Compiler Support for the Fortran 2003, 2008, TS29113, and 2018 Standards Revision 26

Ian D Chivers & Jane Sleightholme
Ian Chivers: Rhymney Consulting, London.
Jane Sleightholme: FortranPlus, London.
ian@rhymneyconsulting.co.uk
jane@fortranplus.co.uk

3 Introduction

This is a repeating article in Fortran Forum. The first version appeared in Fortran Forum in April 2007. It has been revised on a regular basis since.

3.1 Original version

The basis for the entries in the original list of features was a report by John Reid. The document number is N1648. An electronic version can be found at:

`https://wg5-fortran.org/N1601-N1650/N1648.pdf`

If you are a compiler vendor and would like to be included in future versions of this table please email one of us with details and they will be added to the table and published in Fortran Forum.

3.2 Acknowledgements for the original article

An email was sent to the J3 list asking for information about compiler support for the new features of the Fortran 2003 standard. The following people contributed to the original article:

- Bill Long, Cray
- Joost VandeVondele
- Van Snyder
- Tobias Burnus and Brooks Moses, gfortran
- Andy Vaught, g95
- Robert Holmes, NAG

Thanks.

First appeared April 2007.

3.3 Revision 1 - August 2007

Two new compiler vendors were added. The information on the Intel compiler has been taken from the release notes that came with release 10 of the compiler. The information on the IBM entry has been taken from their web site. Ian Bush posted an article to comp.lang.fortran regarding this release (IBM XL Fortran Enterprise Edition for AIX, V11.1). Thanks Ian. See

`http://publib.boulder.ibm.com/infocenter/comphelp/v9v111/index.jsp?topic=/com.ibm.xlf111.aix.doc/getstart/new_features.htm`

for more information.

Intel and IBM were contacted to ask them to verify the information.

- Jim Xia of IBM corrected their entry, thanks Jim.
- Stan Whitlock of Intel corrected their entry, thanks Stan.

If there are any errors please notify us and we will correct them in the next version of this article.

3.4 Revision 2 - August 2008

Sun has been added and there have been a few corrections and updates to some of the other entries.

- Michael Ingrassia of Sun corrected their entry, thanks Michael.

3.5 Revision 3 - April 2009

The entries for Cray, gfortran (11 changes) , Intel (18 changes) and NAG (9 changes) have been updated.

3.6 Revision 4 - August 2009

We've had replies from Cray (Bill Long) , gfortran (Tobias Burnus), g95 (Andy Vaught), Intel (Stan Whitlock), Nag (Malcolm Cohen) and Sun (Robert Corbett).

We've also added two entries suggested by Richard Maine. Here is the text of the message we received from Richard.

> I just got the latest Fortran Forum and noticed two somewhat related Fortran 2003 features that I personally think are important, but aren't reflected in your table of features. If convenient, they might be useful to add to the table.
>
> 1. Allocatable scalars. To me, this is an important feature for object orientation, and in particular for polymorphism. Basically, a polymorphic object has to be either a pointer or an allocatable (or a dummy argument, which is a bit restrictive). In my experiments with polymorphism, the polymorphic objects pretty much always naturally "wanted" to be allocatable. But the NAG compiler (which I was using at the time) didn't yet support allocatable scalars. This meant that I either needed to make all the polymorphic objects pointers or make them arrays (possibly of size 1). Neither of these alternatives was attractive at all. I found this a significant enough shortcoming to keep me from using the polymorphic features. Thus, I'd think this would be something people would want to know about a compiler if they planned to use polymorphism.
>
> 2. Allocatable character length. I think that allocatable character length is one of the biggest "sleeper" features of f2003. It wasn't even on the list of f2003 requirements, and thus sometimes doesn't show up in lists of new features. It just naturally arise from allocatable length parameters for parameterized derived types. It seemed like one should allow the same thing for the one intrinsic type with a length type parameter. And lo, when it was all put together, it seemed like this was finally a good way to do variable length strings in Fortran. It integrates with the rest of the language immensely better than iso_varying_string has any hope of doing. In fact, as I said, it integrates so well that it came about as a consequence of the integration of other features. Allocatable-length character strings act like so many people intuitively think of character strings, unlike the fixed-length character strings that we've had since f77.
>
> Although this is related to allocatable scalars, in that you certainly want to be able to have allocatable character strings that are scalar, it is also a separate feature in that you can have allocatable scalars without necessarily allowing character length to be allocatable. It is also different in application, in that I see the main other usage of allocatable scalars as being for polymorphism, whereas allocatable character strings are not much related to polymorphism. It is also useful independent of parameterized derived types. I personally expect to see allocatable character strings used far more than parameterized derived types, even though it was the requirement for parameterized derived types that lead to allocatable character strings. I could almost see allocatable character strings as becoming the "normal" way that most character string variables are done.

Thanks Richard.

We've also added entries for the Fortran 2008 standard. The entries are based on document N1729, which can be found at

`https://wg5-fortran.org/N1701-N1750/N1729.pdf`

The last change are entries for compilers that support the Fortran 95 standard, and a list of compilers that are no longer under development but did support Fortran 90, and finally compilers which are available but we have no information on.

We have included the above for completeness. Given the widely differing levels of compiler standard conformance today we wanted to make this information available to people choosing a compiler.

Thanks to everyone who has provided the data.

3.7 Revision 5 - August 2010

The IBM entry has been updated. See

`http://www-01.ibm.com/common/ssi/rep_ca/3/897/ENUS210-103/ENUS210-103.PDF`

The entry for gfortran has been updated. There is an entry for HP. The entry refers to the March 2010 release.

John Reid has also updated N1729.pdf and the latest version (N1828) can be found at

`https://wg5-fortran.org/N1801-N1850/N1828.pdf`

3.8 Revision 6 - December 2010

The gfortran entry has been updated. Here is part of an email we received from Tobias Burnus.

- Hi, as the development of GCC has almost reached the end of Stage 1, I thought I could already update the F2003/F2008 conformance status for the December issue of ACM Fortran Forum. (Stage 1 allows for larger changes; it is followed by Stage 3 (!) which allows only smaller bug fixes, regression fixes and documentation updates.) Past experience suggests that 4.6.0 will be released next March as it won't be ready before Christmas - and it takes a while to fix the new issues reported during the Christmas break. (It is really a break as most developers are paid for GCC work ©, Ada, middle-end, target parts) and take off - only gfortran is purely developed in the spare time.).....gfortran 4.6 will presumably also allow to use REAL(16) (128 bit floating-point numbers) on x86, x86-64, and ia64 systems, which are emulated in software; so far only the real kinds 4, 8 and 10 (80bit FP) were supported on those systems. (This library inclusion had to be approved by the Free Software Foundation - but that problem seems to be mostly solved.)

For a complete list of what's new in 4.6 visit:

`http://gcc.gnu.org/gcc-4.6/changes.html`

Thanks Tobias.

3.9 Revision 7 - August 2011

The main driving force for the changes in this revision was an email from Stan Whitlock at Intel. Here is an extract from Stan's email.

