AADL Inspector 1.2
AADL Naming Rules

Ellidiss Technologies
http://www.ellidiss.fr
aadl@ellidiss.fr

1 Static Rules Checkers

AADL Inspector comes with a set of pre-defined static rules checkers. Their goal is to cover the various semantic rules specified by the **AADL** standard. In the standard document **SAE-AS5506B**, they are classified into three categories:

- *naming rules*: specify the appropriate usage of identifiers.
- *legality rules*: express semantic restrictions on syntactically valid text.
- *consistency rules*: define constraints related to the instance model.

This document describes the **AADL Inspector** Naming Rules checker.

These standard naming rules are not fully implemented in the current version of the tool and the following sub-sections provide details about the actual **AADL** rules coverage. For easier traceability, precise references to the standard document are given for each implemented rule.

2 AADL subset

Naming rule identifiers are composed the letter `N` followed by an integer number. Numbering restarts at the beginning of each section in the **AADL** standard document. Rules that are currently implemented refer to the following sections of the document **SAE-AS5506B**:

- *AADL Specifications*: section 4.1
- *Packages*: section 4.2
- *Component Types*: section 4.3
- *Component Implementations*: section 4.4

2.1 AADL Specifications (AS5506B-4.1)

Checked naming rules for global **AADL** specifications are `N1` and `N3`. Note that rules `N2`, `N4`, `N5`, `N7` and `N8` express definitions or permissions and are thus not subject to

checking. In addition, rule `N6` stating that an identifier must not be a reserved word, is checked by the **AADL** syntactic analyser (*aadlrev*).

2.2 Packages (AS5506B-4.2)

Checked naming rules for **AADL** packages are `N1`, `N5`, `N6`, `N7`, `N8`, `N9`, `N10`, `N11`, `N12`, `N13`, `N14`, `N15`, `N16` and `N17`. Note that rules `N2`, `N3` and `N4` express definitions or permissions and are thus not subject to checking.

2.3 Component Types (AS5506A-4.3)

Checked naming rule for **AADL** component types are `N1`, `N2`, `N3`, `N6`, `N8` and `N9`. The other rules will be implemented in the next releases of the product.

2.4 Component Implementations (AS5506A-4.4)

Checked naming rule for **AADL** component implementations are `N1`, `N2`, `N3`, `N4` and `N5`. The other rules will be implemented in the next releases of the product.

2.5 Subcomponents (AS5506A-4.5)

Checked naming rule for **AADL** subcomponents is `N1`. The other rules will be implemented in the next releases of the product.

3 Test Cases

3.1 AADL Specifications (AS5506B-4.1)

N1 An AADL specification has one *global* namespace. The package and property set identifiers reside in this space and must be unique.

```
PROPERTY SET P IS
END P;
```

```
PACKAGE P PUBLIC
END P;
```

ERROR lines 9 and 6: the same identifier P cannot be used for both a Package and a Property Set (ref. AADL v2 - 4.1 - N1)

3.2 Packages (AS5506B-4.2)

N1 A defining package name consists of a sequence of one or more package identifiers separated by a double colon ("::"). **A defining package name must be unique in the global namespace.**

```
PACKAGE P PUBLIC
END P;

PACKAGE P PUBLIC
END P;
```

ERROR lines 6 and 9: Package identifier P must be unique (ref. AADL v2 - 4.2 - N1)
ERROR lines 9 and 6: Package identifier P must be unique (ref. AADL v2 - 4.2 - N1)

N5 The reference to an item declared in another package must be an item *name* qualified with a package name separated by a double colon ("::"). **The package name must be listed** in the `import_declaration` (**with**) of the package with the reference, i.e., **in the `import_declaration` of the public section, if the reference is in the public section, and in the `import_declaration` of the public or private section if the reference is in the private section of the package** (a). **Only classifiers in the public package section can be referenced** (b).
Note: References to items declared in the same package as the reference do not have to be qualified. If they are quailified the package is not required to be listed in the `import_declaration`.

Test case 1:

```
PACKAGE P1
PRIVATE

DATA D1 END D1;
```

```
DATA IMPLEMENTATION D1.I
END D1.I;

SUBPROGRAM S1 END S1;

SUBPROGRAM IMPLEMENTATION S1.I
END S1.I;

END P1;

PACKAGE P2
PUBLIC

RENAMES P1::ALL;

DATA D2 EXTENDS D1
END D2;

DATA IMPLEMENTATION D2.I EXTENDS D1.I
END D2.I;

SUBPROGRAM S2
PROTOTYPES
  X : DATA D1.I;
FEATURES
  P : IN PARAMETER D1.I;
  R : OUT PARAMETER X;
END S2;

SUBPROGRAM IMPLEMENTATION S2.I
SUBCOMPONENTS
  A : DATA D1.I;
CALLS
  S : { C : SUBPROGRAM S1.I; };
END S2.I;

THREAD T END T;

THREAD IMPLEMENTATION T.I
SUBCOMPONENTS
  S : SUBPROGRAM S2.I ( X => DATA D1.I );
ANNEX BEHAVIOR_SPECIFICATION {**
VARIABLES
  V : D1.I;
```

```
STATES
  E : INITIAL COMPLETE FINAL STATE;
TRANSITIONS
  F : S -[]-> S { V := V + 1 };
**};
END T.I;

END P2;
```

