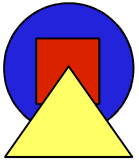


AFS Perl Module

Alf Wachsmann

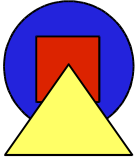
Stanford Linear Accelerator Center

alfw@slac.stanford.edu



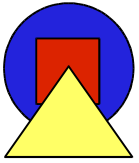
Brief History of AFS Perl

- original AFS interface was written by Roland Schemers @ Stanford University in ~1994
- Norbert Grüner (MPA Garching, Germany) picked up ownership ~ 2001; put it on CPAN
- current, more OO style release was written by him
- Summer 2003:
 - Steven Jenkins @ ETSU suggested extension (vos, bos)
 - SLAC payed Venkata Phani Achanta from ETSU to come to SLAC and do the programming
 - supervised/supported at SLAC by Alf Wachsmann
- API for bos commands still missing on CPAN (first release candidate will come in April '04)



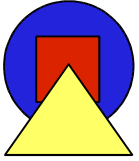
Other AFS Perl Modules

- AFS:
 - This AFS Perl module is written in XS and uses the API directly
 - It is available from CPAN:
<http://search.cpan.org/~nog/AFS-2.2.1/>
- AFS-Command:
 - Phil Moore wrote an AFS Perl module that is an OO wrapper around the AFS command suite binaries
 - It is available from CPAN:
<http://search.cpan.org/~wpmoore/AFS-Command-1.4/>



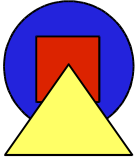
What does the Perl module do?

- Perl functions for all AFS command line tools:
klog, tokens, fs, kas, pts, vos, **bos**
- separated into 14 Perl classes:
 - Access to Cache Manager routines: **AFS::CM**
 - Access to File Server routines: **AFS::FS**
 - Managing ACLs: **AFS::ACL**
 - Manipulating Cell Configuration: **AFS::Cell**
 - Access to Protection Server routines: **AFS::PTS**
 - Access to Authentication Server routines: **AFS::KAS**
 - Manage Kernel Token Cache: **AFS::KTC_TOKEN**
 - Deal with principals: **AFS::KTC_PRINCIPAL**
 - Deal with encryption keys: **AFS::KTC_EKEY**



What does the Perl module do?

- Access to share utility functions: `AFS::Utils`
- Error handling, debugging, etc.: `AFS`
- Access to VLDB related functions: `AFS::VLDB`
- Access to Volume Location Server: `AFS::VOS`
- Access to Basic Overseer Server: `AFS::BOS`



pts Example

Equivalent of `pts adduser -user $user -group $group`

```
#!/usr/bin/perl -w
```

```
use strict;  
use AFS::PTS;
```

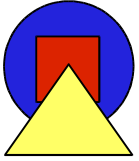
```
die "Usage: $0 user group\n" if $#ARGV != 1;
```

```
my $user = shift;  
my $group = shift;
```

```
my $pts = AFS::PTS->new;
```

```
$ok = $pts->adduser($user, $group);
```

```
print "AFS::CODE = $AFS::CODE\n";  
print "success = $ok\n";
```



kas Example...

Equivalent of `kas examine $user -admin $princ`

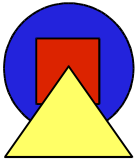
```
#!/usr/bin/perl -w
use strict;
use AFS::KAS;
use AFS::KTC_PRINCIPAL;
use AFS::KTC_TOKEN;
use AFS::KTC_EKEY;

die "Usage: $0 admin user\n" if ($#ARGV != 1);

my $princ = AFS::KTC_PRINCIPAL->new(shift);
my $key   = AFS::KTC_EKEY->ReadPassword($princ->name."s Password:");
my $token = AFS::KTC_TOKEN->GetAdminToken($princ, $key, 300);
my $kas   = AFS::KAS->AuthServerConn($token, &AFS::KA_MAINTENANCE_SERVICE);

my $user = AFS::KTC_PRINCIPAL->new(shift);
my $entry = $kas->getentry($user->name, $user->instance);

print "\ngetentry: User data for ", $user->name, $user->instance, ": \n";
foreach my $tp_key (sort keys %$entry) {
    printf("%20s %s\n", $tp_key, $$entry{$tp_key});
}
```

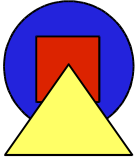


...kas Example

Output from previous example:

admin's Password:

```
getentry: User data for alfw:
change_password_time 1059790609
                    flags      1
                    keyChecksum -8073956
                    key_version 12
max_ticket_lifetime  90000
                    minor_version 2
                    misc_auth_bytes -8073956
modification_time    921830894
modification_user     sysctl
user_expiration      -1
```



vos Example 1...

Equivalent of `vos listvol $server $part -extended`

```
#!/usr/bin/perl -w
```

```
use strict;  
use AFS::VOS;  
use Time::localtime;
```

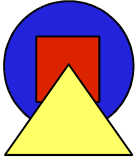
```
die "Usage: $0 server [partition]\n" if $#ARGV < 0;
```

```
my $server = shift; my $part = shift; $part = '' unless $part;  
my $fast = 0; my $extend = 1;
```

```
my $vos = AFS::VOS->new;  
print "Error: ", $AFS::CODE, "\n" if ($AFS::CODE);
```

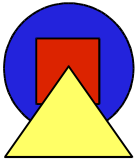
```
my $partlist = $vos->listvol($server, $part, $fast, $extend);  
print "Error: ", $AFS::CODE, "\n" if ($AFS::CODE);
```

```
print_ext($partlist);
```



