jsheffel@bhdcapwsd01 jsheffel-dev]$ svn log --limit 7 --verbose

Authentication realm: <https://svnweb:443> BHE Subversion Server - Apps (Perl)

Password for 'jsheffel': \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

------------------------------------------------------------------------

r307 | jsheffel | 2016-01-29 10:57:51 -0600 (Fri, 29 Jan 2016) | 1 line

Changed paths:

M /branches/jsheffel-dev/cgi-bin/bhp/ElectDc/ClosedBin.cgi

M /branches/jsheffel-dev/cgi-bin/bhp/ElectDc/OpenBin.cgi

M /branches/jsheffel-dev/www/bhp/ElectDc/DcApp.tmp

M /branches/jsheffel-dev/www/bhp/ElectDc/EstLetter.tmp

edca: update bank option text; fix warnings; (CR290039)

------------------------------------------------------------------------

r306 | jsheffel | 2016-01-29 10:56:44 -0600 (Fri, 29 Jan 2016) | 1 line

Changed paths:

M /branches/jsheffel-dev/www/bhp/wos/job\_ticket\_print.tmp

M /branches/jsheffel-dev/www/bhp/wos/jobticket.tmp

jobticket: correct phone number (CR290039)

------------------------------------------------------------------------

r303 | jsheffel | 2016-01-28 13:29:00 -0600 (Thu, 28 Jan 2016) | 1 line

Changed paths:

M /branches/jsheffel-dev/www/bhp/wos/DisburseLog.tmp

M /branches/jsheffel-dev/www/bhp/wos/EditAsap.tmp

…

M /branches/jsheffel-dev/www/bhp/wos/left.tmp

A /branches/jsheffel-dev/www/graphics/logo-webpage-BHE.jpg

jobticket: use new BHE logo for webpages (CR290039)

------------------------------------------------------------------------

r302 | jsheffel | 2016-01-28 13:22:16 -0600 (Thu, 28 Jan 2016) | 1 line

Changed paths:

M /branches/jsheffel-dev/cgi-bin/bhp/ElectDc/searchaddress.cgi

…

M /branches/jsheffel-dev/www/bhp/ElectDc/nonCustomerSave.tmp

edca: rebrand text references to BHE (CR290039)

------------------------------------------------------------------------

r301 | jsheffel | 2016-01-28 13:21:22 -0600 (Thu, 28 Jan 2016) | 1 line

Changed paths:

M /branches/jsheffel-dev/cgi-bin/bhp/wos/TransTo.cgi

…

M /branches/jsheffel-dev/www/bhp/wos/woslist.tmp

jobticket: rebrand text references to BHE (CR290039)

------------------------------------------------------------------------

r300 | jsheffel | 2016-01-26 16:59:12 -0600 (Tue, 26 Jan 2016) | 1 line

Changed paths:

D /branches/jsheffel-dev/cgi-bin/bhp/PolesSw/CoLinePatrol.pl

…

D /branches/jsheffel-dev/cgi-bin/bhp/PolesSw/SearchLp\_moved.cgi

poles: remove obsolete CGI (CR288893)

------------------------------------------------------------------------

$ create-deployment.pl --revision 301:303 --revision 306:307

SVN password:

Enter short one-word deployment description (eg. jobtickets-refactor-CR123456): jobtickets-edca-CR290039

INFO: SVN auth file: /home/jsheffel/.subversion/auth/svn.simple/c461ff6e89bec0681fe28f418ff3feb1

INFO: Creating deployment directory: /opt/bhc/deploy/apps-perl/deployment/jobtickets-edca-CR290039-160129\_1108

INFO: Adding deployment source files to directory: /opt/bhc/deploy/apps-perl/deployment/jobtickets-edca-CR290039-160129\_1108/files

A /opt/bhc/deploy/apps-perl/deployment/jobtickets-edca-CR290039-160129\_1108/files/www/bhp/wos/GetInvoice.tmp

Export complete.

…

A /opt/bhc/deploy/apps-perl/deployment/jobtickets-edca-CR290039-160129\_1108/files/www/bhp/wos/EditDist.tmp

Export complete.