WARNING line 27: Missing import declaration for package P1 (ref. AADL v2 - 4.2 - N5a)
WARNING line 29: Missing import declaration for package P1 (ref. AADL v2 - 4.2 - N5a)
WARNING line 42: Missing import declaration for package P1 (ref. AADL v2 - 4.2 - N5a)
WARNING line 44: Missing import declaration for package P1 (ref. AADL v2 - 4.2 - N5a)
WARNING line 36: Missing import declaration for package P1 (ref. AADL v2 - 4.2 - N5a)
WARNING line 34: Missing import declaration for package P1 (ref. AADL v2 - 4.2 - N5a)
WARNING line 51: Missing import declaration for package P1 (ref. AADL v2 - 4.2 - N5a)
WARNING line 54: Missing import declaration for package P1 (ref. AADL v2 - 4.2 - N5a)
WARNING line 24: Missing import declaration for package P1 (ref. AADL v2 - 4.2 - N11)

Test case 2:

```
PACKAGE P1
PRIVATE

DATA D1 END D1;

DATA IMPLEMENTATION D1.I
END D1.I;

SUBPROGRAM S1 END S1;

SUBPROGRAM IMPLEMENTATION S1.I
```

```
END S1.I;

END P1;

PACKAGE P2
PUBLIC
WITH P1;

RENAMES P1::ALL;

DATA D2 EXTENDS D1
END D2;

DATA IMPLEMENTATION D2.I EXTENDS D1.I
END D2.I;

SUBPROGRAM S2
PROTOTYPES
  X : DATA D1.I;
FEATURES
  P : IN PARAMETER D1.I;
  R : OUT PARAMETER X;
END S2;

SUBPROGRAM IMPLEMENTATION S2.I
SUBCOMPONENTS
  A : DATA D1.I;
CALLS
  S : { C : SUBPROGRAM S1.I; };
END S2.I;

THREAD T END T;

THREAD IMPLEMENTATION T.I
SUBCOMPONENTS
  S : SUBPROGRAM S2.I ( X => DATA D1.I );
ANNEX BEHAVIOR_SPECIFICATION {**
VARIABLES
  V : D1.I;
STATES
  E : INITIAL COMPLETE FINAL STATE;
TRANSITIONS
  F : S -[]-> S { V := V + 1 };
**};
```

```
END T.I;

END P2;
```

ERROR lines 28 and 9: Classifier D1 is not defined in the public section of package P1 (ref. AADL v2 - 4.2 - N5b)
ERROR lines 30 and 12: Classifier D1.I is not defined in the public section of package P1 (ref. AADL v2 - 4.2 - N5b)
ERROR lines 43 and 12: Classifier D1.I is not defined in the public section of package P1 (ref. AADL v2 - 4.2 - N5b)
ERROR lines 45 and 17: Classifier S1.I is not defined in the public section of package P1 (ref. AADL v2 - 4.2 - N5b)
ERROR lines 37 and 12: Classifier D1.I is not defined in the public section of package P1 (ref. AADL v2 - 4.2 - N5b)
ERROR lines 35 and 12: Classifier D1.I is not defined in the public section of package P1 (ref. AADL v2 - 4.2 - N5b)
ERROR lines 52 and 12: Classifier D1.I is not defined in the public section of package P1 (ref. AADL v2 - 4.2 - N5b)
ERROR lines 55 and 12: Classifier D1.I is not defined in the public section of package P1 (ref. AADL v2 - 4.2 - N5b)

N6 The reference to a property other than predeclared properties must be an property *name* qualified with a property set name separated by a double colon ("::"). The property set name must be listed in the **with** declaration of the package with the reference. Predeclared properties may be, but are not required to be qualified by the property set name.

Test case 1:

```
PROPERTY SET P1 IS
  N : AADLINTEGER APPLIES TO (SYSTEM);
END P1;

PACKAGE P2
PUBLIC

SYSTEM S
PROPERTIES
  P1::N => 1;
```

```
  Source_Text => ("s.c");
END S;

END P2;
```

WARNING line 15: Missing import declaration for P1 (ref. AADL v2 - 4.2 - N6a)

Test case 2:

```
PROPERTY SET P1 IS
  N : AADLINTEGER APPLIES TO (SYSTEM);
END P1;

PACKAGE P2
PUBLIC
WITH P1;

SYSTEM S
PROPERTIES
  N => 1;
  SOURCE_TEXT => ("S.C");
END S;

END P2;
```

WARNING line 16: Missing property set qualifier for N (ref. AADL v2 - 4.2 - N6b)

N7 The package name in a `import_declaration` must exist in the global name space.

```
PACKAGE P1
PUBLIC
WITH P2;

END P1;
```

WARNING line 8: Package P2 has not been found (ref. AADL v2 - 4.2 - N7)