> I hope you will be updating the F2008 features list based on John Reid's updated article: ISO/N1828 - Features of F2008 - John Reid - latest adds features.

The feature list for Fortran 2008 used in the original article were taken from John Reid's earlier paper, N1729

`https://wg5-fortran.org/N1701-N1750/N1729.pdf`

The entries in the table are now taken from the contents of the N1828.pdf document. This is available at

`https://wg5-fortran.org/N1801-N1850/N1828.pdf`

We have also added entries for two more Fortran compiler companies, Absoft (thanks Wood Lotz) and PGI (thanks Pat Brooks, Dave Norton and Brent Leback)

3.10 Revision 8 - December 2011

Corrections from Stan Whitlock

> I have attached an updated spreadsheet with F2003 and F2008 tabs for Intel Fortran 12.1. There is little change over 12.0 but there are several typos. New text is in blue; editing directions are in red.

Damian's suggestion to add "generic procedure interfaces named the same as a type in the same module" as yet another F2003 feature is acceptable. I included that in the attached.

Thanks Stan.

Corrections from Malcolm Cohen. Here is the email from Malcolm.

You are probably aware of these, but anyway…

In the F2008 table, the subheading "Input/Output" has a spurious "N" in the PGI column.

Near the end of the table, there are two entries

"Null pointer or unallocated allocatable as an absent dummy argument"

and

"Null pointer as a missing dummy argument"; these would appear to be the same feature.

Also, the "generic resolution" line says "by pointer or allocatable attribute", it is missing any mention of "data object vs. procedure". (I think there is an argument for these two different extensions to generic resolution being mentioned separately.)

Thanks Malcolm, these have now been incorporated into the table.

The following was received from Tobias Burnus.

Dear Ian and Jane, dear all,

first, thanks for providing the helpful list of supported F2003/F2008 features. Unfortunately, coding for Fortran 2003 still requires such a list.

Whitlock, Stan wrote:

> I have attached an updated spreadsheet with F2003 and F2008 tabs for Intel Fortran 12.1. [...] Damian's suggestion to add "generic procedure interfaces named the same as a type in the same module" as yet another F2003 feature is acceptable. I included that in the attached.

Attached you find the modifications for gfortran; I also included an answer to Damian's item (unfortunately: "No" as of now).

I added additional items for the DTR 29113, which might a bit premature.

Most changes apply to GCC 4.6 (released: March 2011) *and* 4.7 (developer version, but builds widely used). Only one F2008 (all constants) and two DTR29113 items are better supported in 4.7 and marked such. (GCC 4.7 will be released around March/April 2012.)

Thanks Tobias.

The following was received from Pat Brooks

Greetings Ian and Greetings Jane,

Updated spreadsheet from PGI attached. Regarding Damian's request to add a item for generic procedure interfaces named the same as a type in the same module, yes please include it, and yes the PGI compiler supports this feature. You may want to verify that with Damian and please let us know if you hear any comments to the contrary.

Thanks and Best Regards, Pat Brooks

Thanks Pat, these have been incorporated.

Updates have also been received from Bill Long at Cray. Thanks Bill.

3.11 Revision 9 - April 2012

We received an email from Rafik Zurob at IBM in January and have corrected the IBM entry. To clarify the comment Rafik raised about compiler version number this is in the 2003 table and 2008 table as people wanted to know which versions of the compilers were being referred to in the tables, and is not a reference to the Fortran 2008 feature. Malcolm Cohen also spotted a couple of mistakes. Thanks to you both.

3.12 Revision 10 - August 2012

We have received a couple of emails regarding the table.

- Hi:, I was looking at your joint article in the latest issue of Fortran Forum. Regarding your table of features: It is fairly obvious that: Y=yes, N=no, but what on earth is "P"? Perhaps question Thank you. Mike Milgram

We have corrected this issue. Thanks Mike.

- Hi Ian, I only spotted 2 things that need changing, as per attached. Malcolm Cohen.

Corrected - thanks Malcolm.

- Hi Ian and Jane - We announced the availability of XL Fortran V14.1 today. This release adds support for several Fortran 2008 features. Please find the updated spreadsheet attached. (In the F2003 tab, I added 13.1 to the compiler version since that's the first version with full Fortran 2003 compliance. I've updated the F2008 tab with the information for the 14.1 compiler.) Also, I noticed that the spreadsheet (and N1828.pdf) do not list support for the ERROR STOP statement. Should that be a separate question Regards Rafik.

Thanks Rafik.

- Greetings Ian, Please find attached an updated spreadsheet for PGI. Also, as we mentioned last time, PGI supports Damian's suggestion for "Generic procedure interfaces named the same as a type in the same module". In the last report, our comments to that effect were published but the table wasn't updated. We'd appreciate it if you could please double check that for us. Let us know if we can provide any further clarification. Thanks and Best Regards, Pat Brooks

Thanks Pat.

3.13 Revision 11 - December 2012

We received emails from a variety of people including Wood Lotz at Absoft and Bill Long at Cray. The Cray compiler now supports the Fortran 2008 standard. Thanks to the contributors. The following document is being worked on at the moment.

- [1891 The new features of Fortran 2008 (Reid) - supersedes N1828]

Visit

`ftp://ftp.nag.co.uk/sc22wg5/N1851-N1900/N1891.pdf`

for up to date information.

3.14 Revision 12 - April 2013

Minor corrections.

3.15 Revision 13 - August 2013

There are updates to several of the compilers. Thanks to Tobias Burnus, Wood Lotz, Polyhedron Software and Intel for providing the information.

The next revision will include an entry for the features introduced with ISO/IEC TS 29113:2012 which was published in 2012. Available at

`https://wg5-fortran.org/N1851-N1900/N1942.pdf`

Tobias Burnus notified us about the document and an article is being prepared by Reinhold Bader which will form the basis for the entries in the table. Thanks to Tobias and Reinhold.

3.16 Revision 14 - December 2013

The major update in this edition is the inclusion of details of TS29113 and the original idea and request came from Tobias Burnus. The content of the table has been based on interchanges that took place between several people including Reinhold Bader, Bill Long and Malcolm Cohen. This edition of Fortran Forum also includes a paper by Reinhold Bader on this TS.

A draft can be found at:

`https://wg5-fortran.org/N1851-N1900/N1942.pdf`

The full TS can be bought at

`http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=45136`

The Absoft, Intel, Nag and PGI entries have also been updated.

3.17 Revision 15 - August 2014

John Reid has produced paper N1891 (March 2014) as an update to paper N1828 (May 2010), which is included in modified form in this edition of Fortran Forum. The table on Fortran 2008 conformance has been modified to reflect these changes. Additional changes have been made to the headings of this table in response comments made by-Malcolm Cohen of NAG and Rafik Zurob of IBM. The Cray, IBM, Intel, Nag, Oracle and PG entries have been updated in response information supplied by these vendors.

Bob Corbett has pointed out that the table of Fortran 2003 features includes ISO TR 15581 Allocatable Enhancements, and that the Fortran 2003 specification of these enhancements is incompatible with the specification in the TR. Some current Fortran implementations implement the old, non conforming semantics.

Thanks to everyone for their contributions.

3.18 Revision 16 - June 2015

Thanks to Stan Whitlock, Bill Long and Mark Leair for the updates, corrections and comments.

3.19 Revision 17 - November 2015

Paul Richard Thomas provided the updates for the gfortran entry to the 5.2 release. Thanks very much to Paul and all of the gfortran team.

Added summary information for each of the following tables

- Fortran 2003 features and conformance,
- Fortran 2008 features and conformance
- TS29113 features and conformance

providing totals by compiler vendor and feature.

3.20 Revision 18 - January 2016

This revision is a set of corrections, and additional explanation about some of the tables. Thanks to Stan Whitlock at Intel for the detailed notes.

3.21 Revision 19 - June 2016

Updated the PGI entry to reflect release 16.4 of the compiler, and corrected some of the entries. Thanks to Mark Leair for the updates and corrections.

Corrected the summary by vendor and summary by feature figures generated by Excel. Thanks to Stan Whitlock for pointing this out.

Updated the Pathscale entry. Thanks to Keith Refson (Royal Holloway College, University of London) and Chris Bergstrom (CEO Pathscale) for the information.

Updated the IBM entry. Thanks to Daniel Chen, IBM.

3.22 Revision 20 - November 2016

Updated the Nag entry to reflect the 6.1 release. The 6.1 update had been omitted from revision 19 by mistake. Thanks to Malcolm Cohen for pointing this out.

Updated the gfortran entry. Email from Paul Richard Thomas [paul.richard.thomas@gmail.com]. Now all Fortran 2003 except PDTs.

Updated the Intel entry. Thanks to Stan Whitlock at Intel.

Corrections to summary table figures.

3.23 Revision 21 - February 2017

Corrected and updated the gfortran entry. Release 6 fully supports submodules and release 7 (current development version) fully supports DTIO.

3.24 Revision 22 - November 2017

The following major changes have been made

- Removed the g95, HP, Pathscale entries;
- Added the Fujitsu entry (thanks to Minoru Tanaka and Yuuji Tsujimori);
- Updated the IBM (thanks to Daniel Chen) , Intel (thanks to Jon Steidel and Lorri Menard), and Oracle (thanks to Calvin Vu) entries;
- split and moved the summary tables;
- made available previous versions of the tables on our web site;
- Annex C Fortran, 2015 standard - Features that were new in Fortran 2008 but not originally listed in its introduction as being new features have been included in the Fortran 2008 table. There are three features in the Annex that had already been added to the base table. These are duplicates of 5.10.1, 5.10.2 and 6.2 from the base table. These new features had already been added by John Reid's paper ISO/N1828 - Features of F2008 - latest added features.

3.25 Revision 23 - April 2018

The following major changes have been made.

- Added the Arm entry. Thanks to Kiran Chandramohan
- Updated the Fujitsu entry. Thanks to Minoru Tanaka.

3.26 Revision 24 - August 2018

The following changes have been made

- Added the NEC entry. Thanks to Yasuharu Hayashi.
- Reformatted the Fortran 2008 table to make typesetting easier.

3.27 Revision 25 - December 2018

The following changes have been made

- Circulated a conformance table for the Fortran 2018 standard to the vendors, based on John Reid's paper. A version of this paper was published in the April 2018 edition of Fortran Forum, and a subsequent revision (August 2018), N2161, can be found on line at `https://wg5-fortran.org/documents.html`
- Updated all urls which referenced the Nag ftp server to point to the new WG5 server;
- Corrected the Nag entry. The 6.2 release fully supports the Fortran 2003 standard.

- Arm have released a new version of their compiler, thanks to Caroline Concatto for the updates. The compiler now supports submodules, do concurrent and internal procedures as actual arguments.
- Intel have released a new version of their compiler, thanks to Jon Steidel for the updates. This releases supports all of F2008.
- PGI have released a new version of their compiler, thanks to Gary Klimowicz for the updates. The majority of the improvements come in Fortran 2008 support, including submodules and do concurrent.

3.28 Revision 26 - August 2019

The following changes have been made

- gfortran Fortran 2008 entries have been updated.

Here is the main Fortran 2003 compliance table.

Fortran 2003 Features	Absoft	Arm	Cray	Fujitsu	gfortran	IBM	Intel	NAG	NEC	Oracle	PGI
Compiler version number	14	19.0	8.4.0	2.0.3	9.x	15.1.5	19.0	6.2	1.2.0	8.8	18.10
ISO TR 15580 IEEE Arithmetic	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
ISO TR 15581 Allocatable Enhancements	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y, 33	Y
Data enhancements and object orientation											
Parameterized derived types	N	Y	Y	Y	Y, 36	Y	Y	Y	Y	Y	Y
Procedure pointers	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Finalization	N	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Procedures bound by name to a type	N	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
The PASS attribute	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Procedures bound to a type as operators	N	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Type extension	N	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Overriding a type-bound procedure	N	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Enumerations	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
ASSOCIATE construct	N	Y	Y	Y	Y	Y	Y	Y	Y	N	Y

Fortran 2003 Features	Absoft	Arm	Cray	Fujitsu	gfortran	IBM	Intel	NAG	NEC	Oracle	PGI
Compiler version number	14	19.0	8.4.0	2.0.3	9.x	15.1.5	19.0	6.2	1.2.0	8.8	18.10
Polymorphic entities	N	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
SELECT TYPE construct	N	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Deferred bindings and abstract types	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Allocatable scalars, 12		Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Allocatable character length, 12		Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Miscellaneous enhancements											
Structure constructors	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Generic procedure interfaces named the same as a type in the same module		Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
The allocate statement	P	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Assignment to an allocatable array	N	Y,2	Y, 3	Y,2	Y	Y	Y, 2	Y	Y	Y	Y,3
Transferring an allocation, 18	N	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
More control of access from a module	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Renaming operators on the USE statement	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Pointer assignment	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Pointer INTENT	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
The VOLATILE attribute	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
The IMPORT statement	Y	Y	Y	Y	Y	Y	Y	Y	YY	Y	Y
Intrinsic modules	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Access to the computing environment	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y

Fortran 2003 Features	Absoft	Arm	Cray	Fujitsu	gfortran	IBM	Intel	NAG	NEC	Oracle	PGI
Compiler version number	14	19.0	8.4.0	2.0.3	9.x	15.1.5	19.0	6.2	1.2.0	8.8	18.10
Support for international character sets	P, 19	P,19	P, 19	P,19	Y	P	P, 19	Y	Y	N	P,19
Lengths of names and statements	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Binary, octal and hex constants	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Array constructor syntax	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Specification and initialization expressions	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Complex constants	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Changes to intrinsic functions	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Controlling IEEE underflow	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Another IEEE class value	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Input/output enhancements											
Derived type i/o	N	Y	Y	Y	Y	Y	Y	Y	N	N	Y
Asynchronous input/output	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y, 10	Y
FLUSH statement	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
IOMSG= specifier	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Stream access input/output	N	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
ROUND= specifier	Y	Y	Y	Y	Y	Y	Y, 20	Y	Y	Y	Y
DECIMAL= specifier	Y	Y	Y	Y	Y	Y	Y, 22	Y	Y	Y	Y
SIGN= specifier	Y	Y	Y	Y	Y	Y	Y, 21	Y	Y	Y	Y

Fortran 2003 Features	Absoft	Arm	Cray	Fujitsu	gfortran	IBM	Intel	NAG	NEC	Oracle	PGI
Compiler version number	14	19.0	8.4.0	2.0.3	9.x	15.1.5	19.0	6.2	1.2.0	8.8	18.10
Kind type parameters of integer specifiers	N	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Recursive input/output	N	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Intrinsic function for newline character	N	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Input and output of IEEE exceptional values	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Comma after a P edit descriptor	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Interoperability with C											
Interoperability of intrinsic types	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Interoperability with C pointers	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Interoperability of derived types	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Interoperability of variables	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Interoperability of procedures	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Interoperability of global data	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y

Notes

Y	Yes
N	No
P	Partial
	Not known
2	Optional under a flag.
3	Can be disabled by an optional flag.
9	kind= of maxloc, minloc, shape missing
10	implemented as synchronous i/o
12	Suggested by Richard Maine
18	MOVE_ALLOC

19	SELECTED_CHAR_KIND only
20	plus RC,RD,RN,RP,RU,RZ
21	plus BLANK=,DELIM=,PAD=,SIZE=
22	plus DC,DP
30	only for output
31	Partial in the 4.9 development version
33	Implemented as specified in the Fortran 2003 standard, not as in ISO TR 15581
34	Only kind type parameters
36	Release 8, current development version.

Here is the summary table by vendor.

F2003	Absoft	Arm	Cray	Fujitsu	gfortran	IBM	Intel	NAG	NEC	Oracle	PGI
	14	19.0	8.4.0	2.0.3	9.x	15.1.5	19.0	6.2	1.2.0	8.8	18.10
Y	37	56	55	56	57	57	53	58	57	53	56
Y with notes	0	1	1	1	1	0	4	0	0	2	1
N	16	0	0	0	0	0	0	0	1	3	0
N with notes	0	0	0	0	0	0	0	0	0	0	0
P	1	0	0	0	0	1	0	0	0	0	0
P with notes	1	1	1	1	0	0	1	0	0	0	0
No information	3	0	1	0	0	0	0	0	0	0	1
	58	58	58	58	58	58	58	58	58	58	58
	Absoft	Arm	Cray	Fujitsu	gfortran	IBM	Intel	NAG	NEC	Oracle	PGI

The summary table by feature is available on our web site.

Here is the Fortran 2008 compliance table, with the additional features taken from Annex C of the Fortran 2018 standard. We have split the table in two to ease typesetting. The first table has a description of the feature. We have added an artificial feature number of make linking the tables easier. The second table summarises implementation status.

Feature number	Fortran 2008 Features	2018 number	2008 number	Notes

Feature number	Fortran 2008 Features	2018 number	2008 number	Notes
1	Submodules		2	
2	Coarrays		3	
	Performance enhancements		4	
3	do concurrent		4.1	
4	Contiguous attribute		4.2	
	Data Declaration		5	
5	Maximum rank + corank <=15		5.1	
6	Long integers (18 digit or 64 bit)		5.2	
7	Allocatable components of recursive type		5.3	
8	Implied-shape arrays		5.4	
9	Pointer initialization		5.5	
10	Data statement restrictions lifted		5.6	
11	Kind of a forall index		5.7	
12	Type statement for intrinsic types TYPE (intrinsic type) specifier		5.8	
13	Declaring type-bound procedures		5.9	
14	value attribute is permitted for any nonallocatable nonpointer noncoarray		5.10.1	1
15	in a pure procedure the intent of an argument need not be specified if it has the value attribute		5.10.2	1
	Data Usage		6	
16	Simply contiguous arrays rank remapping to rank>1 target		4.3	3
17	Omitting an allocatable component in a structure constructor		6.1	
18	Multiple allocations with source=		6.2	1
19	Copying the properties of an object in an allocate statement		6.3	1
20	MOLD= specifier for ALLOCATE		6.3	2
21	copying the bounds of a source array in an allocate statement		6.3	3
22	Polymorphic assignment		6.4	
23	Accessing real and imaginary parts		6.5	

Feature number	Fortran 2008 Features	2018 number	2008 number	Notes
24	Pointer function reference is a variable		6.6	2
25	Elemental dummy argument restrictions lifted		6.7	
	Input/Output		7	
26	Finding a unit when opening a file		7.1	
27	g0 edit descriptor		7.2	
28	Unlimited format item		7.3	
29	Recursive i/o		7.4	
	Execution control		8	
30	The block construct		8.1	
31	Exit statement		8.2	
32	Stop code		8.3	
33	ERROR STOP		8.4	2
	Intrinsic procedures and modules		9	
	Bit processing		9.1	
34	Bit sequence comparison		9.1-1	
35	Combined shifting		9.1-2	
36	Counting bits		9.1-3	
37	Masking bits		9.1-4	
38	Shifting bits		9.1-5	
39	Merging bits		9.1-6	
40	Bit transformat ional functions		9.1.7	
41	Storage size		9.2	
42	Optional argument radix added to selected real kind		9.3	2
43	Extensions to trigonometric and hyperbolic intrinsic functions		9.4	
44	Bessel functions		9.5	
45	Error and gamma functions		9.6	
46	Euclidean vector norms		9.7	

Feature number	Fortran 2008 Features	2018 number	2008 number	Notes
47	Parity		9.8	
48	Execute command line		9.9	
49	Optional back argument added to maxloc and minloc		9.1	
50	Find location in an array		9.1.1	
51	String comparison		9.1.2	
52	Constants		9.1.3	
53	COMPILER_VERSION		9.1.4	4
54	COMPILER_OPTIONS		9.1.4	4
55	Function for C sizeof		9.1.5	
56	Added optional argument for ieee_selected _real_kind		9.1.6	1
	Programs and procedures		10	
57	Save attribute for module and submodule data		10.1	
58	Empty contains section		10.2	
59	Form of the end statement for an internal or module procedure		10.3	
60	Internal procedure as an actual argument or pointer target		10.4	
61	Null pointer or unallocated allocatable as an absent dummy argument		10.5	
62	Non pointer actual for pointer dummy argument		10.6	
63	generic resolution by procedureness		10.7.1	2
64	Generic resolution by pointer versus allocatable		10.7.2	2
65	Impure elemental procedures		10.8	
66	Entry statement becomes obsolescent		10.9	
	Source form		11	
67	Semicolon at line start		11.1	
	Annex C Fortran 2018 standard			
	The following features were new in Fortran 2008 but not originally listed in its introduction as being new features:			