INFO: Writing deploy commands to: /opt/bhc/deploy/apps-perl/deployment/jobtickets-edca-CR290039-160129\_1108/run-deploy.sh

INFO: Creating manifest file: /opt/bhc/deploy/apps-perl/deployment/jobtickets-edca-CR290039-160129\_1108/manifest.txt

INFO: Writing deploy verify commands to: /opt/bhc/deploy/apps-perl/deployment/jobtickets-edca-CR290039-160129\_1108/run-deploy-verify.sh

### Verify the Deployment

$ ./run-deploy-verify.sh --to-cgi-directory 2>&1 | tee --append result-run-deploy-verify-prod.out

Verifying deployment to development or production CGI directory!

Continue [y|n]: y

INFO: Verifying deployment to target directory: /var/www/dakossdev

INFO: Verifying against deployment source: /opt/bhc/deploy/apps-perl/deployment/jobtickets-edca-CR290039-160129\_1108/files

Continue [y|n]: y

INFO: Verifying modified file: /var/www/dakossdev/www/bhp/wos/GetInvoice.tmp

17392 19 /opt/bhc/deploy/apps-perl/deployment/jobtickets-edca-CR290039-160129\_1108/files/www/bhp/wos/GetInvoice.tmp

17392 19 /var/www/dakossdev/www/bhp/wos/GetInvoice.tmp

-rw-rw-r--. 1 mrsadmin dakossadmin 18572 Jan 29 11:53 /var/www/dakossdev/www/bhp/wos/GetInvoice.tmp

INFO: Verifying modified file: /var/www/dakossdev/www/bhp/wos/jobticket.tmp

$ $SVN\_BRANCH/dev/deploy/create-deployment.pl --revision <revisionRange> [--revision <revisionRange> …]

For example,

# create a deployment for SVN revisions 158-173 and 195

$ export SVN\_BRANCH=/home/jsheffel/code/repo/apps-perl/cgi-bin/branches/jsheffel-dev # use your own branch, see .bash\_aliases\_local documentation

$ $SVN\_BRANCH/dev/deploy/create-deployment.pl --revision 158:173 --revision 195

### Tar the Deployment Directory

bhdcapwsd01 deployment]$ tar cf ../tarfiles/jobtickets-edca-CR290039-160129\_1108.tar jobtickets-edca-CR290039-160129\_1108/

### Untar the Deployment Directory

bhdcapwsp01 deployment]$ tar xpvf ../tarfiles/jobtickets-edca-CR290039-160129\_1108.tar

jesadmin@bhdcapwsp01:/$ cd $DEPLOY

jesadmin@bhdcapwsp01:/opt/bhc/deploy/apps-perl/deployment$ cd jobtickets-edca-CR290039-160129\_1108

jesadmin@bhdcapwsp01:/opt/bhc/deploy/apps-perl/deployment/jobtickets-edca-CR290039-160129\_1108$ ll

total 136K

drwxrwxr-x. 4 jsheffel jsheffel 4.0K Jan 29 11:08 files

-rw-rw-r--. 1 jsheffel jsheffel 672 Jan 29 11:08 manifest.txt

-rw-rw-r--. 1 jsheffel jsheffel 152 Jan 29 11:25 README.txt

-rw-rw-rw-. 1 jsheffel jsheffel 37K Jan 29 11:34 result-run-deploy-dev.out

-rw-rw-rw-. 1 jsheffel jsheffel 0 Jan 29 11:08 result-run-deploy-prod.out

-rw-rw-rw-. 1 jsheffel jsheffel 23K Jan 29 11:35 result-run-deploy-verify-dev.out

-rw-rw-rw-. 1 jsheffel jsheffel 0 Jan 29 11:08 result-run-deploy-verify-prod.out

-rwxrwxr-x. 1 jsheffel jsheffel 38K Jan 29 11:22 run-deploy.sh

-rwxrwxr-x. 1 jsheffel jsheffel 19K Jan 29 11:23 run-deploy-verify.sh

jesadmin@bhdcapwsp01:/opt/bhc/deploy/apps-perl/deployment/jobtickets-edca-CR290039-160129\_1108$ ./run-deploy.sh --to-cgi-directory 2>&1 | tee --append result-run-deploy-prod.out

Deploying to actual system CGI directory!