N8 The property set identifier in a `import_declaration` must exist in the global name space.

```
PACKAGE P1
```

```
PUBLIC

WITH P2;
WITH Programming_Properties;

SYSTEM S
PROPERTIES
  P2::N => 1;
  Programming_Properties::Source_Text => ("s.c");
END S;

END P1;

PACKAGE P2
PUBLIC

END P2;
```

WARNING line 9: Property Set P2 has not been found (ref. AADL v2 - 4.2 - N8)

N9 Items declared in the **private** section of the package can only be referenced from within the **private** section of the package.

```
PACKAGE P1
PUBLIC

SYSTEM S END S;

SYSTEM IMPLEMENTATION S.I
SUBCOMPONENTS
  D : DATA D.I;
END S.I;

PRIVATE

DATA D END D;

DATA IMPLEMENTATION D.I
END D.I;

END P1;
```

ERROR lines 21 and 13: Private classifier D.I cannot be referenced from within the

public section of package P1 (ref. AADL v2 - 4.2 - N9)
ERROR line 13: Classifier D.I cannot be found or is not visible in package P1 (ref. AADL v2 - 4.2 - N10)

N10 If the qualifying package identifier of a qualified reference is missing, the referenced component classifier, feature group type, or item in an annex library must exist in the same package as the reference.

```
PACKAGE P1
PUBLIC

DATA D END D;

DATA IMPLEMENTATION D.I
END D.I;

END P1;

PACKAGE P2
PUBLIC

SYSTEM S END S;

SYSTEM IMPLEMENTATION S.I
SUBCOMPONENTS
  D : DATA D.I;
END S.I;

END P2;
```

ERROR line 23: Classifier D.I cannot be found or is not visible in package P2 (ref. AADL v2 - 4.2 - N10)

N11 The package name referenced in an `alias_declaration` must exist in the global namespace and must be listed in the `import_declaration`.

```
PACKAGE P1
PUBLIC

DATA D END D;

END P1;
```

```
PACKAGE P2
PUBLIC

D RENAMES DATA P1::D;

END P2;
```

WARNING line 16: Missing import declaration for package P1 (ref. AADL v2 - 4.2 - N11)

N12 The classifier referenced by the `alias_declaration` must exist in the name space of the **public** section of the package being referenced by the `alias_declaration`.

```
PACKAGE P1
PRIVATE

SYSTEM S END S;

END P1;

PACKAGE P2
PUBLIC
WITH P1;

S RENAMES SYSTEM P1::S;

END P2;
```

ERROR line 17: Classifier S cannot be found in the public section of package P1 (ref. AADL v2 - 4.2 - N12)

N13 The classifier referenced by the alias declaration must refer to a component type or a feature group type.

*This rule is checked by the **AADL** syntactic analyser (aadlrev).*

N14 The defining identifier of an `alias_declaration` must be unique in the namespace of the package containing the `alias_declaration`. If an

`alias_declaration` defines an alias for a package then the alias name must not conflict with any package name listed in an `import_declaration` or that of the package containing the `alias_declaration`.

```
PACKAGE P1
PUBLIC
WITH P2;
WITH P3;

D RENAMES DATA P2::D;
A RENAMES DATA P2::D;
A RENAMES DATA P3::D;
P2 RENAMES PACKAGE P3;
P1 RENAMES PACKAGE P3;

DATA D END D;

END P1;

PACKAGE P2
PUBLIC

DATA D END D;

END P2;

PACKAGE P3
PUBLIC

DATA D END D;

END P3;
```

ERROR lines 11 and 17: Alias identifier D is not unique in package P1 (ref. AADL v2 - 4.2 - N14)
ERROR lines 12 and 13: Alias identifier A is not unique in package P1 (ref. AADL v2 - 4.2 - N14)
ERROR lines 13 and 12: Alias identifier A is not unique in package P1 (ref. AADL v2 - 4.2 - N14)
ERROR line 14: Alias identifier P2 is not valid in package P1 (ref. AADL v2 - 4.2 - N14)

ERROR line 15: Alias identifier P1 is not valid in package P1 (ref. AADL v2 - 4.2 - N14)

N15 The `alias_declaration` makes the publicly visible identifier of classifiers declared in another package accessible in the name space of the package containing the `alias_declaration`. If the alias declaration for a classifier does not include a defining identifier then the referenced classifier identifier is used as defining identifier and this identifier must be unique in the namespace of the package with the alias declaration.

```
PACKAGE P1
PUBLIC

DATA D END D;

END P1;

PACKAGE P2
PUBLIC
WITH P1;

RENAMES DATA P1::D;

DATA D END D;

END P2;
```

ERROR lines 17 and 19: Alias identifier D is not unique in package: P2 (ref. AADL v2 - 4.2 - N15)

N16 If the `alias_declaration` renames all publicly visible identifiers of component types and feature group types by naming the package and **all**, then all those identifiers must also be unique in the namespace of the package with the alias declaration.

```
PACKAGE P1
PUBLIC

DATA D END D;
```

```
END P1;

PACKAGE P2
PUBLIC
WITH P1;

RENAMES P1::ALL;

DATA D END D;

END P2;
```

ERROR lines 17 and 9: Alias identifier D is not unique in package: P2 (ref. AADL v2 - 4.2 - N16)

N17 The identifiers introduced by the `alias_declaration` are only accessible within the package. **a)** When declared in the public package section, they can be referenced within the public and private package section, but not from other packages. **b)** When declared in the private package section, they can be referenced within the private package section only.

