From: Michael Hennecke
To: WG5 and J3
Subject: Interoperability PDTR - type aliasing revised
Date: 1997/07/22

This document contains a set of edits against Fortran 95 which improve the type aliasing mechanism in section 3.3.9 of WG5/N1277. Taking into account several comments (notably those in J3/97-187r1 and J3/97-188r1),

* the ambiguity of R1606 in WG5/N1277 is removed
  (by always requiring the double colon in a <type-alias-stmt>);
* a new keyword ALIAS is used instead of TYPE in the <type-alias-stmt>;
* a structure constructor is only allowed for aliases of derived types;
* it is now possible to define a list of aliases in one stmt;
* integration into IS 1539-1 is improved.

This material may be used for the revision of the Interoperability PDTR, replacing all but the first paragraph of 3.3.9 and Note 3.29 of the PDTR. In Note 3.29 of the PDTR, also change "TYPE XID" to "ALIAS :: XID" and "TYPE Window" to "ALIAS :: Window".

Edits against ISO/IEC 1539-1:1997

Page 10
Subclause 2.1
In

R214	<declaration-construct>	is	<derived-type-def>
		or	<interface-block>
		...	

add after line 7:

		or	<type-alias-stmt>

--
Page 29
Clause 4
On line 5, add " A type alias statement (4.5) may be used to establish additional names for intrinsic or derived types." after "(4.4.1).".
--
Page 29
Clause 4
On line 34, add " A type alias name may only be used if its definition is accessible (4.5)." to the last paragraph.
--

```
Page 44
Subclause 4.4.4
In
  R413  <structure-constructor>  is  <type-name> ( <expr-list> )
add
                                 or  <type-alias-name> ( <expr-list> )
--
Page 44
Subclause 4.4.4
After R413, add
  Constraint: The <type-alias-name> (4.5) shall be an accessible type
              alias for a derived type.
--

Page 45
After subclause 4.4, add a new subclause 4.5 and renumber:

  4.5  Type aliasing

  Additional names for intrinsic and derived data types may be established.
  A type alias statement is required to define the type alias name.

  By default, type alias names defined in the specification part of a
  module are accessible (5.1.2.2, 5.2.3) in any scoping unit that accesses
  the module. This default may be changed to restrict the accessibility of
  such type alias names to the host module itself.
  A particular type alias name may be declared to be public or private
  regardless of the default accessibility declared for the module.

  The type specifier for type aliases is the keyword TYPE followed by
  the type alias name in parantheses (R502).

  R431a  <type-alias-stmt>  is  ALIAS [, <access-spec>] :: <alias-def-list>

  R431b  <alias-def>        is  <type-alias-name> => <type-spec>

  Constraint: An <access-spec> (5.1.2.2) is only allowed if the
              <type-alias-stmt> is within the <specification-part>
              of a module.

  Constraint: A <type-alias-name> shall not be the same as the name of
              any intrinsic type defined in this standard nor the same
              as any other accessible <type-name> or <type-alias-name>.

  If the type specifier (5.1.1) in the <type-spec> (R502) specifies an
  intrinsic type, the <type-alias-name> is an alias for that intrinsic
  type. The type parameters associated with the type alias are as
  specified  in the <type-spec>, with defaults as in 5.1.1.
```

```
  If the <type-spec> denotes a derived type or type alias, the
  <type-alias-name> is an alias for that derived type or type alias
  which shall have been defined previously in the scoping unit
  or be accessible there by use or host association.

  A type alias name may be used in a type declaration statement (5.1),
  where it specifies the same type as the <type-spec> in the
  <alias-def> which defines the type alias name.
  If the type alias is for a derived type, it may also be used in a
  structure constructor (4.4.4) for that derived type.

Note 4.34a
For derived type <type-name>s, this is similar to a <rename> of
the name in a USE statement. The <type-alias-stmt> is more general
in that it also allows aliasing intrinsic types, and is not limited
to the USE statement.

[ Example: huge integer kind and extended precision derived type ]

The accessibility of a type alias name may be declared explicitly by
an <access-spec> (5.1.2.2) in its <type-alias-stmt> or in an
<access-stmt> (5.2.3).
The accessibility is the default if it is not declared explicitly.
If a type alias definition is private, then the type alias name
is accessible only within the module containing the definition.
If a type alias is for a derived type, then the type alias name
may be used in a structure constructor only when it is accessible.

--
Page 47
Subclause 5.1
In
R502  <type-spec>  is  INTEGER [ <kind-selector> ]
           or  REAL [ <kind-selector> ]
           ...
add after line 25:
           or  TYPE(<type-alias-name>)
--
Page 52
Subclause 5.1.1.7
On line 2, change "A TYPE type" to "A TYPE(<type-name>) type".
--
```

Page 52
After subclause 5.1.1.7, add a new subclause 5.1.1.8:

5.1.1.8 Type aliasing

A TYPE(<type-alias-name>) type specifier is used to declare entities of a type specified by the <type-alias-name>.
The type alias name shall have been defined previously in the scoping unit or be accessible there by use or host association.
If the data entity is a function result, the type alias name may be specified in the function statement provided it is defined within the body of the function or is accessible there by use or host association.

The TYPE(<type-alias-name>) type specifier is interpreted as if it were replaced by the <type-spec> in the <alias-def> which defines the type alias name.

--
Page 58
Subclause 5.2.3
In the second constraint, add " type alias," after "derived type,".
--
Page 275
Subclause 14.1.2
In (1) on line 36, add " type alias names," after "derived types,".