SECTION NAME	ID	RULE TEXT	POS	NEG	COMMENTARY
AADL Specifications	p41n1	(N1) An AADL specification has one global namespace. The package and property set identifiers reside in this space and	0	0	Checked by DFS during processing package nodes
AADL Specifications	p41n2	(N2) These package and property set identifiers qualify the names of individual elements contained in them when they ar	0	0	0
AADL Specifications	p41n3	(N3) Package declarations represent labeled namespaces for component type, component implementation, feature grou	0	0	0
AADL Specifications	p41n4	(N4) Property set declarations represent labeled namespaces for property type and property definition declarations.	0	0	0
AADL Specifications	p41n5	(N5) Packages and property sets may be separately stored. Those packages and property sets are considered to be par	0	0	Provided by parser
AADL Specifications	p41n6	(N6) Defining identifiers in AADL must not be one of the reserved words of the language (see Section 15.7).	0	0	Provided by parser
AADL Specifications	p41n7	(N7) The AADL identifiers and reserved words can be in upper or lower case (or a mixture of the two) (see Section 15).	0	0	Provided by realization of AADLIdentifer class
AADL Specifications	p41n8	(N8) The AADL does not require that an identifier be declared before it is referenced.	0	0	0
Packages	p42n1	(N1) A defining package name consists of a sequence of one or more package identifiers separated by a double colon (e	0	0	Checked by counting private and public package declarations in DFS during processing package nodes
Packages	p42n2	(N2) The public and private section of a package may be declared in separate package declarations; these two declaratio	0	0	Provided by parser
Packages	p42n3	(N3) Associated with every package is a package namespace that contains the names for all the elements defined within	0	0	0
Packages	p42n4	(N4) The package namespace is divided into a public section and a private section. Items declared in the public section c	0	0	0
Packages	p42n5	(N5) The reference to an item declared in another package must be an item name qualified with a package name separa	0	0	Can be checked after all possible references are known
Packages	p42n6	(N6) The reference to a property other than predeclared properties must be an property name qualified with a property se	0	0	Can be checked after all possible references are known
Packages	p42n7	(N7) The package name in a import_declaration must exist in the global name space.	0	0	Checked by DFS during processing package nodes - imports
Packages	p42n8	(N8) The property set identifier in a import_declaration must exist in the global name space.	0	0	Checked by DFS during processing package nodes - imports
Packages	p42n9	(N9) Items declared in the private section of the package can only be referenced from within the private section of the pa	0	0	Can be checked after all possible references are known
Packages	p42n10	(N10) If the qualifying package identifier of a qualified reference is missing, the referenced component classifier, feature g	0	0	0
Packages	p42n11	(N11) The package name referenced in an alias_declaration must exist in the global namespace and must be listed in the	0	0	Checked by DFS during processing package nodes - package aliases
Packages	p42n12	(N12) The classifier referenced by the alias_declaration must exist in the name space of the public section of the packag	0	0	Checked by DFS during processing package nodes - classifier aliases
Packages	p42n13	(N13) The classifier referenced by the alias declaration must refer to a component type or a feature group type.	0	0	Provided by parser
Packages	p42n14	(N14) The defining identifier of an alias_declaration must be unique in the namespace of the package containing the alias	0	0	Checked by DFS during processing package nodes - package and classifier aliases
Packages	p42n15	(N15) The alias_declaration makes the publicly visible identifier of classifiers declared in another package accessible in t	0	0	0
Packages	p42n16	(N16) If the alias_declaration renames all publicly visible identifiers of component types and feature group types by nami	0	0	0
Packages	p42n17	(N17) The identifiers introduced by the alias_declaration are only accessible within the package. When declared in the pu	0	0	Not compatible with current realization of N14 check (alias ids == component ids), not all possible references are known
Packages	p42n18	(N18) The alias declared for a component type can be used instead of a qualified component type in a reference to a cor	0	0	0
Packages	p42l1	(L1) The defining package name following the reserved word end must be identical to the defining package name followi	0	0	Provided by parser
Packages	p42l2	(L2) For each package there may be at most one public section declaration and one private section declaration. These tv	0	0	Checked with p42n1
Packages	p42l3	(L3) A component implementation may be declared in both the public and private part of a package. In that case the decl	0	0	Should be checked in component implementations
Packages	p42l4	(L4) The component category in an alias declaration must match the category of the referenced component type.	0	0	0
Component Types	p43n1	(N1) The defining identifier for a component type must be unique in the namespace of the package within which it is decl	0	0	Checked when creating local namespaces of the packages
Component Types	p43n2	(N2) Each component type has a local namespace for defining identifiers of prototypes, features, modes, mode transition	0	0	0
Component Types	p43n3	(N3) The component type identifier of the ancestor in a component type extension, i.e., that appears after the reserved w	0	0	Checked by DFS for each component type node
Component Types	p43n4	(N4) When a component type extends another component type, a component type namespace includes all the identifiers	0	0	0
Component Types	p43n5	(N5) A component type that extends another component type does not include the identifiers of the implementations of its	0	0	0
Component Types	p43n6	(N6) The defining identifier of a feature, flow specification, mode, mode transition, or prototype must be unique in the nan	0	0	0
Component Types	p43n7	(N7) The refinement identifier of a feature, flow specification, or prototype refinement refers to the closest refinement or tl	0	0	0
Component Types	p43n8	(N8) The prototypes referenced by prototype binding declarations must exist in the local namespace of the component ty	0	0	0
Component Types	p43n9	(N9) Mode transitions declared in the component type may not refer to event or event data ports of subcomponents.	0	0	0
Component Types	p43l1	(L1) The defining identifier following the reserved word end must be identical to the defining identifier that appears after tl	0	0	Provided by parser
Component Types	p43l2	(L2) The prototypes, features, flows, modes, and properties subclauses are optional. If a subclause is present but empty,	0	0	Provided by parser(kinda, error is - "No viable alternative")
Component Types	p43l3	(L3) The category of the component type being extended must match the category of the extending component type, i.e.,	0	0	Checked by DFS for each component type node
Component Types	p43l4	(L4) The classifier being extended in a component type extension may include prototype bindings. There must be at mos	0	0	0
Component Types	p43l5	(L5) A component type must not contain both a requires_modes_subclause and a modes_subclause.	0	0	Provided by parser(kinda, error is - extraneous input 'requires' expecting {'ANNEX', 'END', 'PROPERTIES', IDENTIFIER})
Component Types	p43l6	(L6) If the extended component type and an ancestor component type in the extends hierachy contain modes subclauses	0	0	0
Component Implementatior	p44n1	(N1) A component implementation name consists of a component type identifier and a component implementation identif	0	0	0
Component Implementatior	p44n2	(N2) The defining identifier of the component implementation must be unique within the local namespace of the compone	0	0	0
Component Implementatior	p44n3	(N3) Every component implementation defines a local namespace for all defining identifiers of prototypes, subcomponent	0	0	0
Component Implementatior	p44n4	(N4) This local namespace inherits the namespace of the associated component type, i.e., defining identifiers must be ur	0	0	0
Component Implementatior	p44n5	(N5) Refinement identifiers of features must exist in the namespace of the associated component type or one of the com	0	0	0
Component Implementatior	p44n6	(N6) In a component implementation extension, the component type identifier of the component implementation being ex	0	0	0
Component Implementatior	p44n7	(N7) When a component implementation extends another component implementation, the local namespace of the extens	0	0	0

Component Implementatior	p44n8	(N8) Within the scope of the component implementation, subcomponent declarations, connections, subprogram call sequ	0	0	0
Component Implementatior	p44n9	(N9) The prototype referenced by the prototype binding declaration must exist in the local namespace of the component	0	0	0
Component Implementatior	p44l1	(L1) The pair of identifiers separated by a dot (вЂњ.вЂќ) following the reserved word end must be identical to the pair of	0	0	0
Component Implementatior	p44l2	(L2) The prototypes, subcomponents, connections, calls, flows, modes, and properties subclauses are optional. If they ar	0	0	0
Component Implementatior	p44l3	(L3) The category of the component implementation must be identical to the category of the component type for which th	0	0	0
Component Implementatior	p44l4	(L4) If the component implementation extends another component implementation, the category of both must match, i.e.,	0	0	0
Component Implementatior	p44l5	(L5) The classifier being extended in a component implementation extension may include prototype bindings. There mus	0	0	0
Component Implementatior	p44l6	(L6) If the component type of the component implementation contains a requires_modes_subclause then the component	0	0	0
Component Implementatior	p44l7	(L7) If modes are declared in the component type, then modes cannot be declared in component implementations.	0	0	0
Component Implementatior	p44l8	(L8) If modes or mode transitions are declared in the component type, then mode transitions can be added in the compo	0	0	0
Component Implementatior	p44l9	(L9) The category of a subcomponent being refined must match the category of the refining subcomponent declaration, i.	0	0	0
Component Implementatior	p44l10	(L10) For all other refinement declarations the categories must match (see the respective sections).	0	0	0
Component Implementatior	p44l11	(L11) Component implementations and component implementation extensions must not refine prototypes declared in a c	0	0	0
Subcomponents	p45n1	(N1) The defining identifier of a subcomponent declaration placed in a component implementation must be unique within	0	0	0
Subcomponents	p45n2	(N2) The defining identifier of a subcomponent refinement must exist as a defining subcomponent identifier in the local na	0	0	0
Subcomponents	p45n3	(N3) The component type identifier or the component implementation name of a component classifier reference must exi	0	0	0
Subcomponents	p45n4	(N4) The prototype identifier of a prototype reference must exist in the local name space of the component implementation.	0	0	0
Subcomponents	p45n5	(N5) The prototype referenced by the prototype binding declarations must exist in the local namespace of the component	0	0	0
Subcomponents	p45n6	(N6) The modes named in the in modes statement of a subcomponent must refer to modes in the component implementa	0	0	0
Subcomponents	p45l1	(L1) The category of the subcomponent declaration must match the category of its corresponding component classifier re	0	0	0
Subcomponents	p45l2	(L2) The component classifier reference of a subcomponent declaration may include prototype bindings for a subset or a	0	0	0
Subcomponents	p45l3	(L3) In a subcomponent refinement declaration the component category may be refined from abstract to one of the concr	0	0	0
Subcomponents	p45l4	(L4) The Classifier_Substitution_Rule property specifies the rule to be applied when a refinement supplies a classifier an	0	0	0
Subcomponents	p45l5	(L5) In the case of a signature match, the component type of the subcomponent being refined must have a subset of the	0	0	0
Subcomponents	p45l6	(L6) The component category and optional component classifier or prototype reference can be followed by a set of array	0	0	0
Subcomponents	p45l7	(L7) The array size specification for the dimensions is optional. In this case the array declaration is considered incomplet	0	0	0
Subcomponents	p45l8	(L8) When refining a subcomponent array the number of dimensions of the array cannot be changed, but the array size c	0	0	0
Subcomponents	p45l9	(L9) When the subcomponent is declared as an array with array dimension sizes then a list of component implementatior	0	0	0
Subcomponents	p45l10	(L10) Selecting index ranges in one or more dimensions of an array is only possible if the size of the array for these dime	0	0	0
Subcomponents	p45l11	(L11) An array element implementation list is valid only if (a) the subcomponent classifier is a component type and (b) all	0	0	0
Subcomponents	p45c1	(C1) The classifier of a subcomponent cannot recursively contain subcomponents with the same component classifier. In	0	0	0
Abstract Components	p46l1	(L1) An abstract component type declaration can contain feature declarations (including abstract feature declarations), flo	0	0	0
Abstract Components	p46l2	(L2) An abstract component implementation can contain subcomponent declarations of any category. Certain combinatio	0	0	0
Abstract Components	p46l3	(L3) An abstract component implementation can contain a modes subclause, a connections subclause, a flows subclause	0	0	0
Abstract Components	p46l4	(L4) An abstract subcomponent can be contained in the implementation of any component category.	0	0	0
Abstract Components	p46l5	(L5) If an abstract subcomponent is refined to a concrete category, the concrete category must be acceptable to the com	0	0	0
Abstract Components	p46l6	(L6) An abstract subcomponent can be declared as an array of subcomponents.	0	0	0
Abstract Components	p46l7	(L7) If an abstract component type is refined to a concrete category, the features, modes, and flow specifications of the a	0	0	0
Abstract Components	p46l8	(L8) If an abstract component implementation is refined to a concrete category, the subcomponents, call sequences, mod	0	0	0
Prototypes	p47n1	(N1) The prototype identifier on the left-hand side of a prototype binding must exist in the local namespace of the classifie	0	0	0
Prototypes	p47n2	(N2) The prototype identifier on the right-hand side of a prototype binding, if present, must exist in the local namespace o	0	0	0
Prototypes	p47n3	(N3) Unique component classifier references must exist in the public section of the package being identified in the refere	0	0	0
Prototypes	p47n4	(N4) Unique feature group type references must exist in the public section of the package being identified in the reference.	0	0	0
Prototypes	p47l1	(L1) The component category declared in the component prototype binding must match the component category of the p	0	0	0
Prototypes	p47l2	(L2) The component category of the optional component classifier reference in the prototype declaration must match the	0	0	0
Prototypes	p47l3	(L3) If the component prototype only specifies a component category, then any component type and component impleme	0	0	0
Prototypes	p47l4	(L4) If the component prototype declaration includes a component classifier reference, then the classifier supplied in the	0	0	0
Prototypes	p47l5	(L5) The category of the component implementation that contains the prototype declaration places restrictions on the set	0	0	0
Prototypes	p47l6	(L6) If the direction is declared for feature prototypes, then the prototype actual satisfies the direction according to the sa	0	0	0
Prototypes	p47l7	(L7) In the case of feature group prototypes, the supplied feature group types must match the declared feature group typ	0	0	0
Prototypes	p47l8	(L8) A classifier supplied in a feature prototype binding must match the classifier of the prototype declaration, if present, a	0	0	0
Prototypes	p47l9	(L9) Component prototypes declared with square brackets specify that they expect a list of component classifiers. These	0	0	0
Prototypes	p47l10	(L10) The component category of the classifier reference or prototype reference in a prototype binding declaration must r	0	0	0

Prototypes	p47l11	(L11) If a direction is specified for an abstract feature in a prototype declaration, then the direction of the prototype actual	0	0	0
Prototypes	p47l12	(L12) Component prototype bindings must only bind component prototypes, feature group prototype bindings must only b	0	0	0
Prototypes	p47l13	(L13) Component prototype refinements must only refine component prototypes, feature group prototype refinements mu	0	0	0
Annex Subclauses and Anr	p48n1	(N1) The annex identifier must be the name of an approved annex or a project-specific identifier different from the approv	0	0	0
Annex Subclauses and Anr	p48n2	(N2) The mode identifiers in the in_modes statement must refer to modes in the component type or component implemer	0	0	0
Annex Subclauses and Anr	p48l1	(L1) Annex subclauses can only be declared in component types, component implementations, and feature group types.	0	0	0
Annex Subclauses and Anr	p48l2	(L2) A component type, component implementation, or feature group type declaration may contain at most one annex su	0	0	0
Annex Subclauses and Anr	p48l3	(L3) Annex libraries must be declared in packages.	0	0	0
Annex Subclauses and Anr	p48l4	(L4) A package declaration may contain at most one annex library declaration for each annex.	0	0	0
Data	p51l1	(L1) A data type declaration can contain provides subprogram access declarations as well as property associations.	0	0	0
Data	p51l2	(L2) A data type declaration must not contain a flow specification or modes subclause.	0	0	0
Data	p51l3	(L3) A data implementation can contain abstract, data and subprogram subcomponents, access connections, and data p	0	0	0
Data	p51l4	(L4) A data implementation must not contain a flow implementation, an end-to-end flow specification, or a modes subclau	0	0	0
Subprograms and Subprog	p52n1	(N1) The defining identifier of a subprogram call sequence declaration must be unique within the local namespace of the	0	0	0
Subprograms and Subprog	p52n2	(N2) The defining identifier of a subprogram call declaration must be unique within the local namespace of the componer	0	0	0
Subprograms and Subprog	p52n3	(N3) If the called subprogram name is a subprogram classifier reference, its component type identifier or component impl	0	0	0
Subprograms and Subprog	p52n4	(N4) The subprogram classifier reference of a subprogram call may be a subprogram type reference.	0	0	0
Subprograms and Subprog	p52n5	(N5) If the called subprogram name is a subprogram subcomponent reference, the subprogram subcomponent must exis	0	0	0
Subprograms and Subprog	p52n6	(N6) If the called subprogram name is a requires subprogram access reference, the requires subprogram access must ex	0	0	0
Subprograms and Subprog	p52l1	(L1) A subprogram type declaration can contain parameter, out event port, out event data port, and feature group declara	0	0	0
Subprograms and Subprog	p52l2	(L2) A subprogram implementation can contain abstract, subprogram, and data subcomponents, a subprogram calls sub	0	0	0
Subprograms and Subprog	p52l3	(L3) Only one subprogram call sequence can apply to a given mode.	0	0	0
Subprograms and Subprog	p52c1	(C1) The reference to a provides subprogram access of a processor in a subprogram call (processor . provides_subprog	0	0	0
Subprograms and Subprog	p52c2	(C2) A subprogram call may reference a subprogram classifier. A project may enforce a consistency rule that this referen	0	0	0
Subprogram Groups and S	p53n1	(N1) The defining identifier of a subprogram group type must be unique within the package namespace of the package w	0	0	0
Subprogram Groups and S	p53n2	(N2) Each subprogram group provides a local namespace. The defining subprogram identifiers of subprogram declaratio	0	0	0
Subprogram Groups and S	p53n3	(N3) The local namespace of a subprogram group type extension includes the defining identifiers in the local namespace	0	0	0
Subprogram Groups and S	p53n4	(N4) The defining subprogram identifiers of subprogram access feature declarations in feature group refinements must n	0	0	0
Subprogram Groups and S	p53n5	(N5) The package name of the unique subprogram group type reference must refer to a package name in the global nam	0	0	0
Subprogram Groups and S	p53l1	(L1) A subprogram group type can contain provides and requires subprogram access, and provides and requires subpro	0	0	0
Subprogram Groups and S	p53l2	(L2) A subprogram group implementation can contain abstract, data, subprogram group, and subprogram subcomponent	0	0	0
Subprogram Groups and S	p53l3	(L3) A subprogram group type or implementation may contain zero or more subcomponent declarations. If it contains zer	0	0	0
Threads	p54l1	(L1) A thread type declaration can contain port, feature group, requires data access declarations, as well as requires and	0	0	0
Threads	p54l2	(L2) A thread component implementation can contain abstract, data, subprogram, and subprogram group subcomponent	0	0	0
Threads	p54l3	(L3) The Complete out event port, and Error out event data port are predeclared, i.e., are implicitly identifiers in the name	0	0	0
Threads	p54c3	(C3) Either the Compute_Entrypoint, Compute_Entrypoint_Source_Text Compute_Entrypoint_Call_Sequence property n	0	0	0
Threads	p54c4	(C4) The Period property must have a value if the Dispatch_Protocol property value is periodic, sporadic, timed, or hybrid.	0	0	0
Thread Groups	p55l1	(L1) A thread group component type can contain provides and requires data access, as well as port, feature group, provi	0	0	0
Thread Groups	p55l2	(L2) A thread group component implementation can contain abstract, data, subprogram, subprogram group, thread, and	0	0	0
Thread Groups	p55l3	(L3) A thread group implementation can contain a connections subclause, a flows subclause, a modes subclause, and pr	0	0	0
Thread Groups	p55l4	(L4) A thread group must not contain a subprogram calls subclause.	0	0	0
Processes	p56l1	(L1) A process component type can contain port, feature group, provides and requires data access, provides and require	0	0	0
Processes	p56l2	(L2) A process component implementation can contain abstract, data, subprogram, subprogram group, thread, and threa	0	0	0
Processes	p56l3	(L3) A process implementation can contain a connections subclause, a flows subclause, a modes subclause, and a prop	0	0	0
Processes	p56l4	(L4) A thread group must not contain a subprogram calls subclause.	0	0	0
Processes	p56c1	(C1) The complete source text associated with a process component must form a complete and legal program as defined	0	0	0
Processors	p61l1	(L1) A processor component type can contain port, feature group, provides subprogram access, provides subprogram gr	0	0	0
Processors	p61l2	(L2) A processor component implementation can contain declarations of memory, bus, virtual bus, virtual processor, and	0	0	0
Processors	p61l3	(L3) A processor implementation can contain a modes subclause, flows subclause, and a properties subclause.	0	0	0

Processors	p61l4	(L4) A processor implementation can contain bus access, subprogram access, subprogram group access, port, feature, a	0	0	0
Processors	p61l5	(L5) A processor implementation must not contain a subprogram calls subclause.	0	0	0
Virtual Processors	p62l1	(L1) A virtual processor component type can contain port, feature group, provides subprogram access, and subprogram g	0	0	0
Virtual Processors	p62l2	(L2) A virtual processor component implementation can contain declarations of virtual bus, virtual processor, and abstrac	0	0	0
Virtual Processors	p62l3	(L3) A virtual processor implementation can contain a modes subclause, flows subclause, and a properties subclause.	0	0	0
Virtual Processors	p62l4	(L4) A virtual processor implementation must not contain a subprogram calls subclause.	0	0	0
Virtual Processors	p62l5	(L5) A virtual processor implementation can contain subprogram access, subprogram group access, port, feature, and fe	0	0	0
Virtual Processors	p62c1	(C1) In a fully bound system every virtual processor must be directly or indirectly bound to, or directly or indirectly contain	0	0	0
Virtual Processors	p62c2	(C2) In a fully deployed system a requires virtual bus binding of a virtual processor specified by the Required_Virtual_Bus	0	0	0
Memory	p63l1	(L1) A memory type can contain bus access declarations, feature groups, a modes subclause, and property associations	0	0	0
Memory	p63l2	(L2) A memory implementation can contain abstract, memory, and bus subcomponent declarations.	0	0	0
Memory	p63l3	(L3) A memory implementation can contain a modes subclause and property associations.	0	0	0
Memory	p63l4	(L4) A memory implementation can contain bus access connection declarations. Bus access connections can connect a	0	0	0
Memory	p63l5	(L5) A memory implementation must not contain flows subclause, or subprogram calls subclause.	0	0	0
Buses	p64l1	(L1) A bus type can have requires bus access declarations, a modes subclause, and property associations.	0	0	0
Buses	p64l2	(L2) A bus type must not contain any flow specifications.	0	0	0
Buses	p64l3	(L3) A bus implementation can contain virtual bus and abstract subcomponent declarations.	0	0	0
Buses	p64l4	(L4) A bus implementation can contain a modes subclause and property associations.	0	0	0
Buses	p64l5	(L5) A bus implementation must not contain flows subclause, or subprogram calls subclause.	0	0	0
Virtual Buses	p65l1	(L1) A virtual bus type can have property associations.	0	0	0
Virtual Buses	p65l2	(L2) A virtual bus type must not contain flow specifications.	0	0	0
Virtual Buses	p65l3	(L3) A virtual bus implementation can contain virtual bus subcomponent declarations.	0	0	0
Virtual Buses	p65l4	(L4) A virtual bus implementation can contain a modes subclause and property associations.	0	0	0
Virtual Buses	p65l5	(L5) A virtual bus implementation must not contain a connections subclause, flows subclause, or subprogram calls subcla	0	0	0
Virtual Buses	p65c1	(C1) In a fully deployed system virtual buses must be directly or indirectly bound to processors or buses that support thes	0	0	0
Devices	p66l1	(L1) A device type can contain port, feature group, provides subprogram access, provides subprogram group access, bu	0	0	0
Devices	p66l2	(L2) A device component implementation must not contain a subprogram calls subclause.	0	0	0
Devices	p66l3	(L3) A device implementation can contain abstract, data, virtual bus, and bus subcomponents, bus access connections, a	0	0	0
Systems	p71l1	(L1) A system component type can contain subprogram, subprogram group, data and bus access declarations, port, feat	0	0	0
Systems	p71l2	(L2) A system component implementation can contain abstract, data, subprogram, subprogram group, process, and syst	0	0	0
Systems	p71l3	(L3) A system implementation can contain a modes subclause, a connections subclause, a flows subclause, and property	0	0	0
Systems	p71l4	(L4) A thread group must not contain a subprogram calls subclause.	0	0	0
Systems	p71n1	(N1) The defining identifier of a feature must be unique within the namespace of the associated component type.	0	0	0
Systems	p71n2	(N2) Thread features may not be declared using the predeclared ports names Complete or Error.	0	0	0
Systems	p71n3	(N3) Each refining feature identifier that appears in a feature refinement declaration must also appear in a feature declara	0	0	0
Systems	p71n4	(N4) A feature is referenced in one of two ways. Within the component implementations for a component type, a feature	0	0	0
Systems	p71n5	(N5) The path of a contained property association for a feature must refer to an element of a feature group.	0	0	0
Systems	p71l1	(L1) Each feature can be refined at most once in the same type extension.	0	0	0
Systems	p71l2	(L2) A feature refinement declaration of a feature and the original feature must both be declared as port, parameter, acce	0	0	0
Systems	p71l3	(L3) Feature arrays must only be declared for threads, devices, and processors.	0	0	0
Systems	p71l4	(L4) If the feature refinement specifies an array dimension, then the feature being refined must have an array dimension.	0	0	0
Systems	p71l5	(L5) If the refinement specifies an array dimension size, then the feature being refined must not have an array dimension	0	0	0
Systems	p71l6	(L6) A contained property association must only be used when the feature is a feature group.	0	0	0
Systems	p71l7	(L7) In the case of a feature with a classifier reference, the classifier of the refined feature declaration in a component typ	0	0	0
Abstract Features	p81l1	(L1) The feature direction in a refined feature declaration must be identical to the feature direction in the feature declarati	0	0	0
