Subject: Re: something like perl's 'require 5.4'
Posted by John-David T. Smith on Mon, 16 Apr 2001 23:39:03 GMT
View Forum Message <> Reply to Message

Vapuser wrote:
>
> Hi all.
>
> I'm looking for something that check the version of the current IDL
> session against an input version, like perl's 'require 5.4'
> semantics and I *thought* I saw someone mention using something very
> much like this.
>
> Does anyone know of such a thing or was I just hallucinating?

It's ugly:

v=byte(!VERSION.RELEASE) & v=v[where(v ge 48 and v le 57)]
v=float(string(v[0]))+float('.'+string(v[1:*]))

if v lt 5.4 then message, "IDL v5.4 required"

This is to get around release names like '5.3.2a' or some such. I you
are sure you just want 5.4 you can of course do the much cleaner

if !VERSION.RELEASE eq '5.4' ...

JD

Subject: Re: something like perl's 'require 5.4'
Posted by Craig Markwardt on Mon, 16 Apr 2001 23:57:18 GMT
View Forum Message <> Reply to Message

Vapuser <vapuser@catspaw.jpl.nasa.gov> writes:
> Hi all.
>
> I'm looking for something that check the version of the current IDL
> session against an input version, like perl's 'require 5.4'
> semantics and I *thought* I saw someone mention using something very
> much like this.
>
> Does anyone know of such a thing or was I just hallucinating?

Extending on JD's answer with two more possibilities:

if !version.release LT '5.4' then message, 'ERROR'

if double(!version.release) LT 5.4 then message, 'ERROR'

The first comparison is a string compare, while the second one is a
numeric compare. There is a slight difference, but in practice they
are identical, and they also both handle the case of 5.4.1, etc. The
former will handle 5.4.1a, but I think it's rare. I always use the
latter.

Craig

--
--- --------------
Craig B. Markwardt, Ph.D.   EMAIL: craigmnet@cow.physics.wisc.edu
Astrophysics, IDL, Finance, Derivatives | Remove "net" for better response
--- --------------

Subject: Re: something like perl's 'require 5.4'
Posted by John-David T. Smith on Tue, 17 Apr 2001 02:57:29 GMT
View Forum Message <> Reply to Message

Craig Markwardt wrote:
>
> Vapuser <vapuser@catspaw.jpl.nasa.gov> writes:
>> Hi all.
>>
>>   I'm looking for something that check the version of the current IDL
>>   session against an input version, like perl's 'require 5.4'
>>   semantics and I *thought* I saw someone mention using something very
>>   much like this.
>>
>>   Does anyone know of such a thing or was I just hallucinating?
>
> Extending on JD's answer with two more possibilities:
>
> if   !version.release LT '5.4' then message, 'ERROR'
> if double(!version.release) LT 5.4 then message, 'ERROR'
>
> The first comparison is a string compare, while the second one is a
> numeric compare. There is a slight difference, but in practice they
> are identical, and they also both handle the case of 5.4.1, etc. The
> former will handle 5.4.1a, but I think it's rare. I always use the
> latter.

Hmmm...

IDL> print,double('5.4') lt double ('5.4.1')
  0

IDL>print,double('5.4') eq double ('5.4.1')
  1

Hopefully RSI won't try to pull the latter on us.

On the other hand, your string method looks good. The best counter
example I could come up with is:

IDL> print,'5.5a' lt '5.5B'
  0

Not too likely. I think I'll adopt the string version.

Thanks,

JD

Subject: Re: something like perl's 'require 5.4'
Posted by Vapuser on Tue, 17 Apr 2001 16:05:08 GMT
View Forum Message <> Reply to Message

JD Smith <jdsmith@astro.cornell.edu> writes:

> Craig Markwardt wrote:
>>
>> Vapuser <vapuser@catspaw.jpl.nasa.gov> writes:
>>> Hi all.
>>>
>>> I'm looking for something that check the version of the current IDL
>>> session against an input version, like perl's 'require 5.4'
>>> semantics and I *thought* I saw someone mention using something very
>>> much like this.
>>>
>>> Does anyone know of such a thing or was I just hallucinating?
>>
>> Extending on JD's answer with two more possibilities:
>>
>> if   !version.release LT '5.4' then message, 'ERROR'
>> if double(!version.release) LT 5.4 then message, 'ERROR'
>>
>> The first comparison is a string compare, while the second one is a
>> numeric compare. There is a slight difference, but in practice they
>> are identical, and they also both handle the case of 5.4.1, etc. The
>> former will handle 5.4.1a, but I think it's rare. I always use the
>> latter.
>
> Hmmm...