Feature number	Fortran 2008 Features	2018 number	2008 number	Notes
14	An array or object with a nonconstant length type parameter can have the VALUE attribute.	1	5.10.1	D
18	Multiple allocations are permitted in a single ALLOCATE statement with the SOURCE= specifier.	2	6.2	D
68	A PROCEDURE statement can have a double colon before the first procedure name.	3		
15	An argument to a pure procedure can have default INTENT if it has the VALUE attribute.	4	5.10.2	D
69	The PROTECTED attribute can be specified by the procedure declaration statement.	5		
70	A defined-operator can be used in a specification expression.	6		
71	All transformational functions from the intrinsic module IEEE_ARITHMETIC can be used in constant expressions.	7.1		
72	All transformational functions from the intrinsic module IEEE_EXCEPTIONS can be used in constant expressions.	7.2		
73	All transformational functions from the intrinsic module IEEE_ARITHMETIC can be used in specification expressions.	7.3		
74	All transformational functions from the intrinsic module IEEE_EXCEPTIONS can be used in specification expressions.	7.4		
75	All transformational functions from the intrinsic module ISO_C_BINDING can be used in specification expressions.	7.5		
76	Arguments to C_LOC in the intrinsic module ISO_C_BINDING, see note 400 below.	8		
77	The name of an external procedure that has a binding label is a local identifier and not a global identifier.	9		
78	A procedure that is not a procedure pointer can be an actual argument that corresponds a procedure pointer dummy argument with the INTENT (IN) attribute.	10		
79	An interface body for an external procedure that does not exist in a program can be used to specify an explicit specific interface.	11		
	Already added to the base table earlier			

Feature number	Fortran 2008 Features	2018 number	2008 number	Notes
14	An array or object with a nonconstant length type parameter can have the VALUE attribute.	1	5.10.1	D
18	Multiple allocations are permitted in a single ALLOCATE statement with the SOURCE= specifier.	2	6.2	D
15	An argument to a pure procedure can have default INTENT if it has the VALUE attribute.	4	5.10.2	D

Here is the implementation summary.

Feature number	Absoft	Arm	Cray	Fujitsu	gfortran	IBM	Intel	Nag	NEC	Oracle	PGI
	14	19.1	8.4.0	2.0.3	9.x	15.1.5	19.0	6.2	1.2.0	8.8	18.10
1	N	Y	Y	N	Y, 201	Y	Y	N	N	N	Y
2	N	N	Y	P	Y, 301	N	Y	N	N	N	N
3	N	3	Y	Y	P, 700	Y	Y	Y	N	N	Y
4	N	Y	Y	Y	Y	Y	Y	Y	Y	N	Y
5	N	N	Y	P	Y, 701	Y	Y	Y	Y	N	N
6	Y, 100	Y	Y	Y, 100	Y	Y	Y	Y	Y	Y	Y
7	N	N	Y	N	N	N	Y	N	N	Y	N
8	N	N	Y	Y	Y	Y	Y	Y	Y	N	N
9	N	N	Y	N	Y	N	Y	N	N	N	Y,600
10		N	Y	Y	N	N	Y	N	N	N	N
11	N	N	Y	N	N	Y	Y	Y	Y	N	N
12		N	Y	N	Y	Y	Y	Y	Y	N	N
13		N	Y	Y	Y	Y	Y	Y	Y	N	N
14		N	Y	N	Y	P 108	Y		N	N	N

Feature number	Absoft	Arm	Cray	Fujitsu	gfortran	IBM	Intel	Nag	NEC	Oracle	PGI
	14	19.1	8.4.0	2.0.3	9.x	15.1.5	19.0	6.2	1.2.0	8.8	18.10
15		Y	Y	N	Y	Y	Y	Y	Y	N	N
16	N	Y	Y	Y	Y	Y	Y	Y	Y	N	Y
17		N	Y	N	N	N	Y	Y	Y	N	N
18		N	Y	N	Y	Y	Y	N	N	N	N
19		Y	Y	Y	N, 201	Y	Y	Y	Y	N	Y
20		Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
21		Y	Y	Y	N, 210	Y	Y	Y	Y	Y	Y
22	N	N	Y	N	P, 210	Y	Y	Y	Y	Y	N
23	N	P 500	Y	N	N	Y	Y	Y	Y	N	Y
24	N	N	Y	N	P, 201	N	Y	N	N	N	N
25		Y	Y	N	Y	N	Y	N	N	N	N
26	N	Y	Y	N	Y	Y	Y	Y	Y	N	Y
27	N	N	Y	N	Y	N	Y	Y	Y	N	N
28	N	Y	Y	N	Y	N	Y	Y	Y	N	N
29	N	Y	Y	N	Y	N	Y	Y	N	Y	Y
30	N	N	Y	Y	Y	Y	Y	Y	Y	N	N
31	N	N	Y	N	Y	Y	Y	Y	Y	N	Y
32	N	Y	Y	Y	Y	Y	Y	Y	Y	N	Y
33		N		Y	Y	Y	Y		N		N
34	Y	N	Y	Y	Y	N	Y	Y	Y	N	N

Feature number	Absoft	Arm	Cray	Fujitsu	gfortran	IBM	Intel	Nag	NEC	Oracle	PGI
	14	19.1	8.4.0	2.0.3	9.x	15.1.5	19.0	6.2	1.2.0	8.8	18.10
35	Y	N	Y	Y	Y	Y	Y	Y	Y	N	N
36	Y	P 214	Y	Y	Y	Y	Y	Y	Y	N	P, 214
37	N	N	Y	Y	Y	Y	Y	Y	Y	N	N
38	Y	N	Y	Y	Y	Y	Y	Y	Y	N	N
39	Y	N	Y	Y	Y	Y	Y	Y	Y	N	N
40	N	N	Y	N	Y	N	Y	Y	Y	N	N
41	N	Y	Y	Y	Y	Y	Y	Y	Y	N	Y
42	N	N	Y	N	Y	Y	Y	Y	Y	N	N
43	Y	P 501	Y	Y	Y	Y	Y	Y	Y	N	Y
44	Y	Y	Y	N	Y	N	Y	Y	Y	N	Y
45	Y	Y	Y	P	Y	Y	Y	Y	Y	N	Y
46	N	N	Y	N	Y	N	Y	Y	Y	N	Y
47	N	N	Y	N	Y	N	Y	Y	Y	Y	N
48	Y	N	Y	N	Y	Y	Y	Y	Y	N	N
49	N	N	Y	N	N	Y	Y	N	N	N	N
50		Y	Y	N	N	Y	Y	N	N	N	Y
51		Y	Y	Y	Y	N	Y	Y	Y	Y	Y
52	N	Y	Y	P	P, 203	Y	Y	Y	Y	N	Y
53	N	Y	Y	Y	N	Y	Y	Y	Y	N	Y
54	N	N	Y	Y	N	Y	Y	Y	Y	N	Y
55		Y	Y	Y	Y	Y	Y	N	N	N	Y
56		N	Y	N	Y	Y	Y	Y	Y	N	Y
57		N	Y	Y	Y	Y	Y	Y	Y	Y	Y
58	Y	P, 502	Y	N	Y	Y	Y	Y	Y	N	N
59		Y	Y	N	Y	Y	Y	Y	Y	N	Y

Feature number	Absoft	Arm	Cray	Fujitsu	gfortran	IBM	Intel	Nag	NEC	Oracle	PGI
	14	19.1	8.4.0	2.0.3	9.x	15.1.5	19.0	6.2	1.2.0	8.8	18.10
60	N	Y	Y	N	Y	Y	Y	Y	Y	N	Y
61	N	P, 503	Y	Y	Y	Y	Y	Y	Y	N	Y
62		Y	Y	N	N		Y	Y	Y	Y	N
63	N	N	Y	N	N	Y	Y	Y	Y	N	Y
64	N	Y	Y	N	Y	Y	Y	Y	Y	N	N
65	N	Y	Y	N	Y	Y	Y	Y	Y	N	Y
66	Y	P 504	Y	N	Y	Y	Y	Y	Y	N	Y
67	Y	Y	Y	Y	Y	Y	Y	Y	Y	N	Y
14		N	Y	N	Y	N	Y		N	N	N
18		N	Y	N	Y	Y	Y	N	N	N	N
68		N				Y	Y				
15		N	Y	N	Y	Y	Y	Y	N	N	N
69		N				N	Y				
70		N				N	Y				
71		N				N	Y				
72		N				N	Y				
73		N				N	Y				
74		N				N	Y				
75		N				N	Y				
76		N				N	Y				
77		N				Y	Y				
78						N	Y				
79						Y	Y				

Notes

Y	Yes
N	No
P	Partial
	Not known
1	New
2	Renamed
3	Moved from 4.3
4	Missing in earlier version
32	Included in Solaris Studio 12.4
100	INTEGER (KIND=8)
108	Missing VALUE on dummy with non-constant type parameters
201	Implemented in 6.0
203	int and real, and coarray
209	Implemented in 6.0
210	gfortran via allocate but not via intrinsic assignment
212	Pathscale, counting bits, not trailz
213	Waiting for update from IBM.
214	leadz, popcnt, and poppar supported. No trailz
300	Supported in 6.0
301	Single image support since 4.6. Multi-image support using OpenCoarrays (including the Fortran 2015 collective subroutines) since 5.1, except allocatable or pointer components of derived type coarrays.
400	A contiguous array variable that is not interoperable but which has interoperable type and kind type parameter (if any) , and a scalar character variable with length greater than 1 and kind C_CHAR in the intrinsic module ISO_C_BINDING, can be used as the argument of the function C_LOC in the intrinsic module ISO_C_BINDING, provided the variable has the POINTER or TARGET attribute.
500	Not supported for complex arrays.
501	Complex types are not accepted for acosh, asinh, and atanh, Additionally, atan2 cannot be accessed via atan.
502	Not supported for procedures.
503	Not supported for null pointer.
504	Only shows a warning with the -Mstandard flag.
600	but only for NULL as initialiser
700	5.x. Coarrays: Usage of coindexed coarrays now invokes the communication library. However, allocatable/pointer components of coindexed coarrays are not yet supported. GCC currently only ships with a single-image library (libcaf_single), but mult-image support based on MPI and GASNet is provided by the libraries of the OpenCoarrays project.
701	8.x
D	These entries taken from the Annex had already been added to the base table by their inclusion in John Reid's paper: ISO/N1828 - Features of F2008 - latest added features.

Here is the summary table with counts. There are errors in this table currently as we use Excel to produce the counts in the table. This will be corrected in a later edition.

Fortran 2008 conformance	Absoft	Arm	Cray	Fujitsu	gfortran	IBM	Intel	NAG	NEC	Oracle	PGI
Compiler version number	14	18.1	8.4.0	2.0.3	8.2	15.1.5	19.0	6.2	1.2.0	8.8	18.10
Y	12	28	66	27	47	54	79	54	54	10	24
Y with notes	1	0	0	0	2	0	0	0		0	0
N	35	33	0	35	12	26	0	11	16	56	42
N with notes	0	0	0	0	2	0	0	0		0	0
P	0	0	0	4	1	0	0	0		0	0
P with notes	0	6	0	1	3	1	0	0		0	1
No information	32	13	14	12	13	-1	0	15	9	14	13
Total	79	79	79	79	79	79	79	79	79	79	79

Here is the TS 29113 conformance table.

TS29113 Compiler conformance table	Absoft	Arm	Cray	Fujitsu	gfortran	IBM	Intel	Nag	Oracle	PGI
Compiler version number	14.0	18.1	8.4.0	2.0.3	8.2	15.1.5	19.0	6.2	8.8	18.10
TS conformant iso_fortran_binding.h	N		Y	N	N	Y	Y	N	N	N
CFI_CDESC_T macro	N		Y	N	N	Y	Y	N	N	N
CFI_* functions declared in iso_fortran_binding.h	N		Y	N	N	Y	Y	N	N	N
Assumed rank	N		Y	N	P	Y	Y	N	N	N
Assumed type or TYPE(*)	N		Y	N	Y	Y	Y	N	N	N
Pass scalar to TYPE(*) , DIMENSION(*)	N		Y	N	N	Y	Y	N	N	N
Non-interoperable array for C_LOC C_F_POINTER	N		Y	N	N	Y	Y	N	N	N

TS29113 Compiler conformance table	Absoft	Arm	Cray	Fujitsu	gfortran	IBM	Intel	Nag	Oracle	PGI
Compiler version number	14.0	18.1	8.4.0	2.0.3	8.2	15.1.5	19.0	6.2	8.8	18.10
Non-interoperable function for C_FUNLOC C_F_PROCPTR	N		Y	N	N	Y	Y	N	N	N
New semantics for ASYNCHRONOUS attribute	N		Y	N	N	Y	Y	N	N	N
RANK intrinsic function	N		Y	N	N	Y	Y	N	N	N
Changes to-SHAPE, SIZE, and UBOUND	N		Y	N	N	Y	Y	N	N	N
Assumed shape dummy arguments for BIND C	N		Y	N	N	N	Y	N	N	N
Allocatable dummy arguments for BIND C	N		Y	N	N	Y	Y	N	N	N
Pointer dummy arguments for BIND C	N		Y	N	N	Y	Y	N	N	N
Optional dummy arguments for BIND C	N		Y	N	N	Y	Y	N	N	N
Assumed-length character arguments for BIND C	N		Y	N	N	P,10	Y	N	N	N

Notes

Y	Yes
N	No
P	Partial
	Not known

10 The dimension of the corresponding dummy argument must not be an expression that contains non-constant operands, e.g. character(*) :: arg(n+1)

Here is the TS 29113 vendor summary table.

