

## AADL syntax card, v1.0

<b>bold</b>	AADL keyword	{ }	repeatable
[ ]	optional term	{ }+	repeatable once or more
	alternative	::=	expansion term

**AADL\_specification** ::= { AADL\_global\_declaration | AADL\_declaration }+  
**AADL\_global\_declaration** ::= package\_spec | property\_set  
**AADL\_declaration** ::= component\_classifier | port\_group\_classifier | annex\_library  
**component\_classifier** ::= component\_type | component\_type\_extension  
   | component\_implementation | component\_implementation\_extension  
**port\_group\_classifier** ::= port\_group\_type | port\_group\_type\_extension

**package\_spec** ::= **package** defining\_package\_name  
   ( **public** package\_declaration [ **private** package\_declaration ]  
   | **private** package\_declaration )

**end** defining\_package\_name ;  
**package\_declaration** ::= { AADL\_declaration }+  
   [ **properties** ( { property\_association }+ | none\_statement ) ]  
**package\_name** ::= { package\_identifier :: } \* package\_identifier  
**none\_statement** ::= **none** ;

**component\_type** ::= component\_category defining\_component\_type\_identifier  
   [ **features** ( { feature }+ | none\_statement ) ]  
   [ **flows** ( { flow\_spec }+ | none\_statement ) ]  
   [ **properties** ( { component\_type\_property\_association }+ | none\_statement ) ]  
   { annex\_subclause } \*

**end** defining\_component\_type\_identifier ;  
**component\_type\_extension** ::= component\_category defining\_component\_type\_identifier  
   **extends** unique\_component\_type\_identifier  
   [ **features** ( { feature | feature\_refinement }+ | none\_statement ) ]  
   [ **flows** ( { flow\_spec | flow\_spec\_refinement }+ | none\_statement ) ]  
   [ **properties** ( { component\_type\_property\_association }+ | none\_statement ) ]  
   { annex\_subclause } \*

**end** defining\_component\_type\_identifier ;  
**component\_category** ::=  
   software\_category | execution\_platform\_category | composite\_category  
**software\_category** ::= **data** | **subprogram** | **thread** | **thread group** | **process**  
**execution\_platform\_category** ::= **memory** | **processor** | **bus** | **device**  
**composite\_category** ::= **system**  
**unique\_component\_type\_identifier** ::= [ package\_name :: ] component\_type\_identifier

**component\_implementation** ::= component\_category **implementation**  
   defining\_component\_implementation\_name  
   [ **refines type** ( { feature\_refinement }+ | none\_statement ) ]  
   [ **subcomponents** ( { subcomponent }+ | none\_statement ) ]  
   [ **calls** ( { subprogram\_call\_sequence }+ | none\_statement ) ]  
   [ **connections** ( { connection }+ | none\_statement ) ]  
   [ **flows** ( { flow\_implementation | end\_to\_end\_flow\_spec }+ | none\_statement ) ]  
   [ **modes** ( { mode }+ | mode\_transition } \* | none\_statement ) ]  
   [ **properties** ( { property\_association | contained\_property\_association }+  
   | none\_statement ) ]  
   { annex\_subclause } \*

**end** defining\_component\_implementation\_name ;  
**component\_implementation\_name** ::=  
   component\_type\_identifier . component\_implementation\_identifier  
**component\_implementation\_extension** ::= component\_category **implementation**  
   defining\_component\_implementation\_name  
   **extends** unique\_component\_implementation\_name  
   [ **refines type** ( { feature\_refinement }+ | none\_statement ) ]  
   [ **subcomponents**  
   ( { subcomponent | subcomponent\_refinement }+ | none\_statement ) ]  
   [ **calls** ( { subprogram\_call\_sequence }+ | none\_statement ) ]  
   [ **connections** ( { connection | connection\_refinement }+ | none\_statement ) ]  
   [ **flows** ( { flow\_implementation | flow\_implementation\_refinement  
   | end\_to\_end\_flow\_spec | end\_to\_end\_flow\_spec\_refinement }+  
   | none\_statement ) ]  
   [ **modes** ( { mode | mode\_refinement | mode\_transition }+ | none\_statement ) ]

  [ **properties** ( { property\_association }+ | none\_statement ) ]  
   { annex\_subclause } \*  
**end** defining\_component\_implementation\_name ;  
**unique\_component\_implementation\_name** ::=  
   [ package\_name :: ] component\_implementation\_name

**subcomponent** ::= defining\_subcomponent\_identifier :  
   component\_category [ component\_classifier\_reference ]  
   [ { { subcomponent\_property\_association | contained\_property\_association }+ } ]  
   [ in\_modes ] ;  
**subcomponent\_refinement** ::= defining\_subcomponent\_identifier :  
   **refined to** component\_category [ component\_classifier\_reference ]  
   [ { { subcomponent\_property\_association | contained\_property\_association }+ } ]  
   [ in\_modes ] ;  
**component\_classifier\_reference** ::=  
   unique\_component\_type\_name [ . component\_implementation\_name ]

**annex\_subclause** ::= **annex** annex\_identifier  
   { \*\* annex\_specific\_language\_constructs \*\* } ;  
**annex\_library** ::= **annex** annex\_identifier { \*\* annex\_specific\_reusable\_constructs \*\* } ;

**subprogram\_call\_sequence** ::=  
   [ defining\_call\_sequence\_identifier : ] { { subprogram\_call }+ } [ in\_modes ] ;  
**subprogram\_call** ::= defining\_call\_identifier : **subprogram** called\_subprogram  
   [ { { subcomponent\_call\_property\_association }+ } ] ;  
**called\_subprogram** ::= subprogram\_classifier\_reference  
   | data\_unique\_type\_reference . data\_subprogram\_identifier

**feature** ::= port\_spec | port\_group\_spec | server\_subprogram\_spec  
   | data\_subprogram\_spec | subcomponent\_access | parameter  
**feature\_refinement** ::= port\_refinement | port\_group\_refinement  
   | server\_subprogram\_refinement | data\_subprogram\_refinement  
   | subcomponent\_access\_refinement | parameter\_refinement

**port\_spec** ::= defining\_port\_identifier : ( **in** | **out** | **in out** ) port\_type  
   [ { { port\_property\_association }+ } ] ;  
**port\_refinement** ::= defining\_port\_identifier : **refined to** ( **in** | **out** | **in out** ) port\_type  
   [ { { port\_property\_association }+ } ] ;  
**port\_type** ::= **data port** [ data\_classifier\_reference ]  
   | **event data port** [ data\_classifier\_reference ]  
   | **event port**

**port\_group\_type** ::= **port group** defining\_identifier  
   ( **features** { port\_spec | port\_group\_spec } \*  
   | **inverse of** unique\_port\_group\_type\_reference )  
   | **inverse of** unique\_port\_group\_type\_reference )  
   [ **properties** ( { portgroup\_property\_association }+ | none\_statement ) ]  
   { annex\_subclause } \*  
**end** defining\_identifier ;

**port\_group\_type\_extension** ::= **port group** defining\_identifier  
   **extends** unique\_port\_group\_type\_reference  
   ( **features** { port\_spec | port\_refinement | port\_group\_spec  
   | port\_group\_refinement } \*  
   | **inverse of** unique\_port\_group\_type\_reference )  
   | **inverse of** unique\_port\_group\_type\_reference )  
   [ **properties** ( { portgroup\_property\_association }+ | none\_statement ) ]  
   { annex\_subclause } \*  
**end** defining\_identifier ;

**port\_group\_spec** ::= defining\_port\_group\_identifier : **port group**  
   [ unique\_port\_group\_type\_reference ] [ { { portgroup\_property\_association }+ } ] ;  
**port\_group\_refinement** ::= defining\_port\_group\_identifier : **refined to port group**  
   [ unique\_port\_group\_type\_reference ] [ { { portgroup\_property\_association }+ } ] ;  
**unique\_port\_group\_type\_reference** ::= [ package\_name :: ] port\_group\_type\_identifier

**data\_subprogram\_spec** ::= defining\_subprogram\_identifier : **subprogram**  
   [ subprogram\_classifier\_reference ] [ { { subprogram\_property\_association }+ } ] ;  
**data\_subprogram\_refinement** ::= defining\_subprogram\_identifier : **refined to**  
   **subprogram** [ subprogram\_classifier\_reference ]  
   [ { { subprogram\_property\_association }+ } ] ;

**server\_subprogram\_spec** ::= defining\_subprogram\_identifier : **server subprogram**  
   [ unique\_subprogram\_reference ] [ { { subprogram\_property\_association }+ } ] ;  
**server\_subprogram\_refinement** ::= defining\_subprogram\_identifier : **refined to**  
   **server subprogram** [ unique\_subprogram\_reference ]  
   [ { { subprogram\_property\_association }+ } ] ;  
**unique\_subprogram\_reference** ::= subprogram\_classifier\_reference |  
   data\_subprogram\_feature\_classifier\_reference  
**data\_subprogram\_feature\_classifier\_reference** ::=  
   [ package\_name :: ] data\_type\_identifier . subprogram\_identifier

**parameter** ::= defining\_parameter\_identifier : ( **in** | **out** | **in out** ) **parameter**  
   [ data\_classifier\_reference ] [ { { parameter\_property\_association }+ } ] ;  
**parameter\_refinement** ::= defining\_parameter\_identifier :  
   **refined to** ( **in** | **out** | **in out** ) **parameter** [ data\_classifier\_reference ]  
   [ { { parameter\_property\_association }+ } ] ;

**subcomponent\_access** ::= defining\_subcomponent\_access\_identifier :  
   subcomponent\_access\_classifier [ { { access\_property\_association }+ } ] ;  
**subcomponent\_access\_refinement** ::= defining\_subcomponent\_access\_identifier :  
   **refined to** subcomponent\_access\_classifier [ { { access\_property\_association }+ } ] ;

**subcomponent\_access\_classifier** ::= ( **provides** | **requires** ) ( **data** | **bus** ) **access**  
   [ unique\_component\_type\_identifier [ . component\_implementation\_name ] ]

**connection** ::= port\_connection | parameter\_connection | access\_connection  
**connection\_refinement** ::= port\_connection\_refinement  
   | parameter\_connection\_refinement | access\_connection\_refinement

**port\_connection** ::= data\_connection | event\_connection  
   | event\_data\_connection | port\_group\_connection  
**data\_connection** ::= [ defining\_data\_connection\_identifier : ]  
   **data port** source\_unique\_port\_identifier  
   ( immediate\_connection\_symbol | delayed\_connection\_symbol )  
   destination\_unique\_port\_identifier  
   [ { { property\_association }+ } ] [ in\_modes\_and\_transitions ] ;  
**immediate\_connection\_symbol** ::= ->  
**delayed\_connection\_symbol** ::= ->>  
**event\_connection** ::= [ defining\_event\_connection\_identifier : ]  
   **event port** source\_unique\_port\_identifier -> destination\_unique\_port\_identifier  
   [ { { property\_association }+ } ] [ in\_modes\_and\_transitions ] ;  
**event\_data\_connection** ::= [ defining\_event\_data\_connection\_identifier : ]  
   **event data port** source\_unique\_port\_identifier -> destination\_unique\_port\_identifier  
   [ { { property\_association }+ } ] [ in\_modes\_and\_transitions ] ;  
**port\_group\_connection** ::= [ defining\_port\_group\_connection\_identifier : ]  
   **port group** source\_unique\_port\_group\_identifier  
   -> destination\_unique\_port\_group\_identifier  
   [ { { property\_association }+ } ] [ in\_modes\_and\_transitions ] ;  
**port\_connection\_refinement** ::= connection\_identifier : **refined to**  
   ( **data port** | **event port** | **event data port** | **port group** )  
   ( ( { { property\_association }+ } [ in\_modes\_and\_transitions ] )  
   | in\_modes\_and\_transitions ) ;  
**unique\_port\_identifier** ::= component\_type\_port\_identifier  
   | subcomponent\_identifier . port\_identifier  
   | component\_type\_port\_group\_identifier . element\_port\_identifier  
**unique\_port\_group\_identifier** ::= component\_type\_port\_group\_identifier  
   | subcomponent\_identifier . port\_group\_identifier  
   | component\_type\_port\_group\_identifier . element\_port\_group\_identifier

**parameter\_connection** ::= [ defining\_parameter\_connection\_identifier : ] **parameter**  
   source\_unique\_parameter\_identifier -> destination\_unique\_parameter\_identifier  
   [ { { property\_association }+ } ] [ in\_modes ] ;  
**parameter\_connection\_refinement** ::= connection\_identifier : **refined to parameter**  
   { { property\_association }+ } [ in\_modes ] ;  
**unique\_parameter\_identifier** ::= component\_type\_parameter\_identifier  
   | subprogram\_call\_identifier . parameter\_identifier  
   | component\_type\_port\_identifier  
   | component\_type\_port\_group\_identifier . element\_port\_identifier

```

access_connection ::= [ access_connection_identifier : ] ( bus | data ) access
unique_access_provider_identifier -> unique_access_requirement_identifier
[ { { property_association }+ } ] [ in_modes ] ;
access_connection_refinement ::= connection_identifier :
  refined to ( bus | data ) access { { property_association }+ } [ in_modes ] ;
unique_access_provider_identifier ::= component_type_access_identifier
| data_or_bus_subcomponent_identifier . access_identifier
| data_or_bus_subcomponent_identifier

flow_spec ::= flow_source_spec | flow_sink_spec | flow_path_spec
flow_spec_refinement ::= flow_source_spec_refinement
| flow_sink_spec_refinement | flow_path_spec_refinement
flow_source_spec ::= defining_flow_identifier : flow source flow_feature_identifier
[ { { property_association }+ } ] ;
flow_sink_spec ::= defining_flow_identifier : flow sink flow_feature_identifier
[ { { property_association }+ } ] ;
flow_path_spec ::= defining_flow_identifier :
  flow path source_flow_feature_identifier -> sink_flow_feature_identifier
[ { { property_association }+ } ] ;
flow_source_spec_refinement ::= defining_flow_identifier : refined to flow source
{ { property_association }+ } ;
flow_sink_spec_refinement ::= defining_flow_identifier : refined to flow sink
{ { property_association }+ } ;
flow_path_spec_refinement ::= defining_flow_identifier : refined to flow path
{ { property_association }+ } ;
flow_feature_identifier ::= port_identifier | parameter_identifier | port_group_identifier
| port_group_identifier . port_identifier

flow_implementation ::= ( flow_source_implementation | flow_sink_implementation
| flow_path_implementation )
[ { { property_association }+ } ] [ in_modes_and_transitions ] ;
flow_source_implementation ::= flow_identifier : flow source
{ subcomponent_flow_identifier -> connection_identifier -> } * flow_feature_identifier
flow_sink_implementation ::= flow_identifier : flow sink flow_feature_identifier
-> connection_identifier -> subcomponent_flow_identifier *
flow_path_implementation ::= flow_identifier : flow path source_flow_feature_identifier
[ -> connection_identifier -> subcomponent_flow_identifier +
-> connection_identifier -> sink_flow_feature_identifier

flow_implementation_refinement ::= flow_source_implementation_refinement
| flow_sink_implementation_refinement | flow_path_implementation_refinement
flow_source_implementation_refinement ::= flow_identifier : refined to flow source
( { { property_association }+ } ) [ in_modes_and_transitions ]
| in_modes_and_transitions ] ;
flow_sink_implementation_refinement ::= flow_identifier : refined to flow sink
( { { property_association }+ } ) [ in_modes_and_transitions ]
| in_modes_and_transitions ] ;
flow_path_implementation_refinement ::= flow_identifier : refined to flow path
( { { property_association }+ } ) [ in_modes_and_transitions ]
| in_modes_and_transitions ] ;
subcomponent_flow_identifier ::= subcomponent_identifier . flow_spec_identifier

end_to_end_flow_spec ::= defining_end_to_end_flow_identifier : end to end flow
start_subcomponent_flow_identifier
{ -> connection_identifier -> flow_path_subcomponent_flow_identifier } *
-> connection_identifier -> end_subcomponent_flow_identifier
[ { { property_association }+ } ] [ in_modes_and_transitions ] ;
end_to_end_flow_refinement ::= defining_end_to_end_identifier :
  refined to end to end flow
( { { property_association }+ } ) [ in_modes_and_transitions ]
| in_modes_and_transitions ] ;

property_set ::= property set defining_property_set_identifier is
{ property_type_declaration | property_name_declaration | property_constant }+
end defining_property_set_identifier ;

property_type_declaration ::=

```

```

  defining_property_type_identifier : type property_type_designator ;
property_type_designator ::= property_type | unique_property_type_identifier
property_type ::= aadboolean | aadstring | enumeration_type | units_type
| number_type | range_type | classifier_type | reference_type
enumeration_type ::= enumeration ( defining_enumeration_literal_identifier
{ , defining_enumeration_literal_identifier } * )
units_type ::= units units_list
units_list ::= ( defining_unit_identifier
{ , defining_unit_identifier => unit_identifier * numeric_literal } * )
number_type ::= aadreal [ real_range ] [ units units_designator ]
| aadinteger [ integer_range ] [ units units_designator ]
units_designator ::= units_unique_property_type_identifier | units_list
real_range ::= real_lower_bound .. real_upper_bound
real_lower_bound ::= signed_aadreal_or_constant
real_upper_bound ::= signed_aadreal_or_constant
integer_range ::= integer_lower_bound .. integer_upper_bound
integer_lower_bound ::= signed_aadinteger_or_constant
integer_upper_bound ::= signed_aadinteger_or_constant
signed_aadreal_or_constant ::=
( signed_aadreal [ + | - ] real_property_constant_term )
signed_aadinteger_or_constant ::=
( signed_aadinteger [ + | - ] integer_property_constant_term )
signed_aadinteger ::= [ + | - ] integer_literal [ unit_identifier ]
signed_aadreal ::= [ + | - ] real_literal [ unit_identifier ]
range_type ::= range of number_type
| range of number_unique_property_type_identifier
classifier_type ::= classifier ( { component_category { , component_category } * } )
reference_type ::=
  reference ( { referable_element_category { , referable_element_category } * } )
referable_element_category ::=
  component_category | connections | server subprogram
unique_property_type_identifier ::= [ property_set_identifier :: ] property_type_identifier

property_name_declaration ::= defining_property_name_identifier : [ access ] [ inherit ]
( single_valued_property | multi_valued_property )
applies to ( ( property_owner_category { , property_owner_category } * | all ) ) ;
single_valued_property ::= property_type_designator [ => default_property_expression ]
multi_valued_property ::= list of property_type_designator
[ => ( { default_property_expression { , default_property_expression } * } ) ]
property_owner_category ::= component_category [ classifier_reference ] | mode
| port group flow | [ event ] [ data ] port | server subprogram | parameter
| [ connection_type ] connections
connection_type ::= port group | [ event ] [ data ] port | access | parameter

property_constant ::=
  single_valued_property_constant | multi_valued_property_constant
single_valued_property_constant ::= defining_property_constant_identifier : constant
( ( aadinteger | aadreal ) [ units_unique_property_type_identifier ]
| aadstring | aadboolean | enumeration_unique_property_type_identifier
| integer_range_unique_property_type_identifier
| real_range_unique_property_type_identifier
| integer_unique_property_type_identifier | real_unique_property_type_identifier )
=> constant_property_value ;
multi_valued_property_constant ::= defining_property_constant_identifier : constant
list of ( ( aadinteger | aadreal ) [ units_unique_property_type_identifier ]
| aadstring | aadboolean | enumeration_unique_property_type_identifier
| integer_range_unique_property_type_identifier
| real_range_unique_property_type_identifier
| integer_unique_property_type_identifier | real_unique_property_type_identifier )
=> ( [ constant_property_value { , constant_property_value } * ] ) ;
constant_property_value ::= string_literal | signed_integer | signed_real
| boolean_value | enumeration_identifier
| signed_aadinteger .. signed_aadinteger [ delta signed_aadinteger ]
| signed_aadreal .. signed_aadreal [ delta signed_aadreal ]
unique_property_constant_identifier ::=
  value ( [ property_set_identifier :: ] property_constant_identifier )