Continue [y|n]: y

INFO: Using deployment source: /opt/bhc/deploy/apps-perl/deployment/jobtickets-edca-CR290039-160129\_1108/files

INFO: Planned deployment backup directory: /opt/bhc/deploy/apps-perl/backup/jobtickets-edca-CR290039-160129\_1153

INFO: Deploying to target directory: /var/www/dakossdev

Continue [y|n]: y

INFO: Creating deployment backup directory: /opt/bhc/deploy/apps-perl/backup/jobtickets-edca-CR290039-160129\_1153

INFO: Modifying file: /var/www/dakossdev/www/bhp/wos/GetInvoice.tmp

mkdir: created directory `/opt/bhc/deploy/apps-perl/backup/jobtickets-edca-CR290039-160129\_1153/www'

mkdir: created directory `/opt/bhc/deploy/apps-perl/backup/jobtickets-edca-CR290039-160129\_1153/www/bhp'

mkdir: created directory `/opt/bhc/deploy/apps-perl/backup/jobtickets-edca-CR290039-160129\_1153/www/bhp/wos'

`/var/www/dakossdev/www/bhp/wos/GetInvoice.tmp' -> `/opt/bhc/deploy/apps-perl/backup/jobtickets-edca-CR290039-160129\_1153/www/bhp/wos/GetInvoice.tmp'

`/opt/bhc/deploy/apps-perl/deployment/jobtickets-edca-CR290039-160129\_1108/files/www/bhp/wos/GetInvoice.tmp' -> `/var/www/dakossdev/www/bhp/wos/GetInvoice.tmp'

chown: changing group of `/var/www/dakossdev/www/bhp/wos/GetInvoice.tmp': Operation not permitted

chmod: changing permissions of `/var/www/dakossdev/www/bhp/wos/GetInvoice.tmp': Operation not permitted

INFO: Modifying file: /var/www/dakossdev/www/bhp/wos/jobticket.tmp

`/var/www/dakossdev/www/bhp/wos/jobticket.tmp' -> `/opt/bhc/deploy/apps-perl/backup/jobtickets-edca-CR290039-160129\_1153/www/bhp/wos/jobticket.tmp'

`/opt/bhc/deploy/apps-perl/deployment/jobtickets-edca-CR290039-160129\_1108/files/www/bhp/wos/jobticket.tmp' -> `/var/www/dakossdev/www/bhp/wos/jobticket.tmp'

chown: changing group of `/var/www/dakossdev/www/bhp/wos/jobticket.tmp': Operation not permitted

chmod: changing permissions of `/var/www/dakossdev/www/bhp/wos/jobticket.tmp': Operation not permitted

INFO: Modifying file: /var/www/dakossdev/www/bhp/wos/MonthlyJt.tmp

# Documentation

Documentation at Black Hills is mostly authored using Microsoft Word and Visio.

## Document Locations

A document management system does not exist. Documents are (typically) stored on a shared drive:

* N:\BHSC\Information\_Technology\WAM\
  + Software\_Development\
  + <application-folder>\

When someone is editing a shared (Word) document, then subsequent authors are locked from editing the file.

## Types of Documents

### System and Application Overview

### Entity Relationship Diagram (ERD)

### Code Comments

#### Outline of Algorithmic Logic

One way to document the logic of a program is to insert algorithm comments with a special string (like *[logic]*). For example:

# enter function view contract [logic]

# query all related transformers by contract number [logic]

Then, the special string can be grepped to reveal an outline of the algorithm, which can be communicated or added to documentation:

RefundContract]$ grep "\[logic\]" ElectExt.cgi | sed 's/ \[logic\]$//'

# enter function view contract

# query all related transformers by contract number

# query all nodes (also called CribNum in database) by all transformers

# if $getTransMeters, query node of master transformer

# query extension data from ElectExt table by contract number

# for each entry by contract numbe:

# found table record, set edit mode to Update (from Add default)

# calculate owner contribution ($MaxRefund - $Allowance)

# set all webpage template variables (only sensible to execute for a single record)

# use sales tax rate from record for contract

# if $changeTransMeters, query all of transformers by master node

# otherwise, query all of transformers by contract number

Refer to the Refund Contract code and system documentation for more complete example usage.

## Authoring Documents

Embedding (or linking) Visio diagrams into Word documents is preferred, so that any diagram updates are dynamically reflected in the Word document.

* <http://www.indezine.com/products/visio/visioword.html>
* <http://www.makeuseof.com/answers/edit-text-visio-diagrams-embedded-word-document/>

**But**, I can’t get it to work, and a web search returns nothing about this issue (150708, Jsheffel).

A document template exists:

* N:\BHSC\Information\_Technology\WAM\Software\_Development\new-document-template.docx

To use the template:

1. Copy the template to the new document location
2. Update the File -> Info
   1. Authors
3. Update the footer document location

# Appendix

## Document Revisions

|  |  |  |
| --- | --- | --- |
| Revision Date | Author | Description |
|  |  |  |
| 160212 | Jsheffel | Clarify custom Perl LDAP commands; add LDAP example code from Transformers; |
| 160129 | Jsheffel | Add content to *Code Deployment* section; add *Code Debug* section; |
| 151202 | Jsheffel | Add *Code Comments* and *Algorithm Logic* sections; |
| 151029 | Jsheffel | Add *Form Fields and Dropdown Selectors* section; |
| 151019 | Jsheffel | Add Perl system/developer library details; add bash script appendices; add CPAN appendix; add Recommended Reading appendix; |
| 151001 | Jsheffel | Add DBI column bind code examples; |
| 150812 | Jsheffel | Add LDAP security info; update lib/BHP info; |
| 150722 | Jsheffel | Add to Perl Library section; add modperl info; |
| 150708 | Jsheffel | Add Perlinfo.cgi section to appendix; add *Documentation* section; add *Programming Tools* section; |
| 150707 | Jsheffel | Add *Code Clean-up Tasks* section; add *Perl Coding Style* section; |
| 150702 | Jsheffel | Original version |

## Recommended Reading

### Perl Programming

1. Book: *Modern Perl*

### Object Oriented Programming

1. *Perl – Bless My Referents*: http://perl.com/pub/1999/09/refererents.html

## Sample Bash Scripts

### Bash Startup Scripts

#### .bashrc

The following statements should be in the .bashrc file (by default?):

if [ -f ~/.bash\_aliases ]; then

. ~/.bash\_aliases

fi

# enable programmable completion features (you don't need to enable

# this, if it's already enabled in /etc/bash.bashrc and /etc/profile

# sources /etc/bash.bashrc).

if [ -f /etc/bash\_completion ] && ! shopt -oq posix; then

. /etc/bash\_completion

fi

#### .bash\_aliases

The .bash\_aliases file contains bash customizations that are common to all of an individual developer’s (Linux) system logins. In other words, each developer should have a personal .bash\_aliases file that is distributed to all of the systems that the developer uses. Add the following statements at the bottom of the .bash\_aliases file:

…

set -o vi

export GPG\_TTY=`tty`

if [ -f ~/.bash\_aliases\_local ]; then

. ~/.bash\_aliases\_local

fi

#### .bash\_aliases\_local

The .bash\_aliases\_local file contains bash customizations that are unique to the (Linux) login for the specific server.

#

# .bash\_aliases\_local - for BHC bhdcapwsd01:/home/joe-user

echo "Running .bash\_aliases\_local"

export CGI=/var/www/dakossdev/cgi-bin

export SVN\_BRANCH=/home/joe-user/code/repo/apps-perl/cgi-bin/branches/joe-dev

export DEPLOY=/opt/bhc/deploy/apps-perl/deployment

export PERL5LIB=$SVN\_BRANCH/lib:$SVN\_BRANCH/cgi-bin/lib:/opt/bhc/perl/lib/perl5

alias cdcgi='cd $CGI'

alias pdcgi='pushd $CGI'

alias cdsb='cd $SVN\_BRANCH'

alias pdsb='pushd $SVN\_BRANCH'

alias cdd='cd $DEPLOY'