TS 29113 Vendor Summary	Absoft	Arm	Cray	Fujitsu	gfortran	IBM	Intel	NAG	Oracle	PGI
Compiler Version number	14	18.1	8.4.0	2.0.3	8.2	15.1.3	19.0	6.2	8.8	18.10
Y	0		16	0	1	11	16	0	0	0
Y with notes	0		0	0	0	0	0	0	0	0
N	16		0	16	14	5	0	16	16	16
N with notes	0		0	0	0	0	0	0	0	0
P	0		0	0	1	0	0	0	0	0
P with notes	0		0	0	0	0	0	0	0	0
No information	0	16	0	0	0	0	0	0	0	0
Total	16	16	16	16	16	16	16	16	16	16

The Fortran 2018 table will be included in future revisions. At the moment we have only had one vendor providing details of 2018 support.

Here is the feature list table for Fortran 2018. It is based on the Introduction of the Fortran 2018 standard, pages xviii-xx.

Section	Sub section	Description
1		Data declaration:
	1.1	Constant properties of an object declared in its entity-decl can be used in its initialization.
	1.2	The EQUIVALENCE and COMMON statements and the block data program unit have been redundant since Fortran 90 and are now specified to be obsolescent.
	1.3	Diagnosis of the appearance of a PROTECTED TARGET variable accessed by use association as a data-target in a structure constructor is required.
2		Data usage and computation:
	2.1	The declared type of the value supplied for a polymorphic allocatable component in a structure constructor is no longer required to be the same as the declared type of the component.
	2.2	FORALL is now specified to be obsolescent.

Section	Sub section	Description
	2.3	The type and kind of an implied DO variable in an array constructor or DATA statement can be specified within the constructor or statement.
	2.4	The SELECT RANK construct provides structured access to the elements of an assumed-rank array.
	2.5	The standard intrinsic operations <, <=, >, and >= (also known as .LT., .LE., .GT., and .GE.) on IEEE numbers provide compareSignaling (relation) operations;
	2.6	The = and /= operations (also known as .EQ. and .NE.) provide compareQuiet (relation) operations.
	2.7	Finalization of an allocatable subobject during intrinsic assignment has been clarified.
	2.8	The char-length in an executable statement is no longer required to be a specification expression.
3		Input/output:
	3.1	The SIZE= specifier can be used with advancing input.
	3.2	It is no longer prohibited to open a file on more than one unit.
	3.3	The value assigned by the RECL= specifier in an INQUIRE statement has been standardized.
	3.4	The values assigned by the POS= and SIZE= specifiers in an INQUIRE statement for a unit that has pending asynchronous operations have been standardized.
	3.5	The G0.d edit descriptor can be used for list items of type Integer, Logical, and Character.
	3.6	The D, E, EN, and ES edit descriptors can have a field width of zero, analogous to the F edit descriptor.
	3.7	The exponent width e in a data edit descriptor can be zero, analogous to a field width of zero.
	3.8	Floating-point formatted input accepts hexadecimal-significand numbers that conform to ISO/IEC/IEEE 60559:2011.
	3.9	The EX edit descriptor provides hexadecimal-significand formatted output conforming to
	3.10	An error condition occurs if unacceptable characters are presented for logical or numeric editing during execution of a formatted input statement.
4		Execution control:
	4.1	The arithmetic IF statement has been deleted.
	4.2	Labeled DO loops have been redundant since Fortran 90 and are now specified to be obsolescent.
	4.3	The nonblock DO construct has been deleted.
	4.4	The locality of a variable used in a DO CONCURRENT construct can be explicitly specified.

Section	Sub section	Description
	4.5	The stop code in a STOP or ERROR STOP statement can be nonconstant.
	4.6	Output of the stop code and exception summary from the STOP and ERROR STOP statements can be controlled.
5		Intrinsic procedures and modules:
	5.1	In a reference to the intrinsic function CMPLX with an actual argument of type complex, no keyword is needed for a KIND argument.
	5.2	In references to the intrinsic functions ALL, ANY, FINDLOC, IALL, IANY, IPARITY, MAXLOC, MAXVAL, MINLOC, MINVAL, NORM2, PARITY, PRODUCT, SUM, and THIS_IMAGE, the actual argument for DIM can be a present optional dummy argument.
	5.3	The new intrinsic function COSHAPE returns the coshape of a coarray.
	5.4	The new intrinsic function OUT_OF_RANGE tests whether a numeric value can be safely converted to a different type or kind.
	5.5	The new intrinsic subroutine RANDOM_INIT establishes the initial state of the pseudorandom number generator used by RANDOM_NUMBER.
	5.6	The new intrinsic function REDUCE performs user-specified array reductions.
	5.7	A processor is required to report use of a nonstandard intrinsic procedure, use of a nonstandard intrinsic module, and use of a nonstandard procedure from a standard intrinsic module.
	5.8	Integer and logical arguments to intrinsic procedures and intrinsic module procedures that were previously required to be of default kind no longer have that requirement, except for RANDOM_SEED.
	5.9	Specific names for intrinsic functions are now deemed obsolescent.
	5.10	All standard procedures in the intrinsic module ISO_C_BINDING, other than C_F_POINTER, are now pure.
	5.11	The arguments to the intrinsic function SIGN can be of different kind.
	5.12	Nonpolymorphic pointer arguments to the intrinsic functions EXTENDS_TYPE_OF and SAME_TYPE_AS need not have defined pointer association status.
	5.13	The effects of invoking the intrinsic procedures COMMAND_ARGUMENT_COUNT, GET_COMMAND, and GET_COMMAND ARGUMENT, on images other than image one, are no longer processor dependent.
	5.14	Access to error messages from the intrinsic subroutines GET_COMMAND, GET_COMMAND_ARGUMENT, and GET_ENVIRONMENT_VARIABLE is provided by an optional ERRMSG argument.
	5.15	The result of NORM2 for a zero-sized array argument has been clarified.
6		Program units and procedures:

Section	Sub section	Description
	6.1	The IMPORT statement can appear in a contained subprogram or BLOCK construct, and can restrict access via host association;
	6.2	Diagnosis of violation of the IMPORT restrictions is required.
	6.3	The GENERIC statement can be used to declare generic interfaces.
	6.4	The number of procedure arguments is used in generic resolution.
	6.5	In a module, the default accessibility of entities accessed from another module can be controlled separately from the default accessibility of entities declared in the using module.
	6.6	An IMPLICIT NONE statement can require explicit declaration of the EXTERNAL attribute throughout a scoping unit and its contained scoping units.
	6.7	A defined operation need not specify INTENT (IN) for a dummy argument with the VALUE attribute.
	6.8	A defined assignment need not specify INTENT (IN) for the second dummy argument if it has the VALUE attribute.
	6.9	Procedures that are not declared with an asterisk type-param-value, including ELEMENTAL procedures, can be invoked recursively by default;
	6.10	The RECURSIVE keyword is advisory (most procedures are recursive by default) only.
	6.11	The NON_RECURSIVE keyword specifies that a procedure is not recursive.
	6.12	The ERROR STOP statement can appear in a pure subprogram.
	6.13	A dummy argument of a pure function is permitted in a variable definition context, if it has the VALUE attribute.
	6.14	A coarray dummy argument can be referenced or defined by another image.
7		Features previously described by ISO/IEC TS 29113:2012:
	7.1	A dummy data object can assume its rank from its effective argument.
	7.2	A dummy data object can assume the type from its effective argument, without having the ability to perform type selection.
	7.3	An interoperable procedure can have dummy arguments that are assumed-type and/or assumed-rank.
	7.4	An interoperable procedure can have dummy data objects that are allocatable, assumed-shape, optional, or pointers.
	7.5	The character length of a dummy data object of an interoperable procedure can be assumed.
	7.6	The argument to C_LOC can be a noninteroperable array.
	7.7	The FPTR argument to C_F_POINTER can be a noninteroperable array pointer.

