AADL V2 Syntax

bold AADL keyword	{ }* repeatable
[ ] optional term	{ }+ repeatable once or more
| alternative	::= expansion term

```
-- Packages to structure component specs into libraries
AADL_model ::= package_spec | property_set

package_spec ::=
package package_name
( public package_content
| public package_content private package_content
| private package_content )
[ properties ( { property_association }+ | none_statement ) ]
end package_name ;

package_name ::= { package_identifier :: }* package_identifier

package_content ::=
{ component_type | component_implementation | feature_group_type | annex_library }+

none_statement ::= none ;

component_category ::=
data | subprogram | thread | thread group | process | system | abstract memory | processor | bus | device | virtual processor | virtual bus

-- Component interface specification
component_type ::=
component_category component_type_identifier
[ extends component_type_reference [ prototype_bindings ] ]
[ prototypes ( { prototype }+ | none_statement ) ]
[ features ( { feature }+ | none_statement ) ]
[ flows ( { flow_spec }+ | none_statement ) ]
[ modes ( { mode }+ { mode_transition }* | none_statement ) ]
[ properties ( { component_type_property_association }+ | none_statement ) ]
{ annex_subclause }*
end component_type_identifier ;

-- Component implementation blue print
component_implementation ::=
component_category implementation component_type_identifier .
component_implementation_identifier
[ extends component_implementation_reference [ prototype_bindings ] ]
[ prototypes ( { prototype }+ | none_statement ) ]
[ subcomponents ( { subcomponent }+ | none_statement ) ]
[ calls ( { subprogram_call_sequence }+ | none_statement ) ]
[ connections ( { connection }+ | none_statement ) ]
[ flows ( { flow_implementation | end_to_end_flow }+ | none_statement ) ]
[ modes ( { mode }+ { mode_transition }* | none_statement ) ]
[ properties ( { property_association | contained_property_association }+ | none_statement ) ]
{ annex_subclause }*
end component_type_identifier .
component_implementation_identifier ;

-- Component classifier references
component_classifier_reference ::=
component_type_reference |
component_implementation_reference

component_type_reference ::= [
package_name :: ]
component_type_identifier

component_implementation_reference ::=
[ package_name :: ]
component_type_identifier .
component_implementation_identifier

-- subcomponent instance in a composite component implementation
subcomponent ::=
subcomponent_identifier :
component_category [
component_classifier_reference [
prototype_bindings ] | prototype_identifier ]
[ refined to ] component_category [
component_classifier_reference ] [
prototype_bindings ] ]
   [ array_dimensions [
array_element_implementation_list ] ]
[ { { subcomponent_property_association | contained_property_association }+ } ] [
in_modes ] ;

array_dimensions ::=
   { array_dimension }+

array_dimension ::=
   [ [numeral | property_constant_identifier | property_identifier ] ]

array_element_implementation_list ::=
   ( component_implementation_reference { , component_implementation_reference }* )

-- array selection used in contained proper association and references
array_selection_identifier ::=
   identifier { [ numeral [ .. numeral ] ] }+

-- Component interaction points
feature ::=
abstract_feature | port | data_access |
bus_access | subprogram_access |
subprogram_group_access | parameter |
feature_group

-- refinable feature
abstract_feature ::=
( [  in | out ] feature
[data_classifier_reference |
data_prototype_identifier |
feature_prototype_identifier ] )
[ array_dimension ]
[ { { feature_property_association }+ } ] ;

-- Interaction point for directional flow of data, events, and messages
port ::=
port_identifier : [ refined to ] ( in | out | in out ) data port [ data_classifier_reference ] | event port | event data port [
data_classifier_reference ]
[ array_dimension ]
 [ { { port_property_association }+ } ] ;

-- Shared data access interaction point
data_access ::=
data_access_identifier :  [ refined to ] ( provides |  requires ) data access
      [
data_unique_component_classifier_reference | prototype_identifier ]
[ array_dimension ]
 [ { { data_access_property_association }+ }
] ;

-- Bus access interaction point
bus_access ::=
bus_access_identifier :  [ refined to ] ( provides |  requires ) bus access
      [
bus_unique_component_classifier_reference | prototype_identifier ]
[ array_dimension ]
[ { { bus_access_property_association }+ } ]
;

-- Callable subprogram interaction point
subprogram_access ::=
subprogram_access_identifier :  [ refined to ]
( provides |  requires ) subprogram access
      [
subprogram_unique_component_classifier_reference | prototype_identifier ]
[ array_dimension ]
[ { {
subprogram_access_property_association }+
} ] ;

-- Subprogram library access interaction point
subprogram_group_access ::=
data_access_identifier :  [ refined to ] ( provides |  requires ) subprogram group access
      [
subprogram_group_unique_component_classifier_reference | prototype_identifier ]
[ array_dimension ]
[ { {
subprogram_group_access_property_association }+ } ] ;

-- Subprogram parameter
parameter ::=
parameter_identifier : [ refined to ] ( in | out | in out ) parameter
[ data_classifier_reference |
prototype_identifier]
[ { { parameter_property_association }+ } ] ;

-- collection of features as one interaction point
feature_group_spec ::=
feature_group_identifier : [ refined to ] [ in | out ] feature group [ [ inverse of ]
feature_group_type_reference |
prototype_identifier ]
[ array_dimension ]
[ { { feature_group_property_association }+ }
] ;

feature_group_type_reference ::= [
package_name :: ]
feature_group_type_identifier

-- type specification of a collection of features
feature_group_type ::=
feature group feature_group_type_identifier
[ extends feature_group_type_reference ]
[ prototype ( { prototype }+ |
none_statement ) ]
[ features { feature }+ ]
```

[**inverse of**
feature_group_type_reference]
[**properties** ({
*feature_group_*property_association }+ |
none_statement)]
{ annex_subclause }*
end *feature_group_type* _identifier **;**

-- Connections between features
connection ::=
abstract_feature_connection |
port_connection | parameter_connection |
access_connection |
feature_group_connection

abstract_feature_connection ::=
[*abstract_feature_connection_*identifier **:**] [
refined to]
*source_*feature_connection_reference (**->** |
<->)
*destination_*feature_connection_reference
[**{** {
*abstract_feature_connection_*property_associ
ation }+ **}**]
[in_modes_and_transitions] **;**

feature_connection_reference ::=
-- feature in the component type
*component_type_feature_*identifier |
-- feature in a feature group of the
component type
*component_type_feature_group_*identifier **.**
*feature_*identifier |
-- feature in a subcomponent
*subcomponent_*identifier **.**
*feature_*identifier

port_connection ::=
[*port_connection_*identifier **:**] [**refined to**]
port *source_*port_reference (**->** | **<->**)
*destination_*port_connection_reference
[**{** { *port_connection_*property_association }+
}]
[in_modes_and_transitions] **;**

port_connection_reference ::=
-- port in the component type
*component_type_port_*identifier |
-- port in a subcomponent
*subcomponent_*identifier **.** *port_*identifier |
-- port element in a feature group of
the component type
*component_type_feature_group_*identifier**.**
*element_port_*identifier |
-- data element in aggregate data
port
*component_type_port_*identifier **.**
*data_element_*identifier |
-- requires data access in the
component type

*component_type_requires_data_access_*identifi
er |
-- data subcomponent
*data_subcomponent_*identifier |
-- data component provided by a
subcomponent
*subcomponent_*identifier **.**
*provides_data_access_*identifier |
-- data access element in a feature
group of the component type
*component_type_feature_group_*identifier **.**
*element_data_access_*identifier |
-- access to element in a data
subcomponent
*data_subcomponent_*identifier **.**
*data_subcomponent_*identifier |
-- processor port
processor . *processor_port_*identifier |
-- component itself as event or event
data source
self .
*event_or_event_data_source_*identifier

parameter_connection ::=
[*parameter_connection_*identifier **:**] [**refined
to**] **parameter**
*source_*parameter_connection_reference **-
>**
*destination_*parameter_connection_referen
ce
[**{** {
*paramter_connection_*property_association
}+ **}**]
[in_modes_and_transitions] **;**

parameter_connection_reference ::=
-- parameter in the thread or
subprogram type
*component_type_parameter_*identifier [**.**
*parameter_*identifier] |
-- parameter in another subprogram
call
*subprogram_call_*identifier **.**
*parameter_*identifier |
-- data or event data port in the
thread type or an element of that port's
data
*component_type_port_*identifier [**.**
*data_subcomponent_*identifier] |
-- data subcomponent in the thread
or subprogram
*data_subcomponent_*identifier |
-- requires data access in the thread
or subprogram type
*requires_data_access_*identifier |
-- data access element in a feature
group of the component type
*component_type_feature_group_*identifier **.**
*element_data_access_*identifier |
-- port or parameter element in a
feature group of the component type
*component_type_feature_group_*identifier [
. *element_port_*identifier]