Abstract Features	p81l2	(L2) If the direction of an abstract feature is specified, then the direction must be satisfied by the refinement (see also the	0	0	0
Abstract Features	p81l3	(L3) An abstract feature with a feature prototype identifier and the prototype being referenced must both specify the same	0	0	0
Abstract Features	p81l4	(L4) An abstract feature refinement declaration of a feature with a feature prototype reference must only add property ass	0	0	0

Feature Groups and Featur	p82n1	(N1) The defining identifier of a feature group type must be unique within the package namespace of the package where	0	0	0
Feature Groups and Featur	p82n2	(N2) Each feature group type provides a local namespace. The defining identifiers of prototype, feature, and feature grou	0	0	0
Feature Groups and Featur	p82n3	(N3) The local namespace of a feature group type extension includes the defining identifiers in the local namespace of th	0	0	0
Feature Groups and Featur	p82n4	(N4) The defining feature identifiers of feature group declarations must be unique in the local name space of the compon	0	0	0
Feature Groups and Featur	p82n5	(N5) The defining feature group identifier of feature_refinement declarations in component types must exist in the local na	0	0	0
Feature Groups and Featur	p82n6	(N6) The package name of the unique feature group type reference must refer to a package name in the global namespa	0	0	0
Feature Groups and Featur	p82n7	(N7) The prototype reference in a feature group declaration must refer to a prototype of the component type or feature gr	0	0	0
Feature Groups and Featur	p82l1	(L1) A feature group type may contain zero or more elements, i.e., feature or feature groups. If it contains zero elements,	0	0	0
Feature Groups and Featur	p82l2	(L2) A feature group type can be declared to be the inverse of another feature group type, as indicated by the reserved w	0	0	0
Feature Groups and Featur	p82l3	(L3) Only feature group types without inverse of or feature group types with features and inverse of can be extended.	0	0	0
Feature Groups and Featur	p82l4	(L4) A feature group type that is an extension of another feature group type without an inverse of cannot contain an inver	0	0	0
Feature Groups and Featur	p82l5	(L5) The feature group type that is an extension of another feature group type with features and inverse of that adds featu	0	0	0
Feature Groups and Featur	p82l6	(L6) A feature group declaration with an inverse of statement must only reference feature group types without an inverse	0	0	0
Feature Groups and Featur	p82l7	(L7) A feature group refinement may be refined to only add property associations. In this case inclusion of the feature gro	0	0	0
Feature Groups and Featur	p82l8	(L8) The number of feature or feature groups contained in the feature group and its complement must be identical;	0	0	0
Feature Groups and Featur	p82l9	(L9) Each of the declared features or feature groups in a feature group must be a pair-wise complement with that in the fe	0	0	0
Feature Groups and Featur	p82l10	(L10) If both feature group types have zero features, then they are considered to complement each other;	0	0	0
Feature Groups and Featur	p82l11	(L11) Ports are pair-wise complementary if they satisfy the port connection rules specified in Section 9.2.1. This includes	0	0	0
Feature Groups and Featur	p82l12	(L12) Access features are pair-wise complementary if they satisfy the access connection rules in Section 9.4.	0	0	0
Feature Groups and Featur	p82l13	(L13) If an in or out direction is specified as part of a feature group declaration, then all features inside the feature group	0	0	0
Ports	p83n1	(N1) A defining port identifier must adhere to the naming rules specified for all features (see Section 8).	0	0	0
Ports	p83n2	(N2) The defining identifier of a port refinement declaration must also appear in a feature declaration of a component type	0	0	0
Ports	p83n3	(N3) The unique component type identifier of the data classifier reference must be the name of a data component type. T	0	0	0
Ports	p83n4	(N4) The prototype identifier of a prototype reference, if specified, must exist in the namespace of the component type or	0	0	0
Ports	p83l1	(L1) Ports can be declared in subprogram, thread, thread group, process, system, processor, virtual processor, and devic	0	0	0
Ports	p83l2	(L2) Data and event data ports may be incompletely defined by not specifying the data component classifier reference or	0	0	0
Ports	p83l3	(L3) Data, event, and event data ports may be refined by adding a property association. The data component classifier d	0	0	0
Ports	p83l4	(L4) The port category of a port refinement must be the same as the category of the port being refined, or the port being	0	0	0
Ports	p83l5	(L5) The port direction of a port refinement must be the same as the direction of the feature being refined. If the feature b	0	0	0
Subprogram and Subprogra	p84n1	(N1) The defining identifier of a provides or requires subprogram or subprogram group access declaration must be uniqu	0	0	0
Subprogram and Subprogra	p84n2	(N2) The defining identifier of a provides or requires subprogram or subprogram group refinement must exist as a definin	0	0	0
Subprogram and Subprogra	p84n3	(N3) The component type identifier or component implementation name of a subprogram or subprogram group access cl	0	0	0
Subprogram and Subprogra	p84n4	(N4) The prototype identifier of a subprogram or subprogram group access classifier reference, if present, must exist in th	0	0	0
Subprogram and Subprogra	p84l1	(L1) If a subprogram access refers to a component classifier or a component prototype, then the category of the classifie	0	0	0
Subprogram and Subprogra	p84l2	(L2) If a subprogram group access refers to a component classifier or a component prototype, then the category of the cl	0	0	0
