
Subject: Re: Why does IDL strip unary dimensions from structure elements?

Posted by [steinhh](#) on Fri, 27 Oct 1995 07:00:00 GMT

[View Forum Message](#) <> [Reply to Message](#)

In article <308FD223.167E@grossc.gsfc.nasa.gov>, Thomas A McGlynn
<mcglynn@grossc.gsfc.nasa.gov> writes:

The problem with trailing singular array dimensions disappearing:

|> Even this is not guaranteed:

|>

|> a={e:intarr(1,1,1)}

|> help,a,/str yields:

|>

|> ** Structure <40014f08>, 1 tags, length=2, refs=1:

|> E INT Array(1)

|> I.e., it got rid of some of the dimensions but not all. Nor can I use

|> reform to fix a structure element. It's a mess. Has anyone mentioned

|> this to RSI?

|>

I tried to start a discussion on this a couple of months(?) ago, but there wasn't much response. Probably because people seldom come across the problem in everyday use. Once you notice it, though, it's terribly bothersome, adding a great deal of complexity to the otherwise very simple array operations. Since we have a simple way of removing (all) singular dimensions (REFORM), why can't we keep them unless told otherwise!

There's one "good" thing about it: The possibility of indexing everything with an extra zero does compensate a little for the other "feature".

If we have, e.g.,

```
a=intarr(10,1,1)
```

```
help,a
```

```
a      INT      = Array(10)
```

Then the following will not be an error, and it will produce correct results:

```
element = a(10,0,0)
```

So long as you're indexing your array with legal values for the indices (i.e., zero for all singular dimensions), then you get what you expect. It's "just" a matter of not letting your program be confused about it :-)