```

```

property_association ::= [ property_set_identifier :: ] property_name_identifier
( => | +=> ) [ constant ] property_value [ in_modes ] ;
access_property_association ::= [ property_set_identifier :: ] property_name_identifier
( => | +=> ) [ constant ] access property_value [ in_modes ] ;
contained_property_association ::= [ property_set_identifier :: ] property_name_identifier
( => | +=> ) [ constant ] property_value
  applies to contained_unit_identifier { . contained_unit_identifier } *
  [ in_modes ] [ in_modes ] ;
property_value ::= single_property_value | property_list_value
single_property_value ::= property_expression
property_list_value ::= ( [ property_expression { , property_expression } * ] )
in_binding ::=
  in binding ( platform_classifier_reference { , platform_classifier_reference } * )
platform_classifier_reference ::= processor_classifier_reference
| memory_classifier_reference | bus_classifier_reference

property_expression ::= boolean_term | real_term | integer_term | string_term
| enumeration_term | real_range_term | integer_range_term | property_term
| component_classifier_term | reference_term
boolean_term ::= boolean_value | boolean_property_constant_term | not boolean_term
| boolean_term and boolean_term | boolean_term or boolean_term | ( boolean_term
)
boolean_value ::= true | false
real_term ::= signed_aadreal_or_constant
integer_term ::= signed_aadinteger_or_constant
string_term ::= string_literal | string_property_constant_term
enumeration_term ::= enumeration_identifier | enumeration_property_constant_term
integer_range_term ::= integer_term .. integer_term [ delta integer_term ]
| integer_range_property_constant_term
real_range_term ::= real_term .. real_term [ delta real_term ]
| real_range_property_constant_term
property_term ::= value ( [ property_set_identifier :: ] property_name_identifier )
property_constant_term ::=
  value ( [ property_set_identifier :: ] property_constant_identifier )
component_classifier_term ::= component_category
[ unique_component_type_identifier [ . component_implementation_identifier ] ]
reference_term ::= reference subcomponent_identifier [ . subcomponent_identifier ] *
| { subcomponent_identifier . }+ connection_identifier
| { subcomponent_identifier . }+ server_subprogram_identifier

mode ::= defining_mode_identifier : [ initial ] mode [ { { mode_property_association }+ } ] ;
mode_transition ::= source_mode_identifier { , source_mode_identifier } *
- [ unique_port_identifier { , unique_port_identifier } * ] -> destination_mode_identifier ;
mode_refinement ::= defining_mode_identifier : refined to mode
{ { mode_property_association }+ } ;
in_modes ::= in modes ( ( mode_identifier { , mode_identifier } * | none ) )
in_modes_and_transitions ::=
  in modes ( ( mode_or_transition { , mode_or_transition } * | none ) )
mode_or_transition ::= mode_identifier | ( old_mode_identifier -> new_mode_identifier )

```

© Axlog Ingénierie, 2005  
This document can be freely copied and distributed.  
http://www.axlog.fr/R\_d/aad/