Test case 1:

```
PACKAGE P1
PUBLIC

DATA D END D;

END P1;

PACKAGE P2
PUBLIC
WITH P1;

D RENAMES DATA P1::D;

END P2;

PACKAGE P3
PUBLIC
WITH P2;

SYSTEM S END S;
```

```
SYSTEM IMPLEMENTATION S.I
SUBCOMPONENTS
  D : DATA P2::D;
END S.I;

END P3;
```

ERROR lines 17 and 29: Alias identifier D cannot be referenced outside package: P2 (ref. AADL v2 - 4.2 - N17a)

Test case 2:

```
PACKAGE P1
PUBLIC

DATA D END D;

END P1;

PACKAGE P2
PUBLIC
WITH P1;

SYSTEM S END S;

SYSTEM IMPLEMENTATION S.I
SUBCOMPONENTS
  D : DATA D;
END S.I;

PRIVATE
WITH P1;

D RENAMES DATA P1::D;

END P2;
```

ERROR line 21: Classifier D cannot be found or is not visible in package P2 (ref. AADL v2 - 4.2 - N10)
ERROR lines 27 and 21: Alias identifier D cannot be referenced in public section of package: P2 (ref. AADL v2 - 4.2 - N17b)

N18 The alias declared for a component type can be used instead of a qualified component type in a reference to a component implementation.

This rule expresses a permission and is thus not subject to checking.

3.3 Component types (AS5506B-4.3)

N1 The defining identifier for a component type must be unique in the namespace of the package within which it is declared.

```
PACKAGE P1
PUBLIC

THREAD T1
END T1;

DATA T1
END T1;

END P1;
```

ERROR lines 10 and 13: Component type identifier T1 is not unique in package P1 (ref. AADL v2 - 4.3 - N1)
ERROR lines 13 and 10: Component type identifier T1 is not unique in package P1 (ref. AADL v2 - 4.3 - N1)

N2 Each component type has a local namespace for defining identifiers of prototypes, features, modes, mode transitions, and flow specifications. That is, defining prototype, defining feature, defining modes and mode transitions, and defining flow specification identifiers must be unique in the component type namespace.

```
PACKAGE P1
PUBLIC

DATA D END D;

SYSTEM S
PROTOTYPES
  X : DATA D;
```

```
FEATURES
  X : IN DATA PORT D;
  Y : OUT DATA PORT D;
  E : IN EVENT PORT;
FLOWS
  X : FLOW PATH X -> Y;
MODES
  X : INITIAL MODE;
  Y : MODE;
  X : X -[ E ]-> Y;
END S;

END P1;
```

ERROR lines 15 and 13: Identifier X must be unique in component type S (ref. AADL v2 - 4.3 - N2)
ERROR lines 16 and 22: Identifier Y must be unique in component type S (ref. AADL v2 - 4.3 - N2)
ERROR lines 13 and 15: Identifier X must be unique in component type S (ref. AADL v2 - 4.3 - N2)
ERROR lines 21 and 15: Identifier X must be unique in component type S (ref. AADL v2 - 4.3 - N2)
ERROR lines 22 and 16: Identifier Y must be unique in component type S (ref. AADL v2 - 4.3 - N2)
ERROR lines 23 and 15: Identifier X must be unique in component type S (ref. AADL v2 - 4.3 - N2)
ERROR lines 19 and 15: Identifier X must be unique in component type S (ref. AADL v2 - 4.3 - N2)

N3 The component type identifier of the ancestor in a component type extension, i.e., that appears after the reserved word **extends**, must be defined in the specified package namespace. If no package name is specified, then the identifier must be defined in the namespace of the package the extension is declared in.

```
PACKAGE P1
PUBLIC

DATA D1 END D1;

END P1;
```

```
PACKAGE P2
PUBLIC

WITH P1;

DATA D extends P1::D
END D;

END P2;
```

ERROR line 19: Identifier D is not defined in package P1 (ref. AADL v2 - 4.3 - N3)

N4 When a component type extends another component type, a component type namespace includes all the identifiers in the namespaces of its ancestors.

This rule expresses a definition and is thus not subject to checking.

N5 A component type that extends another component type does not include the identifiers of the implementations of its ancestors.

This rule expresses a definition and is thus not subject to checking.