data_access_connection ::=
[*data_access_connection_*identifier **:**] [
refined to] **data access**
*source_data_*access_connection_reference
(**->** | **<->**)
*destination_data_*access_connection_refere
nce
[**{** {
*data_access_connection_*property_associatio
n }+ **}**]
[in_modes_and_transitions] **;**

bus_access_connection ::=
[*bus_access_connection_*identifier **:**] [
refined to] **bus access**
*source_bus_*access_connection_reference
(**->** | **<->**)
*destination_bus_*access_connection_referen
ce
[**{** {
*bus_access_connection_*property_association
}+ **}**]
[in_modes_and_transitions] **;**

subprogram_access_connection ::=
[*subprogram_access_connection_*identifier **:**]
[**refined to**] **data access**
*source_subprogram_*access__connection_re
ference (**->** | **<->**)
*destination_subprogram_*access_connection_
reference
[**{** {
*suprogram_access_connection_*property_asso
ciation }+ **}**]
[in_modes_and_transitions] **;**

subprogram_group_access_connection
::=
[
*subprogram_group_access_connection_*identifi
er **:**] [**refined to**] **data access**
*source_subprogram_group_*access_conenctio
n_reference (**->** | **<->**)
*destination_subprogram_group_*access_conne
ction_reference
[**{** {
*data_access_connection_*property_associatio
n }+ **}**]
[in_modes_and_transitions] **;**

access_connection_reference ::=
-- requires or provides access feature
in the component type
*requires_access_*identifier |
*provides_access_*identifier |
-- requires or provides access feature
a feature group of the component type
*feature_group_*identifier [**.**
*requires_access_*identifier] |
*feature_group_*identifier [**.**
*provides_access_*identifier] |
-- provides or requires access in a
subcomponent
*subcomponent_*identifier **.**
*provides_access_*identifier |
*subcomponent_*identifier **.**
*requires_access_*identifier |
-- data, subprogram, subprogram
group or bus being accessed

*data_subprogram_subprogam_group_or_bus_su
bcomponent_*identifier |
-- subprogram a processor being
accessed
processor .
*provides_subprogram_access_*identifier

feature_group_connection ::=
[*feature_group_connection_*identifier **:**] [
refined to] **bus access**
*source_feature_group_*connection_reference
<->
*destination_feature_group_*connection_refere
nce
[**{** {
*feature_group_connection_*property_associati
on }+ **}**]
[in_modes_and_transitions] **;**

feature_group_reference ::=
-- feature group in the component
type
*component_type_feature_group_*identifier |
-- feature group in a subcomponent
*subcomponent_*identifier **.**
*feature_group_*identifier |
-- feature group element in a feature
group of the component type
*component_type_feature_group_*identifier **.**
*element_feature_group_*identifier

-- call sequences within a thread or
subprogram
subprogram_call_sequence ::=
*call_sequence_*identifier **: {** {
subprogram_call }+ **}**
[in_modes] **;**

