Module Version: 2.08

NAME

Tuxedo - Perl extension module for Tuxedo

SYNOPSIS

use Tuxedo;

DESCRIPTION

This module provides the following functionality...

- **'C' style interface**

  The Tuxedo perl module gives you access to almost all of the tuxedo 8.1 apis from perl. In most cases you can take the C API you already familiar with, apply perl semantics to it, and write working tuxedo programs in perl.

- **Object wrapping of C structures**

  Many tuxedo functions take pointers to C structures as function parameters. To preserve the C interface, this module provides perl objects that encapsulate the C structures used by tuxedo. These objects allow the user to create and manipulate the elements of these C structures, and these objects are then passed as parameters to the perl version of these tuxedo C functions.

- **buffer management**

  Perl classes exist for each buffer type to allow for easy manipulation of buffer contents and automatic memory cleanup when no more references to the buffer exist.

- **callback subs**

  perl subs can be registered as unsolicited message handlers and signal handlers.

- **FML/FML32 field table support**

  This module includes the mkfldpm32.pl script that is the perl equivalent of the tuxedo mkfldhdr32 program. It accepts a field table file as input and produces a *.pm file that can be included in a perl script, so field identifiers can be referenced by id.

- **perl tuxedo services**

  You can now write tuxedo services in perl. When you build the Tuxedo module, it should create a tuxedo server called PERLSVR. This is a tuxedo server that contains an embedded perl interpretor for executing perl tuxedo services. When PERLSVR boots up, it parses the perlsvr.pl script, which at the moment it expects to find in its working directory. The location of perlsvr.pl will be configurable in a future version. The perlsvr.pl script is run as the tpsvrinit routine. You can modify perlsvr.pl to define any subs you want to be tuxedo services and advertise these subs.

  There are a few rules for writing subs that are to be run as tuxedo services.

  1) They must accept a single input parameter which is a reference to a TPSVCINFO_PTR object.

  2) They must return 5 parameters corresponding to the parameters of the tpreturn tuxedo function. You don't call tpreturn directly from a perl sub tuxedo service. When the sub returns, the PERLSVR will extract the return values from the perl stack and call tpreturn for you.

  Below is the perlsvr.pl that is included with this distribution. It demonstrates how to write and advertise two simple perl subs that act as tuxedo services.

```
use Tuxedo;

sub TOUPPER {
    my ($tpsvcinfo) = @_;
    my ($inbuf) = $tpsvcinfo->data;
    $inbuf->value( ($newval = uc($inbuf->value)) );
    return ( TPSUCCESS, 0, $inbuf, $tpsvcinfo->len, 0 );
}

sub REVERSE {
    my ($tpsvcinfo) = @_;
    my ($buf) = $tpsvcinfo->data;
    $buf->value( ($newval = reverse($buf->value)) );
    return ( TPSUCCESS, 0, $buf, $tpsvcinfo->len, 0 );
}

tpadvertise( "TOUPPER", \&TOUPPER );
tpadvertise( "REVERSE", \&REVERSE );
```

Future versions of this module will include

- **workstation and native modules**

Different modules will exist for native and workstation tuxedo development. Currently native is the default.

- **An object oriented tuxedo interface**

  Version 1 of the Tuxedo module only presented an object oriented interface to the user. This version of the Tuxedo module presents the original C interface to make perl tuxedo development easier for experienced tuxedo programmers. The object oriented interface will co-exist with the C interface in a future version of this module.

'C' STYLE INTERFACE

An example is probably the best way to demonstrate the interface provided by the Tuxedo perl module for writing tuxedo programs. The following example shows how to connect to a tuxedo system and make a service call.

```
use Tuxedo;
use tpadm;

my $password = "password";

# Allocate a TPINIT buffer
my $tpinitbfr = tpalloc( "TPINIT",
                         0,
                         TPINITNEED( length($password) )
                         );

# populate the TPINIT buffer
$tpinitbfr->usrname( "Anthony" );
$tpinitbfr->cltname( "PERL" );
$tpinitbfr->data( $password );
$tpinitbfr->passwd( "tuxedo" );
$tpinitbfr->flags( TPMULTICONTEXTS );

# connect to tuxedo
if ( tpinit( $tpinitbfr ) == -1 ) {
   die "tpinit failed: " . tpstrerror(tperrno) . "\n";
}

# allocate FML32 buffers
my $inbuf = tpalloc( "FML32", 0, 1024 );
my $outbuf = tpalloc( "FML32", 0, 1024 );
if ( $inbuf == undef || $outbuf == undef ) {
  die "tpalloc failed: " . tpstrerror(tperrno) . "\n";
}

# populate the FML32 inbuf
$rc = Fappend32( $inbuf, TA_CLASS, "T_CLIENT", 0 );
if ( $rc == -1 ) {
  die "Fappend failed: " . Fstrerror32(Ferror32) . "\n";
}
$rc = Fappend32( $inbuf, TA_OPERATION, "GET", 0 );
$rc = Findex32( $inbuf, 0 );

# call the .TMIB service
```

```
$rc = tpcall( ".TMIB", $inbuf, 0, $outbuf, $olen, 0 );
if ( $rc == -1 ) {
  die ( "tpcall failed: " . tpstrerror(tperrno) . ".\n" );
}

# print the returned buffer
tuxputenv( "FIELDTBLS32=tpadm" );
tuxputenv( "FLDTBLDIR32=" . tuxgetenv("TUXDIR") . "/udataobj" );
Fprint32( $outbuf );

# disconnect from tuxedo
tpterm();
```