N6 The defining identifier of a feature, flow specification, mode, mode transition, or prototype must be unique in the namespace of the component type.

```
PACKAGE P1
PUBLIC

SYSTEM S1
PROTOTYPES
  U : DATA;
FEATURES
  V : IN EVENT PORT;
FLOWS
  W : FLOW SINK V;
MODES
  X : INITIAL MODE;
```

```
  Y : MODE;
  Z : X -[ V ]-> Y;
END S1;

SYSTEM S2 EXTENDS S1
PROTOTYPES
  U : DATA;
FEATURES
  V : IN EVENT PORT;
FLOWS
  W : FLOW SINK V;
MODES
  X : INITIAL MODE;
  Y : MODE;
  Z : X -[ V ]-> Y;
END S2;

SYSTEM S3 EXTENDS S2
PROTOTYPES
  U : DATA;
FEATURES
  V : IN EVENT PORT;
FLOWS
  W : FLOW SINK V;
MODES
  X : INITIAL MODE;
  Y : MODE;
  Z : X -[ V ]-> Y;
END S3;

END P1;
```

ERROR lines 26 and 13: Identifier V must be unique in the namespace of component S2 (ref. AADL v2 - 4.3 - N6)
ERROR lines 39 and 26: Identifier V must be unique in the namespace of component S3 (ref. AADL v2 - 4.3 - N6)
ERROR lines 24 and 11: Identifier U must be unique in the namespace of component S2 (ref. AADL v2 - 4.3 - N6)
ERROR lines 37 and 24: Identifier U must be unique in the namespace of component S3 (ref. AADL v2 - 4.3 - N6)
ERROR lines 30 and 17: Identifier X must be unique in the namespace of component S2 (ref. AADL v2 - 4.3 - N6)

ERROR lines 31 and 18: Identifier Y must be unique in the namespace of component S2 (ref. AADL v2 - 4.3 - N6)
ERROR lines 43 and 30: Identifier X must be unique in the namespace of component S3 (ref. AADL v2 - 4.3 - N6)
ERROR lines 44 and 31: Identifier Y must be unique in the namespace of component S3 (ref. AADL v2 - 4.3 - N6)
ERROR lines 32 and 19: Identifier Z must be unique in the namespace of component S2 (ref. AADL v2 - 4.3 - N6)
ERROR lines 45 and 32: Identifier Z must be unique in the namespace of component S3 (ref. AADL v2 - 4.3 - N6)
ERROR lines 28 and 15: Identifier W must be unique in the namespace of component S2 (ref. AADL v2 - 4.3 - N6)
ERROR lines 41 and 28: Identifier W must be unique in the namespace of component S3 (ref. AADL v2 - 4.3 - N6)

N7 The refinement identifier of a feature, flow specification, or prototype refinement refers to the closest refinement or the defining declaration of the feature going up the component type ancestor hierarchy.

```
PACKAGE P1
PUBLIC

SYSTEM S1
PROTOTYPES
  U : DATA;
FEATURES
  V : IN EVENT PORT;
FLOWS
  W : FLOW SINK V;
END S1;

SYSTEM S2 EXTENDS S1
END S2;

SYSTEM S3 EXTENDS S2
PROTOTYPES
  U1 : REFINED TO DATA;
FEATURES
  V1 : REFINED TO IN EVENT PORT;
FLOWS
```

```
  W1 : REFINED TO FLOW SINK;
END S3;

END P1;
```

ERROR line 23: Refinement identifier U1 must be defined in an ancestor of component type S3 (ref. AADL v2 - 4.3 - N7)
ERROR line 25: Refinement identifier V1 must be defined in an ancestor of component type S3 (ref. AADL v2 - 4.3 - N7)
ERROR line 27: Refinement identifier W1 must be defined in an ancestor of component type S3 (ref. AADL v2 - 4.3 - N7)

N8 The prototypes referenced by prototype binding declarations must exist in the local namespace of the component type being extended.

```
PACKAGE P1
PUBLIC

ABSTRACT A END A;
DATA D END D;

SYSTEM S1
PROTOTYPES
  X : ABSTRACT A;
END S1;

SYSTEM S2
END S2;

SYSTEM S3 EXTENDS S2 ( X => DATA D )
END S3;

END P1;

PACKAGE P2
PUBLIC
WITH P1;

S RENAMES SYSTEM P1::S2;

DATA D END D;
```

```
SYSTEM S3 EXTENDS S ( X => DATA D )
END S3;

END P2;
```

ERROR line 20: Prototype X must exist in the namespace of component type S3 (ref. AADL v2 - 4.3 - N8)
ERROR line 33: Prototype X must exist in the namespace of component type S3 (ref. AADL v2 - 4.3 - N8)

N9 Mode transitions declared in the component type may not refer to event or event data ports of subcomponents.

This rule is not currently checked.

3.4 Component implementations (AS5506B-4.4)

N1 A component implementation name consists of a component type identifier and a component implementation identifier separated by a dot (“.”). The first identifier of the defining component implementation name must name a component type that is declared in the same package as the component implementation, or name an *alias* to a component type in another package.

```
PACKAGE P1
PUBLIC

SYSTEM S1
END S1;

SYSTEM IMPLEMENTATION S1.I
END S1.I;

SYSTEM IMPLEMENTATION S2.I
END S2.I;

END P1;

PACKAGE P2
PUBLIC
WITH P1;
```

```
S RENAMES SYSTEM P1::S2;

SYSTEM IMPLEMENTATION S2.I
END S2.I;

END P2;
```

ERROR line 16: Component type or alias S2 must be declared in package P1 (ref. AADL v2 - 4.4 - N1)
ERROR line 27: Component type or alias S2 must be declared in package P2 (ref. AADL v2 - 4.4 - N1)