>
> IDL> print,double('5.4') lt double ('5.4.1')
> 0
> IDL>print,double('5.4') eq double ('5.4.1')
> 1
>
> Hopefully RSI won't try to pull the latter on us.
>
> On the other hand, your string method looks good. The best counter
> example I could come up with is:
>
> IDL> print,'5.5a' lt '5.5B'
> 0
>
> Not too likely. I think I'll adopt the string version.
>
> Thanks,
>
> JD

How about:

```
 version=strupcase(strjoin(strsplit(!version.release,'[.-_]', /extract), ""))
 input_version=strupcase(strjoin(strsplit(desired_release,'[. -_]',/extract), ""))
if version lt input_version then 'eeek'
```

Clearly `desired_release' would have to be input as a string.

Of couse, this method 'requires' IDL 5.3 (IIRC), to get the regular expression semantics of strsplit. The other question is will RSI use any separator in version designations besides these three?

whd
--
William Daffer: 818-354-0161: William.Daffer@jpl.nasa.gov

Subject: Re: something like perl's 'require 5.4'
Posted by Craig Markwardt on Tue, 17 Apr 2001 18:21:32 GMT
View Forum Message <> Reply to Message

Vapuser <vapuser@catspaw.jpl.nasa.gov> writes:
> How about:
>
> version=strupcase(strjoin(strsplit(!version.release,'[.-_]', /extract), ""))
> input_version=strupcase(strjoin(strsplit(desired_release,'[. -_]',/extract), ""))
> if version lt input_version then 'eeek'
>

> Clearly `desired_release' would have to be input as a string.
>
> Of couse, this method 'requires' IDL 5.3 (IIRC), to get the regular
> expression semantics of strsplit. The other question is will RSI use
> any separator in version designations besides these three?

Umm, I never thought I'd be saying this, but aren't these a bit overengineered? I have found that in 95% of the cases the DOUBLE compare will work, and in 4% of the cases the STRING compare will work.

If you know which version number you are targetting, then clearly you can fashion a !VERSION.RELEASE comparison that will do the job.

William, you were probably thinking of a writing general procedure to handle this type of version enforcement. It's actually a good idea. But I've found that an "IF !VERSION.RELEASE..." is compact enough to go right into my code.

Craig

--

---------- ----------
Craig B. Markwardt, Ph.D. EMAIL: craigmnet@cow.physics.wisc.edu
Astrophysics, IDL, Finance, Derivatives | Remove "net" for better response
---------- ----------

Subject: Re: something like perl's 'require 5.4'
Posted by John-David T. Smith on Tue, 17 Apr 2001 18:45:38 GMT
View Forum Message <> Reply to Message

Craig Markwardt wrote:
>
> Vapuser <vapuser@catspaw.jpl.nasa.gov> writes:
>> How about:
>>
>> version=strupcase(strjoin(strsplit(!version.release,'[.-_]', /extract), ""))
>> input_version=strupcase(strjoin(strsplit(desired_release,'[. -_]',/extract), ""))
>> if version lt input_version then 'eeek'
>>
>> Clearly `desired_release' would have to be input as a string.
>>
>> Of couse, this method 'requires' IDL 5.3 (IIRC), to get the regular
>> expression semantics of strsplit. The other question is will RSI use
>> any separator in version designations besides these three?
>

> Umm, I never thought I'd be saying this, but aren't these a bit
> overengineered? I have found that in 95% of the cases the DOUBLE
> compare will work, and in 4% of the cases the STRING compare will
> work.

cough

<REITERATE>

IDL> print, double('5.4.1') gt 5.4
0

</REITERATE>

So I guess what you really you mean that in 95% of cases you don't care about the (possible) last digit on the version number. This is fine, just say so. Here's a good example of when that last digit matters, from the "What's New in IDL 5.2.1":

>>>> >> >>>>
The lib/compat directory containing the following obsolete date/time routines which are not Y2K compliant has been removed from the IDL distribution.

<SNIP>

REBIN Function Data Types Added:

The REBIN function now accepts 16-bit unsigned, 32-bit unsigned long, 64-bit long, or 64-bit unsigned long integer data types.
...

