How to Propose a Fortran Requirement

Len Moss, X3J3 Requirements Chair

Now that Fortran 90 is an official standard, the Fortran standards committees have begun looking ahead. The future evolution of the language will be managed by an international subcommittee, ISO/IEC JTC1/SC22/WG5, usually known as WG5, with the technical work being done by X3J3, the US member body of WG5. A revision of the language scheduled for 1995 will be primarily devoted to clarifications, corrections and interpretations of Fortran 90, while a more substantial revision is planned for 2000.

WG5 has established an ongoing process for collecting requirements for future Fortran revisions, and will make the final selection of requirements to be addressed by any particular revision. The various national committees, including X3J3, have been asked to solicit proposals for requirements from within their communities as input to WG5.

Both X3J3 and WG5 want to make this process as open as possible and welcome proposals for requirements from any individual or group. On the other hand, most of us who serve on these committees do so on a largely volunteer basis and have only limited time and resources to devote to this effort. To make the process as efficient as possible, and thus to insure that all contributions can receive thoughtful consideration by the committee members, X3J3 has established the following procedures for handling proposed Fortran Requirements, or pFRs, within the US.

What's a pFR?

A pFR is a short document proposing a change to the Fortran standard. It can take a variety of forms, as long as it adheres to the following simple rules.

Describe the problem, not just the solution. People who work with programming languages (the members of X3J3 and WG5 included) tend to be "problem solvers": when they encounter a problem they immediately start thinking of possible solutions. This approach is fine, but if your pFR does not clearly identify the shortcomings of the current standard which your solution is meant to address, it becomes very difficult to evaluate alternative solutions to the same (or similar) problems.

Rule: A pFR must include a "Requirement" section, containing a clear and concise statement of the problem to be solved.

It may be useful to think of a requirement as a set of criteria against which to judge a variety of possible solutions to your problem. (Incidentally, the word "requirement" in this context is not intended to imply any commitment to solve a particular problem, but only a clear statement of what would be required to solve it.)

Explain why this problem should be solved. The Fortran community is extremely large and diverse, and all of us have opinions as to how our language could be "improved". Even after weeding out those with serious flaws, the remaining list of useful and implementable requirements is likely to be far too large to seriously consider adding to the language. Thus you need to explain why it's more important to spend limited resources on this requirement rather than on something else.

Rule: A pFR must include a "Justification" section.

Here are a few questions to think about when writing this section; you don't have to answer all

of them, and you may think of some others. How does this requirement benefit users? Which users does it benefit? Why can't this requirement be satisfied by features already present in Fortran? Why should this requirement be addressed by a Fortran-related standard rather than by individual Fortran implementers or by a standard at some other level (e.g., an operating system standard)? What are the implementation costs of satisfying this requirement? How does this requirement impact existing Fortran codes and processors?

Be brief. Each member of a Fortran standards committee receives and processes a huge volume of information: the papers associated with a typical meeting total between 500 and 1000 pages, and there are usually at least four X3J3 and one WG5 meetings per year. A comparable volume of email is also exchanged between meetings. Given this environment, it is important to make your pFR as clear and concise as possible; a well-written half-page executive summary can be far more effective than a 100 page tome.

> **Rule: The Requirement and Justification sections of a pFR must be no more than about two pages long, and must occur within the first two pages.**

If additional material is truly necessary, please incorporate it in separate, appropriately headed sections (e.g., "Background", "Possible Solution", etc.). If the requirement is complex and requires a lengthy explanation, consider dividing it into several independent requirements, or putting it in a section named something like "Detailed Requirement", and simply putting a summary in the "Requirement" section. A very long requirement section may in fact be a specification of a possible solution rather than a statement of the problem.

Here are some additional suggestions for writing an effective pFR:

- Give your pFR a brief but descriptive title. A good title can help immensely in thinking and talking about a proposal.
- Get someone to criticize your pFR before you submit it, both for clarity and content. This will help you to anticipate and refute possible objections to your proposal (you might even decide not to submit it after all).
- Include your name, phone number and address (email or postal) in the body of your pFR (envelopes and email headers often get lost or mangled).

An example of a pFR is included below, along with a minimal template. The precise format is not important, however, as long as it satisfies the above rules.

Where Do I Send It?

If at all possible, please send your pFR as plain text email to:

ljm@slac.stanford.edu (Internet)
LJM AT SLACVM (BITNET)

If this is not possible, try to send a plain text file on a Macintosh floppy disk to:

Len Moss
SLAC MS 97
P. O. Box 4349
Stanford, CA 94309
(phone: 415-926-3370)

I will do my best to handle proprietary word processor formats and other electronic media, but I cannot

make any promises. As a last resort, you can send your pFR as a paper document to the above address. In this case, however, anything over two pages cannot be accepted.

Please make it clear that you are officially submitting a pFR. This is especially important if you submit your pFR via email, since all of us on the Fortran committees get a lot of casual inquiries and suggestions. A good way to alert me that this is an official submission is to include a subject line beginning with the string "pFR:".

What Happens Next?

When I receive your pFR, I will assign it a number and register it as part of X3J3's standing requirements document, X3J3/9x-004. I will also send you an acknowledgment, including the assigned pFR number.

At the next X3J3 meeting, your pFR will be assigned to a subgroup for review. Unfortunately, we do not have the resources to keep you personally informed of the subsequent fate of your proposal. However, you may follow its progress yourself in the minutes of our meetings. To become an observer and automatically receive copies of X3J3's minutes and meeting announcements, write to:

X3J3 Secretariat, CBEMA
1250 Eye Street NW, Suite 200
Washington, DC 20025

There is a yearly fee of approximately $300 for observers.

If you have ftp access to the Internet, a number of X3J3 documents, including meeting minutes and the requirements document, can be obtained via anonymous ftp from the "/x3j3" directory at ftp.ncsa.uiuc.edu. This directory can also be accessed via an email archive server. To find out how to use it, send email to archive-server@ncsa.uiuc.edu with the command "help" on a single line in the body of the message. You must have a domain-style return email path (e.g., name@site.bitnet, name@site.edu, name@site.UUCP). If this email path cannot be readily deduced from the header of your email, include the command "path <email-path>" on a separate line in the body of the message.

For information about becoming a member of X3J3, please call or write the committee chair:

Jerrold L. Wagener
Amoco Production Research
P. O. Box 3385
Tulsa, OK 74102
(phone: 918-660-3978)
(email: jwagener@amoco.com)

Example

TITLE: Controlling Pointer Bounds

SUBMITTED BY: Jeanne Martin (phone: ...; email: ...)

REFERENCES: ISO/IEC JTC1/SC22/WG5-N780

REQUIREMENT: Allow a user to specify the bounds of a pointer that is associated with a target by the pointer assignment statement.

JUSTIFICATION: Currently a pointer that is associated with a target by the pointer assignment statement has the bounds of the target unless the target is an array section or an array expression, in which case the lower bound is one and the upper bound is the extent. The user has no control over setting the bounds. This is inconsistent with the passing of an actual argument array, array section, or array expression to a dummy argument. The lower bound of a dummy argument may be specified in a declaration statement. A user should have a similar amount of control over the bounds of array pointers.

ESTIMATED IMPACT: This affects only the pointer facility. It will not invalidate any existing programs.

Template

TITLE:

SUBMITTED BY:

REQUIREMENT:

JUSTIFICATION:
