Example Mobile Agents

This document presents a small collection of mobile agents that were developed to test the functionality of the facility.

The `increment1` Example

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

#
# A Simple Perl Mobile Agent.
#
# This is agent increment #1 (refer to Page 7 of "Functional and Requirements
# Specification for A Mobile Agent Execution and Location Environment for
# the Perl Programming Language", Section 2.2.1).
#
# Author: Paul Barry, paul.barry@itcarlow.ie, IT Carlow.
#

use strict;

#my $LOCATION = '149.153.23.52';
# my $LOCATION = 'pbmac.itcarlow.ie';
my $LOCATION = 'localhost';
my $PORT     = 2001;

use Mobile::Executive;

print "Hello on the original Location.\n";
Mobile::Executive::relocate( $LOCATION, $PORT );
print "Hello on the remote Location.\n";
$LOCATION = 'pblinux.itcarlow.ie';
$PORT = 8001;

Mobile::Executive::relocate( $LOCATION, $PORT );
print "Hello on the last Location.\n";
```

The increment2a Example

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

#
# Relocating scalars.
#
# This is agent increment #2a (refer to Page 7 of "Functional and Requirements
# Specification for A Mobile Agent Execution and Location Environment for
# the Perl Programming Language", Section 2.2.2).
#
# Author: Paul Barry, paul.barry@itcarlow.ie, IT Carlow.
#

use strict;

my $LOCATION = shift || 'localhost';
my $PORT     = shift || 2001;

use Mobile::Executive;

my $message = "Did this print?\n";
Mobile::Executive::relocate( $LOCATION, $PORT );
print $message;
```

The increment2b Example

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

#
# Relocating Arrays/Lists.
#
# This is agent increment #2b (refer to Page 7 of "Functional and Requirements
# Specification for A Mobile Agent Execution and Location Environment for
# the Perl Programming Language", Section 2.2.2).
#
# Author: Paul Barry, paul.barry@itcarlow.ie, IT Carlow.
#

use strict;

my $LOCATION = shift || 'localhost';
my $PORT     = shift || 2001;

use Mobile::Executive;

my @countdown = ( '5', '4', '3', '2', '1', 'lift-off!' );
Mobile::Executive::relocate( $LOCATION, $PORT );
foreach my $count ( @countdown )
{
    print $count, "\n";
}
```

The increment2c Example

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

#
# Relocating Associative Arrays (hashes).
#
# This is agent increment #2c (refer to Page 8 of "Functional and Requirements
# Specification for A Mobile Agent Execution and Location Environment for
# the Perl Programming Language", Section 2.2.2).
#
# Author: Paul Barry, paul.barry@itcarlow.ie, IT Carlow.
#

use strict;

my $LOCATION = shift || 'localhost';
my $PORT     = shift || 2001;

use Mobile::Executive;

my %pairings = ( 'Tom'    => 'Jerry',
                 'Woody'  => 'Buzz',
                 'Kermit' => 'Miss Piggy',
                 'Barney' => 'BJ' );
Mobile::Executive::relocate( $LOCATION, $PORT );
while ( my ( $left, $right ) = each ( %pairings ) )
{
    print "$left is associated with $right.\n";
}
```

The `increment2d` Example

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

#
# Relocating References to Scalars.
#
# This is agent increment #2d (refer to Page 8 of "Functional and Requirements
# Specification for A Mobile Agent Execution and Location Environment for
# the Perl Programming Language", Section 2.2.2).
#
# Author: Paul Barry, paul.barry@itcarlow.ie, IT Carlow.
#

use strict;

my $LOCATION = shift || 'localhost';
my $PORT     = shift || 2001;

use Mobile::Executive;

my $countdown = 'lift-off!';
my $countref = \$countdown;
Mobile::Executive::relocate( $LOCATION, $PORT );
print $$countref, "\n";
```

The increment2e Example

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

#
# Relocating References to Arrays/Lists.
#
# This is agent increment #2e (refer to Page 8 of "Functional and Requirements
# Specification for A Mobile Agent Execution and Location Environment for
# the Perl Programming Language", Section 2.2.2).
#
# Author: Paul Barry, paul.barry@itcarlow.ie, IT Carlow.
#

use strict;

my $LOCATION = shift || 'localhost';
my $PORT     = shift || 2001;

use Mobile::Executive;

my @countdown = ( '5', '4', '3', '2', '1', 'lift-off!' );
my $countref = \@countdown;
Mobile::Executive::relocate( $LOCATION, $PORT );
print @$countref[ 2 ], "\n";
```

The `increment2f` Example

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

#
# Relocating References to Hashes.
#
# This is agent increment #2f (refer to Page 8 of "Functional and Requirements
# Specification for A Mobile Agent Execution and Location Environment for
# the Perl Programming Language", Section 2.2.2).
#
# Author: Paul Barry, paul.barry@itcarlow.ie, IT Carlow.
#

use strict;

my $LOCATION = shift || 'localhost';
my $PORT     = shift || 2001;

use Mobile::Executive;

my %countdown = ( "five"  => '5',
                  "four"  => '4',
                  "three" => '3',
                  "two"   => '2',
                  "one"   => '1',
                  "off"   => 'lift-off!' );
my $countref = \%countdown;
Mobile::Executive::relocate( $LOCATION, $PORT );
print $$countref{ "two" }, "\n";
```

The increment3a Example

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

#
# This is agent increment #3 (refer to Page 8 of "Functional and Requirements
# Specification for A Mobile Agent Execution and Location Environment for
# the Perl Programming Language", Section 2.2.3).
#
# Author: Paul Barry, paul.barry@itcarlow.ie, IT Carlow.
#

use strict;

my $LOCATION = shift || 'localhost';
my $PORT     = shift || 2001;

use Mobile::Executive;

use Objs::Simple;
use Objs::Simple2;

my $simple_object = Objs::Simple->new( "This is a Simple Object." );

Mobile::Executive::relocate( $LOCATION, $PORT );

$simple_object->println_value;
```