alias pdd='pushd $DEPLOY'

alias tailssl='tail -f /var/log/httpd/ssl\_access\_log /var/log/httpd/ssl\_error\_log'

export PATH="/opt/bhc/bin/:$PATH" # directory that devs can share programs

export PATH="$SVN\_BRANCH/opt/bin/:$PATH" # add svn repo bin

export LD\_LIBRARY\_PATH="/u01/app/oracle/product/11.1.0/client\_1/lib" # to run Perl DBD Oracle

export ORACLE\_HOME="/u01/app/oracle/product/11.1.0/client\_1" # to run Perl DBD Oracle

## CPAN Configuration

CPAN modules are easily installed, to the Perl system auxiliary library, using the command:

$ perl –MCPAN ‘shell’

The following user configuration should first be maintained (by first running the above command, then editing the file?):

joe-dev@bhdcapwsd01:~/.cpan/CPAN$ cat MyConfig.pm

$CPAN::Config = {

'applypatch' => q[],

'auto\_commit' => q[0],

'build\_cache' => q[100],

'build\_dir' => q[/home/joe-dev/.cpan/build],

'build\_dir\_reuse' => q[0],

'build\_requires\_install\_policy' => q[ask/yes],

'bzip2' => q[/usr/bin/bzip2],

'cache\_metadata' => q[1],

'check\_sigs' => q[0],

'colorize\_output' => q[0],

'commandnumber\_in\_prompt' => q[1],

'connect\_to\_internet\_ok' => q[1],

'cpan\_home' => q[/home/joe-dev/.cpan],

'curl' => q[/usr/bin/curl],

'ftp' => q[],

'ftp\_passive' => q[1],

'ftp\_proxy' => q[],

'getcwd' => q[cwd],

'gpg' => q[/usr/bin/gpg],

'gzip' => q[/bin/gzip],

'halt\_on\_failure' => q[0],

'histfile' => q[/home/joe-dev/.cpan/histfile],

'histsize' => q[100],

'http\_proxy' => q[],

'inactivity\_timeout' => q[0],

'index\_expire' => q[1],

'inhibit\_startup\_message' => q[0],

'keep\_source\_where' => q[/home/joe-dev/.cpan/sources],

'load\_module\_verbosity' => q[v],

'lynx' => q[],

'make' => q[/usr/bin/make],

'make\_arg' => q[],

'make\_install\_arg' => q[],

'make\_install\_make\_command' => q[/usr/bin/make],

'makepl\_arg' => q[INSTALL\_BASE=/opt/bhc/perl],

'mbuild\_arg' => q[],

'mbuild\_install\_arg' => q[],

'mbuild\_install\_build\_command' => q[./Build],

'mbuildpl\_arg' => q[--installdirs site],

'ncftp' => q[],

'ncftpget' => q[],

'no\_proxy' => q[],

'pager' => q[/usr/bin/less],

'patch' => q[/usr/bin/patch],

'perl5lib\_verbosity' => q[v],

'prefer\_external\_tar' => q[1],

'prefer\_installer' => q[MB],

'prefs\_dir' => q[/home/joe-dev/.cpan/prefs],

'prerequisites\_policy' => q[ask],

'scan\_cache' => q[atstart],

'shell' => q[/bin/bash],

'show\_unparsable\_versions' => q[0],

'show\_upload\_date' => q[0],

'show\_zero\_versions' => q[0],

'tar' => q[/bin/tar],

'tar\_verbosity' => q[v],

'term\_is\_latin' => q[1],

'term\_ornaments' => q[1],

'test\_report' => q[0],

'trust\_test\_report\_history' => q[0],

'unzip' => q[/usr/bin/unzip],

'urllist' => [q[http://cpan.cse.msu.edu/], q[http://cpan.yimg.com/]],

'use\_sqlite' => q[0],

'version\_timeout' => q[15],

'wget' => q[/usr/bin/wget],

'yaml\_load\_code' => q[0],

'yaml\_module' => q[YAML],

};

1;

\_\_END\_\_

## Perlinfo.cgi Output

A perlinfo.cgi test script exists to show various global variable values; much like the phpinfo() method. The test script can be run from a web browser.