subprogram_call ::=
*subprogram_call_*identifier **: subprogram**
called_subprogram
[**{** {
*subcomponent_call_*property_association }+
}] **;**

```
called_subprogram ::=
   -- identification by classifier

subprogram_component_classifier_referenc
e |
   data_component_type_reference .
data_provides_subprogram_access_identifier |

subprogram_group_component_type_referen
ce . provides_subprogram_access_identifier |
   -- identification by prototype
   subprogram_prototype_identifier |
   -- identification by processor
subprogram access feature
   processor .
provides_subprogram_access_identifier |
   -- identification by subprogram instance
   subprogram_subcomponent_identifier |
   subprogram_group_subcomponent_identifier
. provides_subprogram_access_identifier |
   requires_subprogram_access_identifier |

requires_subprogram_group_access_identifier .
provides_subprogram_access_identifier

-- flow specifications from an incoming to
an outgoing feature of a component
flow_spec ::= flow_source_spec |
flow_sink_spec | flow_path_spec

flow_source_spec ::=
flow_identifier : [ refined to ] flow source
out_flow_feature_identifier
[ { { property_association }+ } ]
[ in_modes ] ;
-- refined to has no flow feature identifier

flow_sink_spec ::=
flow_identifier : [ refined to ] flow sink
in_flow_feature_identifier
[ { { property_association }+ } ]
[ in_modes ] ;
-- refined to has no flow feature identifier

flow_path_spec ::=
flow_identifier : [ refined to ] flow path
in_flow_feature_identifier ->
out_flow_feature_identifier
[ { { property_association }+ } ]
[ in_modes ] ;
-- refined to has no flow feature identifiers

flow_feature_identifier ::=
feature_identifier | feature_group_identifier .
feature_identifier | feature_group_identifier .
feature_group_identifier

-- realization of a flow specification in the
component implementation
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_reference ->
connection_identifier -> }*
out_flow_feature_identifier

flow_sink_implementation ::=
flow_identifier : flow sink
flow_feature_identifier { ->
connection_identifier ->
subcomponent_flow_reference }*

flow_path_implementation ::=
flow_identifier : flow path
in_flow_feature_identifier [ { ->
connection_identifier ->
subcomponent_flow_identifier }+ ->
connection_identifier ] ->
out_flow_feature_identifier

-- flow can go through subcomponent
flows and through data components
subcomponent_flow_reference::=
subcomponent_identifier . flow_spec_identifier
| data_subcomponent_identifier |
requires_data_access_identifier |
provides_data_access_identifier

-- end-to-end flows can be composed of
other end-to-end flows and subcomponent
flows
end_to_end_flow ::=
end_to_end_flow_identifier : [ refined to ]
end to end flow
start_subcomponent_flow_or_etef_identifie
r { -> connection_identifier ->
flow_path_subcomponent_flow_or_etef_ref
erence }* -> connection_identifier ->
end_subcomponent_flow_or_etef_referenc
e
[ { ( property_association }+ } ]
[ in_modes_and_transitions ] ;
-- refined to has no flow start to end path

subcomponent_flow_or_etef_reference ::=
subcomponent_flow_identifier |
end_to_end_flow_identifier

-- modes and mode transitions
mode ::=
mode_identifier : [ refined to ] [ initial ]
mode
[ { { mode_property_assocation }+ } ] ;

mode_transition ::=
[ mode_identifier : ] source_mode_identifier -[
mode_transition_trigger { ,
mode_transition_trigger }* ]->
destination_mode_identifier ;

mode_transition_trigger ::=
   port_identifier | subcomponent_identifier .
port_identifier | self . event_source_identifier
| processor . port_identifier

-- mode-specific subcomponents,
connections, flows, annexes
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 )

-- name-binding of inheritable modes
component_in_modes ::=
   in modes ( ( mode_name { ,
mode_name }* ) | all )

mode_name ::=
local_mode_identifier [ =>
subcomponent_mode_identifier ]


-- sublanguage annotation
annex_subclause ::=
annex annex_identifier {**
annex_specific_language_constructs **} [
in_modes ] ;

annex_library ::=
annex annex_identifier {**
annex_specific_reusable_constructs **} [
in_modes ];

-- parameterization of component types
and component implementations
prototype ::=
   prototype_identifier : (
component_prototype |
feature_group_type_prototype |
feature_prototype )
      [ { { prototype_property_association }+
} ] ;

component_prototype ::=
[ refined to ] component_category [
component_classifier_reference ] [ [] ]

feature_group_type_prototype ::=
   feature group [
feature_group_type_reference ]

feature_prototype ::=
   [ in | out ] feature [
component_classifier_reference ]

prototype_bindings ::=
   ( prototype_binding { , prototype_
binding }* )

prototype_binding ::=
   prototype_identifier =>
      ( component_prototype_actual |
component_prototype_actual_list
      | feature_group_type_prototype_actual
| feature_prototype_actual )

component_prototype_actual ::=
   component_category (
component_classifier_reference [
prototype_bindings ] | prototype_identifier )

component_prototype_actual_list ::=
   ( component_prototype_actual { ,
component_prototype_actual }* )

feature_group_type_prototype_actual ::=
      ( feature group
feature_group_type_reference )
      | (feature group
feature_group_type_prototype_identifier )

feature_prototype_actual ::=
      (( in | out | in out) ( event | data |
event data ) port ) |
      ( ( requires | provides )
         ( bus | data | subprogram group |
subprogram ) access )
         component_classifier_reference )