The increment3b Example

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

#
# This is agent increment #3 (refer to Page 8 of "Functional and Requirements
# Specification for A Mobile Agent Execution and Location Environment for
# the Perl Programming Language", Section 2.2.3).
#
# Author: Paul Barry, paul.barry@itcarlow.ie, IT Carlow.
#

use strict;

my $LOCATION = shift || 'localhost';
my $PORT     = shift || 2001;

use Mobile::Executive;

use Objs::Simple;
use Objs::Simple2;

my $simple_object = Objs::Simple->new( "This is a Simple Object." );
my $simple2_object = Objs::Simple2->new( "This is a Simple2 Object." );

Mobile::Executive::relocate( $LOCATION, $PORT );

$simple_object->println_value;
$simple2_object->println_value;
```

The basic_increment Example

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

#
# The most basic of mobile agents.  It does nothing but relocate.
#
# Author: Paul Barry, paul.barry@itcarlow.ie, IT Carlow.
#

use Mobile::Executive;

relocate( '127.0.0.1', 2001 );
relocate( '127.0.0.1', 6001 );
```

The drivespace Example

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

#
# drivespace
#

use strict;
use Mobile::Executive;

my %space = ();

sub do_it {
    # This code is executed at each Location.

    return scalar ‘df -Th‘;
}

relocate( ’pbmac.itcarlow.ie’, 2001 );
$space{ ’pbmac.itcarlow.ie’ } = do_it;

relocate( ’pblinux.itcarlow.ie’, 4001 );
$space{ ’pblinux.itcarlow.ie’ } = do_it;

relocate( ’testimac.itcarlow.ie’, 3001 );
$space{ ’testimac.itcarlow.ie’ } = do_it;

relocate( ’pblinux.itcarlow.ie’, 4001 );

foreach my $host ( keys %space )
{
    print "$host reported diskspace of: \n\n$space{ $host }\n";
}
```

The howlongup Example

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

#
# howlongup
#

use strict;
use Mobile::Executive;

my %howlong = ();

sub do_it {
    # This code is executed at each Location.

    return scalar `uptime`;
}

relocate( 'pbmac.itcarlow.ie', 2001 );
$howlong{ 'pbmac.itcarlow.ie' } = do_it;

relocate( 'testimac.itcarlow.ie', 2001 );
$howlong{ 'testimac.itcarlow.ie' } = do_it;

relocate( 'pblinux.itcarlow.ie', 2001 );
$howlong{ 'pblinux.itcarlow.ie' } = do_it;

foreach my $host ( keys %howlong )
{
    print "$host reported uptime of: $howlong{ $host }";
}
```

The `witch`[1] Example

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

#
# witch - which version of an executable runs a particular command.
#

use strict;
use Mobile::Executive;

my %which = ();

my $which_program = shift;

if ( !defined( $which_program ) ) { die "provide an exec name to which\n"; }

sub do_it {
    # This code is executed at each Location.
    my $exec = shift;

    $ENV{'PATH'} = '/bin:/usr/bin:/usr/local/bin';
    delete @ENV{ 'IFS', 'CDPATH', 'ENV', 'BASH_ENV' };

    return scalar `which $exec`;
}

relocate( 'testimac', 3001 );
$which{ 'testimac.itcarlow.ie' } = do_it( $which_program );

relocate( 'pbmac.itcarlow.ie', 2001 );
$which{ 'pbmac.itcarlow.ie' } = do_it( $which_program );

relocate( 'pblinux.itcarlow.ie', 4001 );
$which{ 'pblinux.itcarlow.ie' } = do_it( $which_program );

foreach my $exec ( keys %which )
{
    print "$exec has $which_program available at: $which{ $exec }";
}
```

[1]Note: `which` is an operating system command, and is therefore a reserved word.

The scooby_multiwho Example

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

#
# scooby_multiwho - The "Scooby" version of the multiwho program.
#

use strict;
use Mobile::Executive;

my @who_list = ();

sub do_who {
    # This code is executed at each Location.

    $ENV{'PATH'} = '/bin:/usr/bin:/usr/local/bin';
    delete @ENV{ 'IFS', 'CDPATH', 'ENV', 'BASH_ENV' };

    my @who_output    = `who`;
    my @do_who_output = ();

    foreach my $line ( @who_output )
    {
        $line =~ /^(\w+) /;
        push @do_who_output, $1;
    }

    return @do_who_output;
}

relocate( 'localhost', 2001 );
@who_list = ( @who_list, do_who );

# relocate( 'pbmac.itcarlow.ie', 2001 );
# @who_list = ( @who_list, do_who );

# relocate( 'glasnost.itcarlow.ie', 2001 );
# @who_list = ( @who_list, do_who );

relocate( 'pblinux.itcarlow.ie', 6001 );

foreach my $user ( @who_list )
{
    print "$user\n";
}
```