...vos Example 1...

```
sub print_ext {
  my $partlist = shift;
  foreach my $part (sort keys %$partlist) {
    print "Partition $part:\n";
    foreach my $vol (sort keys %{$partlist->{$part}}) {
      if ($vol =~ /total/) {
        print "\tKey: $vol, Value: $partlist->{$part}->{$vol}\n";
      } else {
        print "\tVolume: $vol\n";
        foreach my $key (sort keys %{$partlist->{$part}->{$vol}}) {
          if ($key =~ /(Reads|Writes|>1wk|1day-1wk|1hr-1day|10min-1hr|1-10min|0-60sec)/) {
            print "\t\tKey $key:\n";
            foreach (sort keys %{$partlist->{$part}->{$vol}->{$key}}) {
              print "\t\t\tKey: $_, Value: $partlist->{$part}->{$vol}->{$key}->{$_}\n";
            }
          } elsif ($key =~ /(updateDate|creationDate)/) {
            print "\tKey: $key, Value: ", ctime($partlist->{$part}->{$vol}->{$key}), "\n";
          } else {
            print "\tKey: $key, Value: $partlist->{$part}->{$vol}->{$key}\n";
          }
        }
      }
    }
  }
}
```



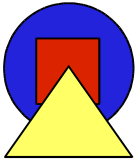
...vos Example 1

Output from previous example: Partition /vicepa:

.....

```
Volume: root.afs
Key: backupID, Value: 536870914
Key: cloneID, Value: 536870913
Key: creationDate, Value: Mon May 19 13:41:40 1997
Key: inUse, Value: On-line
Key: maxquota, Value: 5000
Key: parentID, Value: 536870912
Key: partition, Value: /vicepa
Key: server, Value: afs05.slac.stanford.edu
Key: type, Value: RW
Key: updateDate, Value: Tue Dec 9 14:16:10 2003
Key: valid, Value: 536870912
  Key 0-60sec:
    Key: dirDiffAuth, Value: 0
    Key: dirSameAuth, Value: 0
    Key: fileDiffAuth, Value: 0
    Key: fileSameAuth, Value: 0
```

.....



vos Example 2

Equivalent of `vos listvldb $vol`

```
#!/usr/bin/perl -w
```

```
use strict;  
use AFS::VLDB;
```

```
my $vol = shift;
```

```
$vldb = AFS::VLDB->new;
```

```
my $vldblist = $vldb->listvldbentry($vol);  
print "Error: ", $AFS::CODE, "\n"  
if ($AFS::CODE);
```

```
print_vldblist($vldblist);
```

Key: `root.afs`

Key: `nServers`, Value: `9`

Key: `flags`, Value: `28672`

Key: `Backup`, Value: `536870914`

Key: `ROnly`, Value: `536870913`

Key: `RWrite`, Value: `536870912`

Server number 1:

Key: `serverFlags`, Value: `4`

Key: `name`, Value: `afs05.slac.stanford.edu`

Key: `type`, Value: `RW`

Key: `partition`, Value: `/vicepa`

Server number 2:

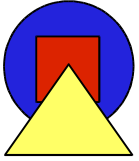
Key: `serverFlags`, Value: `2`

Key: `name`, Value: `afs10.slac.stanford.edu`

Key: `type`, Value: `RO`

Key: `partition`, Value: `/vicepa`

.....



Creating AFS accounts...

```
#!/usr/bin/perl -w
# basic script to create a new AFS account

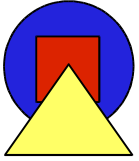
use strict;
use AFS::VOS;
use AFS::PTS;
use AFS::ACL;
use AFS::Cell qw(localcell);
use AFS::FS qw(setquota);

my $user = shift;
my $initial = substr($user, 0, 1);

# create PTS entry
my $pts = AFS::PTS->new;
my $id = $pts->createuser($user);

# create user volume
my $vos = AFS::VOS->new;
$ok = $vos->create('server1', '/vicepa', "u.$user"); #load balance?!

# continue next slide
```



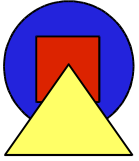
...Creating AFS accounts

```
# rest of basic AFS account creation

# create volume mountpoint
my $cell = localcell;
$ok = mkmount("/afs/$cell/user/$initial/$user", "u.$user");
$ok = setquota("/afs/$cell/user/$initial/$user", 50000);

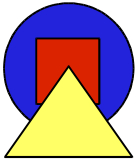
# set ACLs in new directory
my $acl = AFS::ACL->new('');
$acl->set($user                => 'rlidwka');
$acl->set('system:administrators' => 'rlidwka');
$acl->set('system:backup'         => 'rl');
$acl->set('system:anyuser'       => 'l');
my $ok = $acl->apply("/afs/$cell/user/$initial/$user");

# create all subdirectories and set ACLs
# copy .dot files into new home directory
```



Implementation

- A lot of code was "cut-and-pasted" from OpenAFS sources into AFS.xs to achieve same functions as AFS command suite binaries
- By using C API directly, significantly faster than any wrapper can be
- Compiles with OpenAFS and IBM/Transarc's AFS
- Compiles on many Unix platforms



Further Information

- AFS Perl module comes with excellent documentation
- a lot of helpful example scripts included
- plays nicely with other modules
- Kerberos 5:
 - `AFS::KAS` needs to be replaced with Perl module for your Kerberos 5 distro (`MIT: Authn::Krb5::Admin`, `Heimdal: Heimdal-Kamd5`)
 - `AFS::KTC_PRINCIPAL`, `AFS::KTC_TOKEN` need to be replaced with `Authn::Krb5`
- available on CPAN