OBJECT WRAPPING OF C STRUCTURES

The Tuxedo module provides perl objects for creating and reading/writing elements of tuxedo C structures. The objects and methods available are...

- TPINIT_PTR

  This object is returned by a call to **tpalloc** when specifying a "TPINIT" buffer type. The methods available on this object are...

  -> usrname

  get and set the usrname

  -> cltname

  get and set the cltname

  -> passwd

  get and set the passwd

  -> grpname

  get and set the grpname

  -> flags

  get and set the flags

  -> datalen

  get and set the datalen

  -> data

  get and set the data

- CLIENTID_PTR

  ->new

create a new instance of a CLIENTID_PTR object.

```
# example of creating a new CLIENTID_PTR object
$clientid = CLIENTID_PTR::new();
```

->clientdata

Get and set the clientdata.

```
# to set the clientdata element
$clientid->clientdata( 1, 2, 3, 4 );

# to get the clientdata element.  This returns an arary of 4 longs
@clientdata = $clientid->clientdata;
```

- TPTRANID_PTR

  ->new

  create a new instance of a TPTRANID_PTR object.

```
# example of creating a new TPTRANID_PTR object
$tptranid = TPTRANID_PTR::new();
```

  ->info

  Get and set the info.

```
# to set the info element
$tptranid->info( 1, 2, 3, 4, 5, 6 );

# to get the info element.  This returns an arary of 6 longs
@info = $tptranid->info;
```

- XID_PTR

  ->new

  create a new instance of a XID_PTR object.

```
# example of creating a new XID_PTR object
$xid = XID_PTR::new();
```

  ->formatID

  Get and set the formatID.

  ->gtrid_length

Get and set the gtrid_length.

->bqual_length

Get and set the bqual_length.

->data

Get and set the data.

- TPQCTL_PTR

->new

create a new instance of a TPQCTL object.

```
# example of creating a new TPQCTL_PTR object
$tpqctl = TPQCTL_PTR::new();
```

->flags

Get and set the flags.

->deq_time

Get and set the deq_time.

->priority

Get and set the priority.

->diagnostic

Get and set the diagnostic.

->msgid

Get and set the msgid.

->corrid

Get and set the corrid.

->replyqueue

Get and set the replyqueue.

->failurequeue

Get and set the failurequeue.

->cltid

Get and set the cltid.

->urcode

Get and set the urcode.

->appkey

Get and set the appkey.

->delivery_qos

Get and set the delivery_qos.

->reply_qos

Get and set the reply_qos.

->exp_time

Get and set the exp_time.

- TPEVCTL_PTR

  ->new

  create a new instance of a TPEVCTL_PTR object.

  ```
  # example of creating a new TPEVCTL_PTR object
  $tpevctl = TPEVCTL_PTR::new();
  ```

  ->flags

  Get and set the flags.

  ->name1

  Get and set the name1.

  ->name2

  Get and set the name2.

  ->qctl

  Get and set the qctl.

- TXINFO_PTR

  ->new

  create a new instance of a TXINFO_PTR object.

```
# example of creating a new TXINFO_PTR object
$txinfo = TXINFO_PTR::new();
```

->xid

Get and set the xid.

->when_return

Get and set the when_return.

->transaction_control

Get and set the transaction_control.

->transaction_timeout

Get and set the transaction_timeout.

->transaction_state

Get and set the transaction_state.

BUFFER MANAGEMENT

All buffers returned by tpalloc are blessed as the type of buffer that you allocate. This means there are methods you can call on the returned buffer to manipulate the buffer contents. For example, to allocate and populate a TPINIT buffer, you would do the following.

```
# Allocate a TPINIT buffer
my $tpinitbfr = tpalloc( "TPINIT",
                         0,
                         TPINITNEED( length($password) )
                         );

# populate the TPINIT buffer
$tpinitbfr->usrname( "Anthony" );
$tpinitbfr->cltname( "PERL" );
```

In this example, tpalloc returns a reference to a TPINIT_PTR object which has usrname, cltname and other methods available to modify the contents of the underlying TPINIT buffer. If you allocate an FML32 buffer, tpalloc will return a FBFR32_PTR object which has different methods available to manipulate buffer contents.