N2 The defining identifier of the component implementation must be unique within the local namespace of the component type.

```
PACKAGE P1
PUBLIC

SYSTEM S END S;

SYSTEM IMPLEMENTATION S.I
END S.I;

SYSTEM IMPLEMENTATION S.I
END S.I;

END P1;

PACKAGE P2
PUBLIC
WITH P1;

S RENAMES SYSTEM P1::S;

SYSTEM IMPLEMENTATION S.I
END S.I;

END P2;
```

ERROR lines 12 and 15: Implementation identifier I is not unique in the namespace of component type S (ref. AADL v2 - 4.4 - N2)

ERROR lines 12 and 26: Implementation identifier I is not unique in the namespace of component type S (ref. AADL v2 - 4.4 - N2)
ERROR lines 15 and 12: Implementation identifier I is not unique in the namespace of component type S (ref. AADL v2 - 4.4 - N2)
ERROR lines 15 and 26: Implementation identifier I is not unique in the namespace of component type S (ref. AADL v2 - 4.4 - N2)
ERROR lines 26 and 12: Implementation identifier I is not unique in the namespace of component type S (ref. AADL v2 - 4.4 - N2)
ERROR lines 26 and 15: Implementation identifier I is not unique in the namespace of component type S (ref. AADL v2 - 4.4 - N2)

N3 Every component implementation defines a *local namespace* for all defining identifiers of prototypes, subcomponents, subprogram calls, connections, flows, and modes declared within the component implementation. The defining identifier of a prototype, subcomponent, subprogram call, connection, flow, or mode must be unique within this namespace. For example, a subcomponent and a mode cannot have the same defining identifier within the same component implementation, or a mode with the same defining identifier cannot be declared in a component type and a component implementation.

```
PACKAGE P1
PUBLIC

DATA D END D;

SUBPROGRAM F
FEATURES
  A : IN PARAMETER D;
  B : OUT PARAMETER D;
END F;

SYSTEM S
FEATURES
  A : IN DATA PORT D;
  B : OUT DATA PORT D;
  E : IN EVENT PORT;
END S;

SYSTEM IMPLEMENTATION S.I
PROTOTYPES
```

```
  X : DATA D;
SUBCOMPONENTS
  X : DATA D;
CALLS X : {
  X : SUBPROGRAM F;
};
CONNECTIONS
  X : PORT A -> F.A;
  Y : PORT F.B -> B;
FLOWS
  X : FLOW PATH A -> F.A -> F.B -> B;
MODES
  X : INITIAL MODE;
  Y : MODE;
  X : X -[ E ]-> Y;
END S.I;

END P1;
```

ERROR lines 26 and 28: Identifier X must be unique for component implementation S.I (ref. AADL v2 - 4.4 - N3)
ERROR lines 28 and 26: Identifier X must be unique for component implementation S.I (ref. AADL v2 - 4.4 - N3)
ERROR lines 31 and 26: Identifier X must be unique for component implementation S.I (ref. AADL v2 - 4.4 - N3)
ERROR lines 30 and 26: Identifier X must be unique for component implementation S.I (ref. AADL v2 - 4.4 - N3)
ERROR lines 33 and 26: Identifier X must be unique for component implementation S.I (ref. AADL v2 - 4.4 - N3)
ERROR lines 34 and 39: Identifier Y must be unique for component implementation S.I (ref. AADL v2 - 4.4 - N3)
ERROR lines 38 and 26: Identifier X must be unique for component implementation S.I (ref. AADL v2 - 4.4 - N3)
ERROR lines 39 and 34: Identifier Y must be unique for component implementation S.I (ref. AADL v2 - 4.4 - N3)
ERROR lines 40 and 26: Identifier X must be unique for component implementation S.I (ref. AADL v2 - 4.4 - N3)
ERROR lines 36 and 26: Identifier X must be unique for component implementation S.I (ref. AADL v2 - 4.4 - N3)

N4 This local namespace inherits the namespace of the associated component type, i.e., defining identifiers must be unique within the local namespace and also within the component type namespace.

```
PACKAGE P1
PUBLIC

DATA D END D;

SUBPROGRAM F
FEATURES
  A : IN PARAMETER D;
  B : OUT PARAMETER D;
END F;

SYSTEM S1
FEATURES
  A : IN DATA PORT D;
  B : OUT DATA PORT D;
  E : IN EVENT PORT;
  X1 : IN EVENT PORT;
  X2 : IN EVENT PORT;
  X3 : IN EVENT PORT;
  X4 : IN EVENT PORT;
  X5 : IN EVENT PORT;
  X6 : IN EVENT PORT;
  X7 : IN EVENT PORT;
  X8 : IN EVENT PORT;
END S1;

SYSTEM S2 EXTENDS S1
END S2;

SYSTEM IMPLEMENTATION S2.I
PROTOTYPES
  X1 : DATA D;
SUBCOMPONENTS
  X2 : DATA D;
CALLS X3 : {
  X4 : SUBPROGRAM F;
};
CONNECTIONS
  X5 : PORT A -> F.A;
```

```
  Y1 : PORT F.B -> B;
FLOWS
  X6 : FLOW PATH A -> F.A -> F.B -> B;
MODES
  X7 : INITIAL MODE;
  Y2 : MODE;
  X8 : X7 -[ E ]-> Y2;
END S2.I;

END P1;
```