      | ( [ in | out ] feature
feature_prototype_identifier )



-- Assigning property values
property_association ::=
[ property_set_identifier :: ]
property_name_identifier ( => | +=> ) [
constant ] assignment
[ in_binding ] ;

contained_property_association ::=
[ property_set_identifier :: ]
property_name_identifier ( => | +=> ) [
constant ] assignment
applies to
contained_model_path_element_identifier { .
contained_ model_path_element _identifier }*
```

```
[ in_binding ] ;

-- path to model element to which property
association applies. Any named model
element can be part of path
contained_model_element_path ::=
    contained_model_element { .
contained_model_element }* [ annex_path
] |
    annex_path

contained_model_element ::=
    named_element_identifier |
named_element_array_selection_identifier
    -- array selection only applies to named
elements that support multiplicity. In core
AADL subcomponent & feature

-- path leading to model elements in an
annex subclause
annex_path ::=
    annex annex_identifier {** <annex
specific path> **}
    -- It is recommended this path follows
the dot-separation syntax of the
component path.

-- deployment binding specific property
values, e.g., processor type specific
execution time. Platform components are
processor, bus, memory.
in_binding ::= in binding (
platform_component_classifier_reference {
, platform_component_classifier_reference
}* )

-- values(s) to be assigned
assignment ::= property_value |
modal_property_value

modal_property_value ::= ( property_value
in_modes , }* property_value [ in_modes ]
)

property_value ::= property_expression | (
[ property_expression { ,
property_expression }* ] )

-- single property value
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 | record_term |
computed_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_aadlreal_or_constant

integer_term ::=
signed_aadlinteger_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 ::= [ property_set_identifier ::
] property_identifier

property_constant_term ::= [
property_set_identifier :: ]
property_constant_identifier

component_classifier_term ::= classifier (
component_classifier_reference )

reference_term ::= reference (
contained_model_element_path )

record_term ::= ( record_field_identifier =>
property_value ; { record_field_identifier
=> property_value ; }*  )

computed_term ::= compute (
function_identifier )

-- Defining new properties
property_set ::=
    property set property_set_identifier is
        { import_declaration }*
        { property_type_definition |
property_definition |
property_constant_definition }*
    end property_set_identifier ;

--  property type is defined in terms of a
base type or inline type constructor, or by
referring to another property type
property_type_definition ::=
property_type_identifier : type
basic_property_type |
property_type_constructor ;

-- built-in basic types and type
constructors
basic_property_type ::= aadlboolean |
aadlstring | aadlreal | aadlinteger

property_type_constructor ::=
enumeration_type | units_type |
number_type | range_type | classifier_type
| reference_type

enumeration_type ::= enumeration (
enumeration_literal_identifier { ,
enumeration_literal_identifier }* )

units_type ::= units ( defining_unit_identifier
{ , defining_unit_identifier => unit_identifier *
numeric_literal }* )

number_type ::=
aadlreal units units_designator  |
aadlreal real_range | aadlreal real_range
units units_designator |
aadlinteger units units_designator |
aadlinteger integer_range | aadlinteger
integer_range units units_designator|

units_designator ::=
units_property_type_reference | units_list

-- integer and real ranges
real_range ::=
signed_aadlreal_or_constant ..
signed_aadlreal_or_constant

integer_range ::=
signed_aadlreal_or_constant ..
signed_aadlreal_or_constant

signed_aadlreal_or_constant ::=
[ + | - ] integer_literal [ unit_identifier ] |
 [ + | - ] real_property_constant_term

signed_aadlinteger_or_constant ::=
([ + | - ] real_literal [ unit_identifier ] |
[ + | - ] integer_property_constant_term )

range_type ::= range of aadlreal | range
of aadlinteger | range of
number_property_type_reference

classifier_type ::= classifier [ (
classifier_category_reference { ,
classifier_category_reference }* ) ]

-- any Meta model element that is a
subclass of classifier including those of
annex sublanguages
classifier_category_reference ::= [ {
annex_identifier }** ]
meta_model_class_identifier

reference_type ::= reference [ (
named_element_meta_model_identifier { ,
named_element_meta_model_identifier }* ) ]

record_type ::=  record ( { field_identifier :
[ list of ] property_type_reference ; }* )

-- all properties other than predeclared
properties must be qualified by their
property set
property_type_reference ::= [
property_set_identifier :: ]
property_type_identifier

property_type_designator ::=
basic_property_type |
property_type_constructor |
property_type_reference

-- new user defined property
property_definition ::= property_identifier : [
inherit ] ( single_valued_property |
multi_valued_property )
applies to ( ( property_owner { ,
property_owner }* | 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 ::=
named_element_meta_model_identifier  |
component_classifier_reference |
feature_group_type_reference

-- property constant with optional default
value. The value must match the type
property_constant ::=
property_constant_identifier : constant
property_type_designator =>
property_value ;
```