### Development Environment

Note, the sample run below is out-of-date.

https://bhdcapwsd01.bhcorp.ad/cgi-bin/test/perlinfo.cgi?debug=1&option=Loves%2520Energy

Perl hash - %ENV

$ENV{AUTHENTICATE\_SAMACCOUNTNAME} = jsheffel

$ENV{AUTH\_TYPE} = Basic

$ENV{DOCUMENT\_ROOT} = /var/www/dakossdev/www

$ENV{GATEWAY\_INTERFACE} = CGI/1.1

$ENV{HTTPS} = on

$ENV{HTTP\_ACCEPT} = text/html, application/xhtml+xml, \*/\*

$ENV{HTTP\_ACCEPT\_ENCODING} = gzip, deflate

$ENV{HTTP\_ACCEPT\_LANGUAGE} = en-US

$ENV{HTTP\_CONNECTION} = Keep-Alive

$ENV{HTTP\_COOKIE} = userid=jsheffel; fullname=Sheffel%2C%20Jeff; email=Jeff.Sheffel%40blackhillscorp.com; ldap=ldapadrc.bhcorp.ad%3A636; referring\_page=%2Fcgi-bin%2Findex.cgi

$ENV{HTTP\_HOST} = bhdcapwsd01.bhcorp.ad

$ENV{HTTP\_USER\_AGENT} = Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7.0; rv:11.0) like Gecko

$ENV{LD\_LIBRARY\_PATH} = /u01/app/oracle/product/11.1.0/client\_1/lib

$ENV{ORACLE\_HOME} = /u01/app/oracle/product/11.1.0/client\_1

$ENV{PATH} = /sbin:/usr/sbin:/bin:/usr/bin

$ENV{QUERY\_STRING} = debug=1&option=Loves%2520Energy

$ENV{REMOTE\_ADDR} = 10.10.46.173

$ENV{REMOTE\_PORT} = 57721

$ENV{REMOTE\_USER} = jsheffel

$ENV{REQUEST\_METHOD} = GET

$ENV{REQUEST\_URI} = /cgi-bin/test/perlinfo.cgi?debug=1&option=Loves%2520Energy

$ENV{SCRIPT\_FILENAME} = /var/www/dakossdev/cgi-bin/test/perlinfo.cgi

$ENV{SCRIPT\_NAME} = /cgi-bin/test/perlinfo.cgi

$ENV{SCRIPT\_URI} = https://bhdcapwsd01.bhcorp.ad/cgi-bin/test/perlinfo.cgi

$ENV{SCRIPT\_URL} = /cgi-bin/test/perlinfo.cgi

$ENV{SERVER\_ADDR} = 10.1.15.12

$ENV{SERVER\_ADMIN} = root@localhost

$ENV{SERVER\_NAME} = bhdcapwsd01.bhcorp.ad

$ENV{SERVER\_PORT} = 443

$ENV{SERVER\_PROTOCOL} = HTTP/1.1

$ENV{SERVER\_SIGNATURE} = Apache/2.2.15 (Red Hat) Server at bhdcapwsd01.bhcorp.ad Port 443

$ENV{SERVER\_SOFTWARE} = Apache/2.2.15 (Red Hat)

$ENV{SSL\_CIPHER} = ECDHE-RSA-AES256-SHA384

$ENV{SSL\_CIPHER\_ALGKEYSIZE} = 256

$ENV{SSL\_CIPHER\_EXPORT} = false

$ENV{SSL\_CIPHER\_USEKEYSIZE} = 256

$ENV{SSL\_CLIENT\_VERIFY} = NONE

$ENV{SSL\_COMPRESS\_METHOD} = NULL

$ENV{SSL\_PROTOCOL} = TLSv1.2

$ENV{SSL\_SECURE\_RENEG} = true

$ENV{SSL\_SERVER\_A\_KEY} = rsaEncryption

$ENV{SSL\_SERVER\_A\_SIG} = sha1WithRSAEncryption

$ENV{SSL\_SERVER\_I\_DN} = /DC=ad/DC=bhcorp/CN=bhdcaddcp01.bhcorp.ad

$ENV{SSL\_SERVER\_I\_DN\_CN} = bhdcaddcp01.bhcorp.ad

$ENV{SSL\_SERVER\_M\_SERIAL} = 126E0CF700000003EDCB

$ENV{SSL\_SERVER\_M\_VERSION} = 3

$ENV{SSL\_SERVER\_S\_DN} = /C=US/ST=South Dakota/L=Rapid City/O=Black Hills Corporation/OU=IT/CN=bhdcapwsp01.bhcorp.ad/emailAddress=webmaster@blackhillscorp.com