ERROR lines 37 and 22: Identifier X1 is already defined in component type S2 (ref. AADL v2 - 4.4 - N4)
ERROR lines 39 and 23: Identifier X2 is already defined in component type S2 (ref. AADL v2 - 4.4 - N4)
ERROR lines 42 and 24: Identifier X3 is already defined in component type S2 (ref. AADL v2 - 4.4 - N4)
ERROR lines 41 and 25: Identifier X4 is already defined in component type S2 (ref. AADL v2 - 4.4 - N4)
ERROR lines 44 and 26: Identifier X5 is already defined in component type S2 (ref. AADL v2 - 4.4 - N4)
ERROR lines 49 and 28: Identifier X7 is already defined in component type S2 (ref. AADL v2 - 4.4 - N4)
ERROR lines 51 and 29: Identifier X8 is already defined in component type S2 (ref. AADL v2 - 4.4 - N4)
ERROR lines 47 and 27: Identifier X6 is already defined in component type S2 (ref. AADL v2 - 4.4 - N4)

N5 Refinement identifiers of features must exist in the namespace of the associated component type or one of the component type's ancestors. Refinement identifiers of prototype, subcomponent, and connection refinements must exist in the local namespace of an ancestor component implementation.

```
PACKAGE P1
PUBLIC

DATA D END D;

SYSTEM S1 END S1;
```

```
SYSTEM S2 EXTENDS S1
END S2;

SYSTEM S3 EXTENDS S2
END S3;

SYSTEM IMPLEMENTATION S1.I
PROTOTYPES
  X1 : DATA D;
SUBCOMPONENTS
  X2 : DATA D;
CONNECTIONS
  X3 : PORT A -> F.A;
  X4 : PORT F.B -> B;
FLOWS
  X5 : FLOW PATH A -> F.A -> F.B -> B;
END S1.I;

SYSTEM IMPLEMENTATION S2.I EXTENDS S1.I
END S2.I;

SYSTEM IMPLEMENTATION S3.I EXTENDS S2.I
PROTOTYPES
  Y1 : REFINED TO DATA D;
SUBCOMPONENTS
  Y2 : REFINED TO DATA D;
CONNECTIONS
  Y3 : REFINED TO PORT A -> F.A;
  Y4 : REFINED TO PORT F.B -> B;
FLOWS
  Y5 : REFINED TO FLOW PATH;
END S3.I;

END P1;
```

ERROR line 36: Refinement identifier Y1 must be defined in an ancestor of component implementation S3.I (ref. AADL v2 - 4.4 - N5)
ERROR line 38: Refinement identifier Y2 must be defined in an ancestor of component implementation S3.I (ref. AADL v2 - 4.4 - N5)
ERROR line 40: Refinement identifier Y3 must be defined in an ancestor of component implementation S3.I (ref. AADL v2 - 4.4 - N5)
ERROR line 41: Refinement identifier Y4 must be defined in an ancestor of component

implementation S3.I (ref. AADL v2 - 4.4 - N5)
ERROR line 43: Refinement identifier Y5 must be defined in an ancestor of component implementation S3.I (ref. AADL v2 - 4.4 - N5)

N6 In a component implementation extension, the component type identifier of the component implementation being extended, which appears after the reserved word **extends**, must be the same as or an ancestor of the component type of the extension. The component implementation being extended may exist in another package. In this case the component implementation name is qualified with the package name.

```
PACKAGE P1
PUBLIC

SYSTEM S1 END S1;

SYSTEM S2 EXTENDS S1
END S2;

SYSTEM S3 EXTENDS S2
END S3;

SYSTEM IMPLEMENTATION S1.I
END S1.I;

SYSTEM IMPLEMENTATION S2.I EXTENDS S1.I
END S2.I;

SYSTEM IMPLEMENTATION S3.I EXTENDS S2.I
END S3.I;

END P1;

PACKAGE P2
PUBLIC
WITH P1;

RENAMES P1::ALL;

SYSTEM S4 EXTENDS S2 END S4;

SYSTEM IMPLEMENTATION S4.I EXTENDS S3.I
END S4.I;
```

```
SYSTEM IMPLEMENTATION S4.J EXTENDS S4.I
END S4.J;

END P2;
```

ERROR line 36: Component type identifier S3 must be the same as or an ancestor of S4 (ref. AADL v2 - 4.4 - N6a)