<SNIP>

HISTOGRAM Function Error with BINSIZE Set Fixed:

The HISTOGRAM function error resulting when the BINSIZE keyword is set has been fixed in this release.
...
>>>> >> >>>>

I like your even simpler string compare method, but this one just doesn't work as advertised.

JD

Subject: Re: something like perl's 'require 5.4'
Posted by Paul van Delst on Tue, 17 Apr 2001 19:44:23 GMT
View Forum Message <> Reply to Message

JD Smith wrote:
>
> Craig Markwardt wrote:
>>
>> Vapuser <vapuser@catspaw.jpl.nasa.gov> writes:
>>>> How about:
>>>>
>>>> version=strupcase(strjoin(strsplit(!version.release,'[.-_]', /extract), ""))
>>>> input_version=strupcase(strjoin(strsplit(desired_release,'[. -_]',/extract), ""))
>>>> if version lt input_version then 'eeek'
>>>>
>>>> Clearly `desired_release' would have to be input as a string.
>>>>
>>>> Of couse, this method 'requires' IDL 5.3 (IIRC), to get the regular
>>>> expression semantics of strsplit. The other question is will RSI use
>>>> any separator in version designations besides these three?
>>
>> Umm, I never thought I'd be saying this, but aren't these a bit
>> overengineered? I have found that in 95% of the cases the DOUBLE
>> compare will work, and in 4% of the cases the STRING compare will
>> work.
>
> *cough*
>
> <REITERATE>
>
> IDL> print, double('5.4.1') gt 5.4
> 0
>
> </REITERATE>

wot about

IDL> print, double('5.4.1') ge 5.4d0
1

?

Doesn't assuage your other concerns regarding the significance of the last digit however.
I use comparisons like the above for code that contains BREAK, CONTINUE, SWITCH, etc statements. Or similar for the version in which pointers and objects were introduced (5.2? can't remember).

I think it would be pretty difficult (for unsophisticated ol' me) to maintain code that contains version tests for when RSI fixes a bug -- like in the examples you listed....

although, HISTOGRAM and REBIN? Never use 'em. :o) What _possible_ use could they be?? (*).

paulv

(*) sarcasm alert for the irony impaired

--
Paul van Delst A little learning is a dangerous thing;
CIMSS @ NOAA/NCEP Drink deep, or taste not the Pierian spring;
Ph: (301)763-8000 x7274 There shallow draughts intoxicate the brain,
Fax:(301)763-8545 And drinking largely sobers us again.
paul.vandelst@noaa.gov Alexander Pope.

Subject: Re: something like perl's 'require 5.4'
Posted by Vapuser on Tue, 17 Apr 2001 20:44:19 GMT
View Forum Message <> Reply to Message

Craig Markwardt <craigmnet@cow.physics.wisc.edu> writes:

> Vapuser <vapuser@catspaw.jpl.nasa.gov> writes:
>> How about:
>>
>> version=strupcase(strjoin(strsplit(!version.release,'[.-_]', /extract), ""))
>> input_version=strupcase(strjoin(strsplit(desired_release,'[. -_]',/extract), ""))
>> if version lt input_version then 'eeek'
>>
>> Clearly `desired_release' would have to be input as a string.
>>
>> Of couse, this method 'requires' IDL 5.3 (IIRC), to get the regular
>> expression semantics of strsplit. The other question is will RSI use
>> any separator in version designations besides these three?
>
>
> Umm, I never thought I'd be saying this, but aren't these a bit
> overengineered? I have found that in 95% of the cases the DOUBLE
> compare will work, and in 4% of the cases the STRING compare will
> work.
>

When these two lines, which probably are too much for inline code,
are in their own procedure/function, I'd say 'no.'

I'll just have a 'require, "5.4"' as (one of) the first
things in any piece of code.

> If you know which version number you are targetting, then clearly you

> can fashion a !VERSION.RELEASE comparison that will do the job.
>

Probably so, but then why not just put it in a little procedure
which just fails to the command line?

Alternately, one could have a function like:

```
function collapseversion, version
  if n_elements(version) eq 0 then ERROR
   return,strupcase(strjoin(strsplit(version,'[.-_]',/extract), ""))
end
```

-- and do --

```
if collapseversion(!version.release) lt '54' then ERROR.
```

The best of both worlds.