$ENV{SSL\_SERVER\_S\_DN\_C} = US

$ENV{SSL\_SERVER\_S\_DN\_CN} = bhdcapwsp01.bhcorp.ad

$ENV{SSL\_SERVER\_S\_DN\_Email} = webmaster@blackhillscorp.com

$ENV{SSL\_SERVER\_S\_DN\_L} = Rapid City

$ENV{SSL\_SERVER\_S\_DN\_O} = Black Hills Corporation

$ENV{SSL\_SERVER\_S\_DN\_OU} = IT

$ENV{SSL\_SERVER\_S\_DN\_ST} = South Dakota

$ENV{SSL\_SERVER\_V\_END} = Apr 20 15:32:52 2020 GMT

$ENV{SSL\_SERVER\_V\_START} = Apr 22 15:32:52 2015 GMT

$ENV{SSL\_SESSION\_ID} = 75F7CFF5935C8A225FA8559B196FA321FDCF47B30323A5CE3474B97F2F3B034E

$ENV{SSL\_TLS\_SNI} = bhdcapwsd01.bhcorp.ad

$ENV{SSL\_VERSION\_INTERFACE} = mod\_ssl/2.2.15

$ENV{SSL\_VERSION\_LIBRARY} = OpenSSL/1.0.1e-fips

Perl function - CGI::param()

param("debug") = 1

param("option") = Loves%20Energy

BHC function - DirAccess::directory\_structure()

Return Array = /var/www/dakossdev

Return Array = /cgi-bin

Return Array = /www

Return Array = /test

### Production Environment

Note, the sample run below is out-of-date.

https://dakoss.bhcorp.ad/cgi-bin/test/perlinfo.cgi?debug=1&myFlag=Loves%20Energy

Perl hash - %ENV

$ENV{AUTHENTICATE\_SAMACCOUNTNAME} = jsheffel

$ENV{AUTH\_TYPE} = Basic

$ENV{DOCUMENT\_ROOT} = /var/www/dakossdev/www

$ENV{GATEWAY\_INTERFACE} = CGI/1.1

$ENV{HTTPS} = on

$ENV{HTTP\_ACCEPT} = text/html, application/xhtml+xml, \*/\*

$ENV{HTTP\_ACCEPT\_ENCODING} = gzip, deflate

$ENV{HTTP\_ACCEPT\_LANGUAGE} = en-US

$ENV{HTTP\_CONNECTION} = Keep-Alive

$ENV{HTTP\_COOKIE} = userid=jsheffel; fullname=Sheffel%2C%20Jeff; email=Jeff.Sheffel%40blackhillscorp.com; ldap=ldapadrc.bhcorp.ad%3A636; referring\_page=%2Fcgi-bin%2Findex.cgi

$ENV{HTTP\_HOST} = dakoss.bhcorp.ad

$ENV{HTTP\_USER\_AGENT} = Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7.0; rv:11.0) like Gecko