N7 When a component implementation **extends** another component implementation, the local namespace of the extension is a superset of the local namespace of the ancestor. That is, the local namespace of a component implementation inherits all the identifiers in the local namespaces of its ancestors (including the identifiers of their respective component type namespaces).

```
PACKAGE P1
PUBLIC

DATA D END D;

SUBPROGRAM F
FEATURES
  A : IN PARAMETER D;
  B : OUT PARAMETER D;
END F;

SYSTEM S
FEATURES
  A : IN DATA PORT D;
  B : OUT DATA PORT D;
  E : IN EVENT PORT;
END S;

SYSTEM IMPLEMENTATION S.I
PROTOTYPES
  X1 : DATA D;
SUBCOMPONENTS
  X2 : DATA D;
CALLS X3 : {
  X4 : SUBPROGRAM F;
};
CONNECTIONS
```

```
  X5 : PORT A -> X4.A;
  Y1 : PORT X4.B -> B;
FLOWS
  X6 : FLOW PATH A -> X4.A -> X4.B -> B;
MODES
  X7 : INITIAL MODE;
  Y2 : MODE;
  X8 : X7 -[ E ]-> Y2;
END S.I;

END P1;

PACKAGE P2
PUBLIC
WITH P1;

DATA D END D;

SUBPROGRAM F
FEATURES
  A : IN PARAMETER D;
  B : OUT PARAMETER D;
END F;

SYSTEM S EXTENDS P1::S
END S;

SYSTEM IMPLEMENTATION S.I EXTENDS P1::S.I
PROTOTYPES
  X1 : DATA D;
SUBCOMPONENTS
  X2 : DATA D;
CALLS X3 : {
  X4 : SUBPROGRAM F;
};
CONNECTIONS
  X5 : PORT A -> X4.A;
  Y1 : PORT X4.B -> B;
FLOWS
  X6 : FLOW PATH A -> X4.A -> X4.B -> B;
MODES
  X7 : INITIAL MODE;
  Y2 : MODE;
  X8 : X7 -[ E ]-> Y2;
```

```
END S.I;

END P2;
```

ERROR lines 62 and 26: Identifier X1 is already defined in an ancestor of component S.I (ref. AADL v2 - 4.4 - N7)
ERROR lines 64 and 28: Identifier X2 is already defined in an ancestor of component S.I (ref. AADL v2 - 4.4 - N7)
ERROR lines 67 and 31: Identifier X3 is already defined in an ancestor of component S.I (ref. AADL v2 - 4.4 - N7)
ERROR lines 66 and 30: Identifier X4 is already defined in an ancestor of component S.I (ref. AADL v2 - 4.4 - N7)
ERROR lines 69 and 33: Identifier X5 is already defined in an ancestor of component S.I (ref. AADL v2 - 4.4 - N7)
ERROR lines 70 and 34: Identifier Y1 is already defined in an ancestor of component S.I (ref. AADL v2 - 4.4 - N7)
ERROR lines 74 and 38: Identifier X7 is already defined in an ancestor of component S.I (ref. AADL v2 - 4.4 - N7)
ERROR lines 75 and 39: Identifier Y2 is already defined in an ancestor of component S.I (ref. AADL v2 - 4.4 - N7)
ERROR lines 76 and 40: Identifier X8 is already defined in an ancestor of component S.I (ref. AADL v2 - 4.4 - N7)
ERROR lines 72 and 36: Identifier X6 is already defined in an ancestor of component S.I (ref. AADL v2 - 4.4 - N7)

N8 Within the scope of the component implementation, subcomponent declarations, connections, subprogram call sequences, mode transitions, and property associations can refer directly to identifiers in the local namespace, i.e., to declared prototypes, subcomponents, connections, and modes, as well as to required bus, data, subprogram, and subprogram group subcomponents and features declared in the associated component type.

This rule expresses a permission and is thus not subject to checking.

N9 The prototype referenced by the prototype binding declaration must exist in the local namespace of the component implementation being extended. In other words, prototype binding declarations may bind prototypes of the component type and of

the component implementation.

```
PACKAGE P1
PUBLIC

ABSTRACT A END A;
DATA D END D;

SYSTEM S1
PROTOTYPES
  X1 : ABSTRACT A;
END S1;

SYSTEM IMPLEMENTATION S1.I
PROTOTYPES
  Y1 : ABSTRACT A;
END S1.I;

END P1;

PACKAGE P2
PUBLIC
WITH P1;

S RENAMES SYSTEM P1::S1;

SYSTEM S2 EXTENDS S
PROTOTYPES
  Z1 : ABSTRACT P1::A;
END S2;

SYSTEM IMPLEMENTATION S2.I EXTENDS S.I (
  X => DATA P1::D,
  Y => DATA P1::D,
  Z => DATA P1::D )
END S2.I;

END P2;
```

ERROR line 36: Prototype X must exist in the namespace of component implementation S2.I (ref. AADL v2 - 4.4 - N9)
ERROR line 37: Prototype Y must exist in the namespace of component implementation S2.I (ref. AADL v2 - 4.4 - N9)

ERROR line 38: Prototype Z must exist in the namespace of component implementation S2.I (ref. AADL v2 - 4.4 - N9)

3.5 Subcomponents (AS5506B-4.5)

N1 The defining identifier of a subcomponent declaration placed in a component implementation must be unique within the local namespace of the component implementation that contains the subcomponent.

This rule is already checked by rule 4.4-N3.

www.ellidiss.com

Sales office:
TNI Europe Limited
Triad House
Mountbatten Court
Worall Street
Congleton
Cheshire
CW12 1AG
UK
info@ellidiss.com
+44 1260 291 449

Technical support :
Ellidiss Technologies
24 quai de la douane
29200 Brest
Brittany
France

aadl@ellidiss.fr
+33 298 451 870