Another benefit of this approach is that a DESTROY method is automatically called when the reference count of each tuxedo buffer becomes zero, so that any allocated memory is consequently automatically freed for you.

CALLBACK SUBS

The Tuxedo module allows you to create perl subs that are registered as unsolicited message and signal handers. The example below demonstrates how to do this.

```
# create a sub to use as an unsolicited message handler
sub unsol_msg_handler
{
  my( $buffer, $len, $flags ) = @_;

  # assume the recieved message is an FML32 buffer
  Fprint32( $buffer );

  printf( "unsol_msg_handler called!\n" );
}

# create a sub to use as a signal handler
sub sigusr2_handler
{
  my( $signum ) = @_;
  printf( "caught SIGUSR2\n" );
}

# register unsol_msg_hander with tuxedo
tpsetunsol( \&unsol_msg_handler );

# register sigusr2_handler with tuxedo.  SIGUSR2 is 17
Usignal( 17, \&sigusr2 );
```

FML/FML32 FIELD TABLE SUPPORT

This version of the perl module also includes a useful utility script, mkfldpm32.pl, which is the perl equivalent of mkfldhdr32. It will parse a field table file and create a .pm file that you can include in any perl scripts to access fields in an FML/FML32 buffer directly by id instead of name.

Exported constants

```
  BADFLDID
  FLD_CARRAY
  FLD_CHAR
  FLD_DOUBLE
  FLD_FLOAT
  FLD_FML32
  FLD_LONG
  FLD_PTR
  FLD_SHORT
  FLD_STRING
  FLD_VIEW32
  TP_CMT_COMPLETE
  TP_CMT_LOGGED
  TPABSOLUTE
  TPACK
  TPAPPAUTH
  TPCONV
  TPCONVCLTID
```

```
    TPCONVMAXSTR
    TPCONVTRANID
    TPCONVXID
    TPEABORT
    TPEBADDESC
    TPEBLOCK
    TPEDIAGNOSTIC
    TPEEVENT
    TPEHAZARD
    TPEHEURISTIC
    TPEINVAL
    TPEITYPE
    TPELIMIT
    TPEMATCH
    TPEMIB
    TPENOENT
    TPEOS
    TPEOTYPE
    TPEPERM
    TPEPROTO
    TPERELEASE
    TPERMERR
    TPESVCERR
    TPESVCFAIL
    TPESYSTEM
    TPETIME
    TPETRAN
    TPEXIT
    TPFAIL
    TPGETANY
    TPGETANY
    TPGOTSIG
    TPINITNEED
    TPMULTICONTEXTS
    TPNOAUTH
    TPNOBLOCK
    TPNOCHANGE
    TPNOREPLY
    TPNOTIME
    TPNOTRAN
    TPRECVONLY
    TPSA_FASTPATH
    TPSA_PROTECTED
    TPSENDONLY
    TPSIGRSTRT
    TPSUCCESS
    TPSYSAUTH
    TPTOSTRING
    TPTRAN
    TPU_DIP
    TPU_IGN
    TPU_MASK
    TPU_SIG
    TPU_THREAD

    TPQCORRID
    TPQFAILUREQ
    TPQBEFOREMSGID
```

```
    TPQGETBYMSGIDOLD
    TPQMSGID
    TPQPRIORITY
    TPQTOP
    TPQWAIT
    TPQREPLYQ
    TPQTIME_ABS
    TPQTIME_REL
    TPQGETBYCORRIDOLD
    TPQPEEK
    TPQDELIVERYQOS
    TPQREPLYQOS
    TPQEXPTIME_ABS
    TPQEXPTIME_REL
    TPQEXPTIME_NONE
    TPQGETBYMSGID
    TPQGETBYCORRID
    TPQQOSDEFAULTPERSIST
    TPQQOSPERSISTENT
    TPQQOSNONPERSISTENT

    TPKEY_SIGNATURE
    TPKEY_DECRYPT
    TPKEY_ENCRYPT
    TPKEY_VERIFICATION
    TPKEY_AUTOSIGN
    TPKEY_AUTOENCRYPT
    TPKEY_REMOVE
    TPKEY_REMOVEALL
    TPKEY_VERIFY
    TPEX_STRING
    TPSEAL_OK
    TPSEAL_PENDING
    TPSEAL_EXPIRED_CERT
    TPSEAL_REVOKED_CERT
    TPSEAL_TAMPERED_CERT
    TPSEAL_UNKNOWN
    TPSIGN_OK
    TPSIGN_PENDING
    TPSIGN_EXPIRED
    TPSIGN_EXPIRED_CERT
    TPSIGN_POSTDATED
    TPSIGN_REVOKED_CERT
    TPSIGN_TAMPERED_CERT
    TPSIGN_TAMPERED_MESSAGE
    TPSIGN_UNKNOWN
```

AUTHOR

Anthony Fryer, apfryer@hotmail.com

SEE ALSO

perl(1). http://e-docs.bea.com/tuxedo/tux81/interm/ref.htm