> William, you were probably thinking of a writing general procedure to
> handle this type of version enforcement. It's actually a good idea.

Yep. My home software cache, which *was* built on my work stuff mostly,
got nuked and so I've decided to just rebuild it from scratch. I'm
in the utility writing section of the rebuild and finding myself
wanting to use features that have come up since I wrote most of
these utilites the first time, hence the need for this sort of
checking.

And since a failure of this sort of requirement checking really is
kinda fatal, i.e. this procedure/function calls stregex for which
you *must* have idl.version >= 5.3, I don't see what the problem
is. Put it in a proceduce and fail the sucker if the version isn't
high enough.

Oh... wait. I just thought of something! Maybe you could branch
around code that had some special dependencies.

No problem: write 2, a procedure that fails to the prompt and a
function that returns true/false.

> But I've found that an "IF !VERSION.RELEASE..." is compact enough to
> go right into my code.

True. I guess I'm of a different bent. I write lots of utility

functions/prodecures that I use all through my code, mainly because
I get tired of typing their contents all the time.

But thanks for the suggestions.

whd
--
William Daffer: 818-354-0161: William.Daffer@jpl.nasa.gov

Subject: Re: something like perl's 'require 5.4'
Posted by thompson on Tue, 17 Apr 2001 21:06:36 GMT
View Forum Message <> Reply to Message

Craig Markwardt <craigmnet@cow.physics.wisc.edu> writes:

> Vapuser <vapuser@catspaw.jpl.nasa.gov> writes:
>> How about:
>>
>> version=strupcase(strjoin(strsplit(!version.release,'[.-_]', /extract), ""))
>> input_version=strupcase(strjoin(strsplit(desired_release,'[. -_]',/extract), ""))
>> if version lt input_version then 'eeek'
>>
>> Clearly `desired_release' would have to be input as a string.
>>
>> Of couse, this method 'requires' IDL 5.3 (IIRC), to get the regular
>> expression semantics of strsplit. The other question is will RSI use
>> any separator in version designations besides these three?

> Umm, I never thought I'd be saying this, but aren't these a bit
> overengineered? I have found that in 95% of the cases the DOUBLE
> compare will work, and in 4% of the cases the STRING compare will
> work.

I've always found that a simple string comparison works in 100% of the cases.
Where doesn't it work?

Bill Thompson

Subject: Re: something like perl's 'require 5.4'
Posted by John-David T. Smith on Tue, 17 Apr 2001 21:10:36 GMT
View Forum Message <> Reply to Message

Paul van Delst wrote:

>
> JD Smith wrote:
>>
>> Craig Markwardt wrote:
>>>
>>> Vapuser <vapuser@catspaw.jpl.nasa.gov> writes:
>>>> How about:
>>>>
>>>> version=strupcase(strjoin(strsplit(!version.release,'[.-_]', /extract), ""))
>>>> input_version=strupcase(strjoin(strsplit(desired_release,'[. -_]',/extract), ""))
>>>> if version lt input_version then 'eeek'
>>>>
>>>> Clearly `desired_release' would have to be input as a string.
>>>>
>>>> Of couse, this method 'requires' IDL 5.3 (IIRC), to get the regular
>>>> expression semantics of strsplit. The other question is will RSI use
>>>> any separator in version designations besides these three?
>>>
>>> Umm, I never thought I'd be saying this, but aren't these a bit
>>> overengineered? I have found that in 95% of the cases the DOUBLE
>>> compare will work, and in 4% of the cases the STRING compare will
>>> work.
>>
>> *cough*
>>
>> <REITERATE>
>>
>> IDL> print, double('5.4.1') gt 5.4
>> 0
>>
>> </REITERATE>
>
> wot about
>
> IDL> print, double('5.4.1') ge 5.4d0
> 1
>

Because it's exactly the same! Yes it's ge, but is it gt?

```
IDL>  print, double('5.4.1') gt 5.4d0
  0

IDL>  print, double('5.4.1') eq 5.4d0
  1
```

No it's not, it's eq. Same problem. So use this if you don't care
about the last digit and don't want to be open about it (it's not

exactly obvious this is the case). Use the string compare method
otherwise.