$ENV{LD\_LIBRARY\_PATH} = /u01/app/oracle/product/11.1.0/client\_1/lib

$ENV{ORACLE\_HOME} = /u01/app/oracle/product/11.1.0/client\_1

$ENV{PATH} = /sbin:/usr/sbin:/bin:/usr/bin

$ENV{QUERY\_STRING} = debug=1&myFlag=Loves%20Energy

$ENV{REMOTE\_ADDR} = 10.10.46.173

$ENV{REMOTE\_PORT} = 57157

$ENV{REMOTE\_USER} = jsheffel

$ENV{REQUEST\_METHOD} = GET

$ENV{REQUEST\_URI} = /cgi-bin/test/perlinfo.cgi?debug=1&myFlag=Loves%20Energy

$ENV{SCRIPT\_FILENAME} = /var/www/dakossdev/cgi-bin/test/perlinfo.cgi

$ENV{SCRIPT\_NAME} = /cgi-bin/test/perlinfo.cgi

$ENV{SCRIPT\_URI} = https://dakoss.bhcorp.ad/cgi-bin/test/perlinfo.cgi

$ENV{SCRIPT\_URL} = /cgi-bin/test/perlinfo.cgi

$ENV{SERVER\_ADDR} = 10.1.8.201

$ENV{SERVER\_ADMIN} = root@localhost

$ENV{SERVER\_NAME} = dakoss.bhcorp.ad

$ENV{SERVER\_PORT} = 443

$ENV{SERVER\_PROTOCOL} = HTTP/1.1

$ENV{SERVER\_SIGNATURE} = Apache/2.2.15 (Red Hat) Server at dakoss.bhcorp.ad Port 443

$ENV{SERVER\_SOFTWARE} = Apache/2.2.15 (Red Hat)

$ENV{SSL\_CIPHER} = ECDHE-RSA-AES256-SHA384

$ENV{SSL\_CIPHER\_ALGKEYSIZE} = 256

$ENV{SSL\_CIPHER\_EXPORT} = false

$ENV{SSL\_CIPHER\_USEKEYSIZE} = 256

$ENV{SSL\_CLIENT\_VERIFY} = NONE

$ENV{SSL\_COMPRESS\_METHOD} = NULL

$ENV{SSL\_PROTOCOL} = TLSv1.2

$ENV{SSL\_SECURE\_RENEG} = true

$ENV{SSL\_SERVER\_A\_KEY} = rsaEncryption

$ENV{SSL\_SERVER\_A\_SIG} = sha1WithRSAEncryption

$ENV{SSL\_SERVER\_I\_DN} = /DC=ad/DC=bhcorp/CN=bhdcaddcp01.bhcorp.ad

$ENV{SSL\_SERVER\_I\_DN\_CN} = bhdcaddcp01.bhcorp.ad

$ENV{SSL\_SERVER\_M\_SERIAL} = 126E0CF700000003EDCB

$ENV{SSL\_SERVER\_M\_VERSION} = 3

$ENV{SSL\_SERVER\_S\_DN} = /C=US/ST=South Dakota/L=Rapid City/O=Black Hills Corporation/OU=IT/CN=bhdcapwsp01.bhcorp.ad/emailAddress=webmaster@blackhillscorp.com

$ENV{SSL\_SERVER\_S\_DN\_C} = US

$ENV{SSL\_SERVER\_S\_DN\_CN} = bhdcapwsp01.bhcorp.ad

$ENV{SSL\_SERVER\_S\_DN\_Email} = webmaster@blackhillscorp.com

$ENV{SSL\_SERVER\_S\_DN\_L} = Rapid City

$ENV{SSL\_SERVER\_S\_DN\_O} = Black Hills Corporation

$ENV{SSL\_SERVER\_S\_DN\_OU} = IT

$ENV{SSL\_SERVER\_S\_DN\_ST} = South Dakota

$ENV{SSL\_SERVER\_V\_END} = Apr 20 15:32:52 2020 GMT

$ENV{SSL\_SERVER\_V\_START} = Apr 22 15:32:52 2015 GMT

$ENV{SSL\_SESSION\_ID} = 5E09DF13A90C93E8910FA23FDA22CCFD44D1DF498176ADF39A639343BFC3E67D

$ENV{SSL\_TLS\_SNI} = dakoss.bhcorp.ad

$ENV{SSL\_VERSION\_INTERFACE} = mod\_ssl/2.2.15

$ENV{SSL\_VERSION\_LIBRARY} = OpenSSL/1.0.1e-fips

Perl function - CGI::param()

param("debug") = 1

param("myFlag") = Loves Energy

BHC function - DirAccess::directory\_structure()

Return Array = /var/www/dakossdev

Return Array = /cgi-bin

Return Array = /www

Return Array = /test