Subprogram and Subprogra	p84l3	(L3) An abstract feature can be refined into a subprogram access or a subprogram group access. In this case, the abstra	0	0	0
Subprogram and Subprogra	p84l4	(L4) A subprogram or subprogram group access declaration that does not specify a component classifier reference is inc	0	0	0
Subprogram and Subprogra	p84l5	(L5) A subprogram or subprogram group access declaration may be refined by adding a property association. Inclusion o	0	0	0
Subprogram and Subprogra	p84l6	(L6) A provides subprogram access cannot be refined to a requires subprogram access and a requires subprogram acce	0	0	0
Subprogram and Subprogra	p84c1	(C1) A provides subprogram access feature indicates that a subprogram is made available to be referenced. A project ma	0	0	0
Subprogram Parameters	p85n1	(N1) The defining identifier of a parameter must be unique within the namespace of the subprogram type containing the p	0	0	0
Subprogram Parameters	p85n2	(N2) The defining parameter identifier of a parameter refinement declaration must also appear in a feature declaration of	0	0	0
Subprogram Parameters	p85n3	(N3) The data classifier reference must refer to a data component type or a data component implementation.	0	0	0
Subprogram Parameters	p85n4	(N4) The prototype identifier, if present, must exist in the namespace of the subprogram classifier that contains the paran	0	0	0
Subprogram Parameters	p85l1	(L1) Parameters can be declared for subprogram component types.	0	0	0
Subprogram Parameters	p85l2	(L2) A parameter declaration that does not specify a data classifier reference is incomplete. Such a reference can be add	0	0	0
Subprogram Parameters	p85l3	(L3) A parameter declaration may be refined by adding a property association. Inclusion of the data classifier reference is	0	0	0
Subprogram Parameters	p85l4	(L4) The parameter direction of a parameter refinement must be the same as the direction of the feature being refined. If	0	0	0
Data Component Access	p86n1	(N1) The defining identifier of a provides or requires data access declaration must be unique within the namespace of the	0	0	0
Data Component Access	p86n2	(N2) The defining identifier of a provides or requires data access refinement must exist as a defining identifier of a provid	0	0	0
Data Component Access	p86n3	(N3) The component type identifier or component implementation name of a data access classifier reference must exist i	0	0	0
Data Component Access	p86n4	(N4) The prototype identifier, if present, must exist in the namespace of the classifier that contains the data access decla	0	0	0

Data Component Access	p86l1	(L1) If a data access refers to a component classifier or a component prototype, then the category of the classifier or prot	0	0	0
Data Component Access	p86l2	(L2) A data access declaration may be refined by refining the data classifier, by adding a property association, or by doin	0	0	0
Data Component Access	p86l3	(L3) A provides data access cannot be refined to a requires data access and a requires data access cannot be refined to	0	0	0
Data Component Access	p86l4	(L4) An abstract feature can be refined into a data access. In this case, the abstract feature must not have a direction spe	0	0	0
Data Component Access	p86c1	(C1) A data access declaration that does not specify a data classifier reference is incomplete. Such a reference can be a	0	0	0
Data Component Access	p86c2	(C2) If the source code of a component does access shared data, then the component type declaration must specify a re	0	0	0
Data Component Access	p86c3	(C3) A data access refinement may refine an abstract feature declaration. If the abstract feature declaration specifies a d	0	0	0
Bus Component Access	p87n1	(N1) The defining identifier of a provides or requires bus access declaration must be unique within the namespace of the	0	0	0
Bus Component Access	p87n2	(N2) The defining identifier of a provides or requires bus refinement must exist as a defining identifier of a requires or pro	0	0	0
Bus Component Access	p87n3	(N3) The component type identifier or component implementation name of a bus access classifier reference must exist in	0	0	0
Bus Component Access	p87n4	(N4) The prototype identifier, if present, must exist in the namespace of the classifier that contains the bus access declar	0	0	0
Bus Component Access	p87l1	(L1) If a bus access refers to a component classifier or a component prototype, then the category of the classifier or proto	0	0	0
Bus Component Access	p87l2	(L2) A bus access declaration may be refined by refining the bus classifier, by adding a property association, or by doing	0	0	0
Bus Component Access	p87l3	(L3) A provides bus access cannot be refined to a requires bus access and a requires bus access cannot be refined to a	0	0	0
Bus Component Access	p87l4	(L4) An abstract feature can be refined into a bus access. In this case, the abstract feature must not have a direction spe	0	0	0
Bus Component Access	p87c1	(C1) A bus access declaration that does not specify a bus classifier reference is incomplete. Such a reference can be ad	0	0	0
Bus Component Access	p87c2	(C2) If a bus access feature is a refinement of an abstract feature, then the direction of the abstract feature, if specified, i	0	0	0
Bus Component Access	p87n1	(N1) The defining identifier of a defined connection declaration must be unique in the local namespace of the component	0	0	0
Bus Component Access	p87n2	(N2) The connection identifier in a connection refinement declaration must refer to a named connection declared in an ar	0	0	0
Bus Component Access	p87l1	(L1) A connection refinement must contain at least one of the following: a connection source and destination subclause, a	0	0	0