> Doesn't assuage your other concerns regarding the significance of the last digit however.
> I use comparisons like the above for code that contains BREAK, CONTINUE, SWITCH, etc
> statements. Or similar for the version in which pointers and objects were introduced (5.2?
> can't remember).

The problem here is you'll not err cleanly... unknown control statements
will cause compile errors. Not a lot that we can do about this.

JD

Subject: Re: something like perl's 'require 5.4'
Posted by Craig Markwardt on Tue, 17 Apr 2001 21:17:44 GMT
View Forum Message <> Reply to Message

JD Smith <jdsmith@astro.cornell.edu> writes:
>> Umm, I never thought I'd be saying this, but aren't these a bit
>> overengineered? I have found that in 95% of the cases the DOUBLE
>> compare will work, and in 4% of the cases the STRING compare will
>> work.
>
> *cough*
>
> <REITERATE>
>
> IDL> print, double('5.4.1') gt 5.4
> 0
>
> </REITERATE>
>
>
> So I guess what you really you mean that in 95% of cases you don't care
> about the (possible) last digit on the version number. This is fine,
> just say so. Here's a good example of when that last digit matters,
> from the "What's New in IDL 5.2.1":

Right, so use the string compare in this case. I'm easy. Since you
know what version you need, you can code a test that will work in
that case.

IDL> print, '5.2' LT '5.2.1' -> 1
IDL> print, '5.2.1' LT '5.2.1' -> 0
IDL> print, '5.3' LT '5.2.1' -> 0

if !version.release LT '5.2.1' then $

message, 'ERROR: sorry, no histograms for you!'

This approach will have a problem when IDL version 10 comes around
though. :-)

The overengineering part was my gentle chide for using features and
complexity that are more likely to get one into trouble that they are
to solve the problem at hand.

Craig

--
---------- ----------
Craig B. Markwardt, Ph.D. EMAIL: craigmnet@cow.physics.wisc.edu
Astrophysics, IDL, Finance, Derivatives | Remove "net" for better response
---------- ----------

Subject: Re: something like perl's 'require 5.4'
Posted by Stein Vidar Hagfors H[1] on Tue, 17 Apr 2001 21:20:10 GMT
View Forum Message <> Reply to Message

thompson@orpheus.nascom.nasa.gov (William Thompson) writes:
[...]
> I've always found that a simple string comparison works in 100% of the cases.
> Where doesn't it work?

In IDL 10.0 ? :-)

--
Stein Vidar Hagfors Haugan
ESA SOHO SOC/European Space Agency Science Operations Coordinator for SOHO

Subject: Re: something like perl's 'require 5.4'
Posted by Paul van Delst on Tue, 17 Apr 2001 21:41:10 GMT
View Forum Message <> Reply to Message

JD Smith wrote:
>
> Paul van Delst wrote:
>>
>> wot about
>>
>> IDL> print, double('5.4.1') ge 5.4d0
>> 1
>>

>
> Because it's exactly the same! Yes it's ge, but is it gt?
>
> IDL> print, double('5.4.1') gt 5.4d0
> 0
>
> IDL> print, double('5.4.1') eq 5.4d0
> 1
>
> No it's not, it's eq. Same problem. So use this if you don't care
> about the last digit and don't want to be open about it (it's not
> exactly obvious this is the case). Use the string compare method
> otherwise.
>
>> Doesn't assuage your other concerns regarding the significance of the last digit however.
>> I use comparisons like the above for code that contains BREAK, CONTINUE, SWITCH, etc
>> statements. Or similar for the version in which pointers and objects were introduced (5.2?
>> can't remember).
>
> The problem here is you'll not err cleanly... unknown control statements
> will cause compile errors. Not a lot that we can do about this.

Nuh-uh. They're interpreted as user functions/procedures.

```
IDL> $more testit.pro
pro testit
  for i = 0, 10 do begin
    if ( i eq 5 ) then break
  endfor
end
IDL> print, !version
{ mipseb IRIX unix 5.3 Nov 11 1999}
IDL> .run testit
% Compiled module: TESTIT.
IDL> testit
% Attempt to call undefined procedure/function: 'BREAK'.
% Execution halted at:  TESTIT             5 /modishome/paulv/tmp/testit.pro
%                      $MAIN$
```