Section	Sub section	Description
	7.8	The argument to C_FUNLOC can be a noninteroperable procedure.
	7.9	The FPTR argument to C_F_PROCPOINTER can be a noninteroperable procedure pointer.
	7.10	There is a new named constant C_PTRDIFF_T to provide interoperability with the C type ptrdiff_t.
	7.11	Additionally to ISO/IEC TS 29113:2012, a scalar actual argument can be associated with an assumed-type assumed-size dummy argument, an assumed-rank dummy data object that is not associated with an assumed-size array can be used as the argument to the function C_SIZEOF from the intrinsic module ISO_C_BINDING, and the type argument to CFI_establish can have a positive value corresponding to an interoperable C type.
8		Changes to the intrinsic modules IEEE_ARITHMETIC, IEEE_EXCEPTIONS, and IEEE_FEATURES for conformance with ISO/IEC/IEEE 60559:2011:
	8.1	There is a new, optional, rounding mode IEEE_AWAY.
	8.2	The new type IEEE_MODES_TYPE encapsulates all floating-point modes.
	8.3	Features associated with subnormal numbers can be accessed with functions and types named ...SUBNORMAL... (the old ...DENORMAL... names remain).
	8.4	The new function IEEE_FMA performs fused multiply-add operations.
	8.5	The function IEEE_INT performs rounded conversions to integer type.
	8.6	The new functions IEEE_MAX_NUM, IEEE_MAX_NUM_MAG, IEEE_MIN_NUM, and IEEE_MIN_NUM_MAG calculate maximum and minimum numeric values.
	8.7	The new functions IEEE_NEXT_DOWN and IEEE_NEXT_UP return the adjacent machine numbers.
	8.8	The new functions IEEE_QUIET_EQ, IEEE_QUIET_GE, IEEE_QUIET_GT, IEEE_QUIET_LE, IEEE_QUIET_LT, and IEEE_QUIET_NE perform quiet comparisons.
	8.9	The new functions IEEE_SIGNALING_EQ, IEEE_SIGNALING_GE, IEEE_SIGNALING_GT, IEEE_SIGNALING_GE, IEEE_SIGNALING_LE, IEEE_SIGNALING_LT, and IEEE_SIGNALING_NE perform signaling comparisons.
	8.10	The decimal rounding mode can be inquired and set independently of the binary rounding mode, using the RADIX argument to IEEE_GET_ROUNDING_MODE and IEEE_SET_ROUNDING_MODE.
	8.11	The new function IEEE_REAL performs rounded conversions to real type.
	8.12	The function IEEE_REM now requires its arguments to have the same radix.
	8.13	The function IEEE_RINT now has a ROUND argument to perform specific rounding.
	8.14	The new function IEEE_SIGNBIT tests the sign bit of an IEEE number.

Section	Sub section	Description
9		Features previously described by ISO/IEC TS 18508:2015:
	9.1	The CRITICAL statement has optional ERRMSG= and STAT= specifiers.
	9.2	The intrinsic subroutines ATOMIC_DEFINE and ATOMIC_REF have an optional STAT argument.
	9.3	The new intrinsic subroutines ATOMIC_ADD, ATOMIC_AND, ATOMIC_CAS, ATOMIC_FETCH_ADD, ATOMIC_FETCH_AND, ATOMIC_FETCH_OR, ATOMIC_FETCH_XOR, ATOMIC_OR, and ATOMIC_XOR perform atomic operations.
	9.4	The new intrinsic functions FAILED_IMAGES and STOPPED_IMAGES return indices of images known to have failed or stopped respectively.
	9.5	The new intrinsic function IMAGE_STATUS returns the image execution status of an image.
	9.6	The intrinsic subroutine MOVE_ALLOC has optional ERRMSG and STAT arguments.
	9.7	The intrinsic functions IMAGE_INDEX and NUM_IMAGES have additional forms with a TEAM or TEAM_NUMBER argument.
	9.8	The intrinsic function THIS_IMAGE has an optional TEAM argument.
	9.9	The EVENT POST and EVENT WAIT statements, the intrinsic subroutine EVENT_QUERY, and the type EVENT_TYPE provide an event facility for one-sided segment ordering.
	9.10	The CHANGE TEAM construct, derived type TEAM_TYPE, FORM TEAM and SYNC TEAM statements, intrinsic functions GET_TEAM and TEAM_NUMBER, and the TEAM= and TEAM_NUMBER= specifiers on image selectors, provide a team facility for a subset of the programs images to act in concert as if it were the set of all images. This team facility allows an allocatable coarray to be allocated or deallocated on a subset of images.
	9.11	The new intrinsic subroutines CO_BROADCAST, CO_MAX, CO_MIN, CO_REDUCE, and CO_SUM perform collective reduction operations on the images of the current team.
	9.12	The concept of failed images, the FAIL IMAGE statement, the STAT= specifier on image selectors, and the named constant STAT_FAILED_IMAGE from the intrinsic module ISO_FORTRAN_ENV provide support for fault-tolerant parallel execution.
10		Changes to features previously described by ISO/IEC TS 18508:2015:
	10.1	The CHANGE TEAM and SYNC TEAM statements, and the TEAM= specifier on image selectors, permit the team to be specified by an expression.
	10.2	The intrinsic functions FAILED_IMAGES and STOPPED_IMAGES have no restriction on the kind of their result.
	10.3	The name of the function argument to the intrinsic function CO_REDUCE is OPERATION instead of OPERATOR; this argument is not required to be commutative.
	10.4	The named constant STAT_UNLOCKED_FAILED_IMAGE from the intrinsic module ISO_FORTRAN_ENV indicates that a lock variable was locked by an image that failed.

Section	Sub section	Description
	10.5	The team number for the initial team can be used in image selectors, and in the intrinsic functions NUM_IMAGES and IMAGE_INDEX.
	10.6	A team variable that appears in a CHANGE TEAM statement can no longer be defined or become undefined during execution of the CHANGE TEAM construct.
	10.7	All images of the current team are no longer required to execute the same CHANGE TEAM statement.
	10.8	A variable of type TEAM_TYPE from the intrinsic module ISO_FORTRAN_ENV is not permitted to be a coarray.
	10.9	A variable of type TEAM_TYPE from the intrinsic module ISO_FORTRAN_ENV can have a pointer component, and a team variable becomes undefined if assigned a value from another image.
	10.10	The intrinsic function UCOBOUND produces a value for the final upper cobound that is always relative to the current team.
	10.11	An EXIT statement can be used to complete execution of a CHANGE TEAM or CRITICAL construct.