So:

```
IDL> $more testit.pro
pro testit
  if ( double( !version.release ) lt 5.4d0 ) then begin
    message, 'Need IDL 5.4 to use this procedure', /continue
    return
  endif
```

```
  for i = 0, 10 do begin
    if ( i eq 5 ) then break
  endfor
end
IDL> .run testit
% Compiled module: TESTIT.
IDL> testit
% TESTIT: Need IDL 5.4 to use this procedure
IDL>
```

Pre-5.2, nothing says I can't have a function/array called PTRARR or PTR_NEW (or the OBJ equivalent).

paulv

--
Paul van Delst A little learning is a dangerous thing;
CIMSS @ NOAA/NCEP Drink deep, or taste not the Pierian spring;
Ph: (301)763-8000 x7274 There shallow draughts intoxicate the brain,
Fax:(301)763-8545 And drinking largely sobers us again.
paul.vandelst@noaa.gov Alexander Pope.

Subject: Re: something like perl's 'require 5.4'
Posted by John-David T. Smith on Tue, 17 Apr 2001 21:49:32 GMT
View Forum Message <> Reply to Message

Paul van Delst wrote:
>
> JD Smith wrote:
>>
>> Paul van Delst wrote:
>>>
>>> wot about
>>>
>>> IDL> print, double('5.4.1') ge 5.4d0
>>> 1
>>>
>>
>> Because it's exactly the same! Yes it's ge, but is it gt?
>>
>> IDL> print, double('5.4.1') gt 5.4d0
>> 0
>>
>> IDL> print, double('5.4.1') eq 5.4d0
>> 1
>>

>> No it's not, it's eq. Same problem. So use this if you don't care
>> about the last digit and don't want to be open about it (it's not
>> exactly obvious this is the case). Use the string compare method
>> otherwise.
>>
>>> Doesn't assuage your other concerns regarding the significance of the last digit however.
>>> I use comparisons like the above for code that contains BREAK, CONTINUE, SWITCH, etc
>>> statements. Or similar for the version in which pointers and objects were introduced (5.2?
>>> can't remember).
>>
>> The problem here is you'll not err cleanly... unknown control statements
>> will cause compile errors. Not a lot that we can do about this.
>
> Nuh-uh. They're interpreted as user functions/procedures.

Right. I stand corrected. Just hope they don't have any such named
routines lying about on their path. Dereferencing pointers will cause
compile error in old versions though, right?

JD

Subject: Re: something like perl's 'require 5.4'
Posted by Paul van Delst on Wed, 18 Apr 2001 11:58:39 GMT
View Forum Message <> Reply to Message

JD Smith wrote:
>
> Paul van Delst wrote:
>>
>> JD Smith wrote:
>>>
>>> Paul van Delst wrote:
>>>>
>>>> wot about
>>>>
>>>> IDL> print, double('5.4.1') ge 5.4d0
>>>> 1
>>>>
>>>
>>> Because it's exactly the same! Yes it's ge, but is it gt?
>>>
>>> IDL> print, double('5.4.1') gt 5.4d0
>>> 0
>>>
>>> IDL> print, double('5.4.1') eq 5.4d0
>>> 1
>>>

>>> No it's not, it's eq. Same problem. So use this if you don't care
>>> about the last digit and don't want to be open about it (it's not
>>> exactly obvious this is the case). Use the string compare method
>>> otherwise.
>>>
>>>> Doesn't assuage your other concerns regarding the significance of the last digit however.
>>>> I use comparisons like the above for code that contains BREAK, CONTINUE, SWITCH, etc
>>>> statements. Or similar for the version in which pointers and objects were introduced (5.2?
>>>> can't remember).
>>>
>>> The problem here is you'll not err cleanly... unknown control statements
>>> will cause compile errors. Not a lot that we can do about this.
>>
>> Nuh-uh. They're interpreted as user functions/procedures.
>
> Right. I stand corrected. Just hope they don't have any such named
> routines lying about on their path.

Me too. :o)

> Dereferencing pointers will cause
> compile error in old versions though, right?

I would hope so. But I do have utility and wrapper codes that simply pass pointers along,
i.e. doesn't dereference them.

paulv

--
Paul van Delst A little learning is a dangerous thing;
CIMSS @ NOAA/NCEP Drink deep, or taste not the Pierian spring;
Ph: (301)763-8000 x7274 There shallow draughts intoxicate the brain,
Fax:(301)763-8545 And drinking largely sobers us again.
paul.vandelst@noaa.gov Alexander Pope.