Bus Component Access	p87l2	(L2) If a semantic connection may be active in a particular mode, then the ultimate source and ultimate destination comp	0	0	0
Bus Component Access	p87l3	(L3) If a semantic connection may be active in a particular mode transition, then the ultimate source component must be	0	0	0
Feature Connections	p91n1	(N1) A source or destination reference in a feature connection or feature connection refinement declaration must referen	0	0	0
Feature Connections	p91n2	(N2) The subcomponent reference may refer to a subcomponent or a subcomponent array.	0	0	0
Feature Connections	p91l1	(L1) If the feature connection declaration represents a connection between features of sibling components, then the sour	0	0	0
Feature Connections	p91l2	(L2) If the feature connection declaration represents a connection between features up the containment hierarchy, then th	0	0	0
Feature Connections	p91l3	(L3) If the feature connection declaration represents a connection between features down the containment hierarchy, the	0	0	0
Feature Connections	p91l4	(L4) If the feature connection declaration specifies a directional connection, then the direction of the connection must be s	0	0	0
Feature Connections	p91l5	(L5) The individual connections of a semantic connection must be bidirectional or have the same direction. The direction	0	0	0
Port Connections	p92n1	(N1) The connection identifier in a port connection refinement declaration must refer to a named port or feature connectio	0	0	0
Port Connections	p92n2	(N2) A source or destination reference in a port connection or port connection refinement declaration must reference a po	0	0	0
Port Connections	p92n3	(N3) The subcomponent reference may also consist of a reference to a subcomponent array.	0	0	0
Port Connections	p92n4	(N4) The event_or_event_data identifier of event source specifications (self.event_or_event_data_identifier) must not cor	0	0	0
Port Connections	p92l1	(L1) In the case of a directional port connection the connection end representing the source of the flow must be the sourc	0	0	0
Port Connections	p92l2	(L2) In the case of a bidirectional port connection either connection end can be the source. If the bidirectional connection	0	0	0
Port Connections	p92l3	(L3) If the source connection end is a data access feature it must have read access rights; if the destination connection e	0	0	0
Port Connections	p92l4	(L4) The feature identifier of a subcomponent reference may refer to a feature array, if the subcomponent is a thread, dev	0	0	0
Port Connections	p92l5	(L5) The following are acceptable sources and destinations of port connections. The left column shows connections betw	0	0	0
Port Connections	p92l6	(L6) If the port connection declaration represents a connection between ports of sibling components, then the source mus	0	0	0
Port Connections	p92l7	(L7) If the port connection declaration represents a connection between ports up the containment hierarchy, then the sou	0	0	0
Port Connections	p92l8	(L8) If the port connection declaration represents a connection between ports down the containment hierarchy, then the s	0	0	0
Port Connections	p92l9	(L9) The individual connections of a semantic port connection must be bidirectional or have the same direction. The direc	0	0	0
Port Connections	p92l10	(L10) Self.<identifier> must only be referenced as the source of a connection.	0	0	0
Port Connections	p92l11	(L11) A data port cannot be the destination of more than one semantic port connection unless each semantic port conne	0	0	0
Port Connections	p92l12	(L12) A semantic connection cannot contain connection declarations with both immediate and delayed Timing property va	0	0	0
Port Connections	p92l13	(L13) For connections between data ports, event data ports and data access, the data classifier of the source port must m	0	0	0
Port Connections	p92l14	(L14) The following rules are supported: вЂў вЂў вЂў вЂў Classifier_Match: The source data type and data implementa	0	0	0
Port Connections	p92l15	(L15) If more than one port connection declaration in a semantic port connection has a property association for a given co	0	0	0
Port Connections	p92l16	(L16) A processor port specification must only be used in event connections within threads and subprograms.	0	0	0
Port Connections	p92c1	(C1) There cannot be cycles of immediate connections between threads, devices, and processors.	0	0	0
Port Connections	p92c2	(C2) The processor port identifier of a processor port specification (processor.processor_port_identifier) must name a por	0	0	0
Port Connections	p92c3	(C3) The Supports_Classifier_Subset_Matches property may be associated with a bus or virtual bus. This specifies the s	0	0	0
Port Connections	p92c4	(C4) The Supports_Type_Conversions property may be associated with a bus or virtual bus. This specifies the subset ma	0	0	0

Parameter Connections	p93n1	(N1) The connection identifier in a parameter connection refinement declaration must refer to a named parameter or feat	0	0	0
Parameter Connections	p93n2	(N2) A source (destination) reference in a parameter connection declaration must reference a parameter of a preceding (	0	0	0
Parameter Connections	p93l1	(L1) The source of a parameter connection must be an incoming data or event data port of the containing thread, an inco	0	0	0
Parameter Connections	p93l2	(L2) The following source/destination pairs are acceptable for parameter connection declarations: threadport -> call.para	0	0	0
Parameter Connections	p93l3	(L3) A parameter cannot be the destination feature reference of more than one parameter connection declaration unless	0	0	0
Parameter Connections	p93l4	(L4) The data classifier of the source and destination must match. The matching rules as specified by the Classifier_Matc	0	0	0
Access Connections	p94n1	(N1) The connection identifier in an access connection refinement declaration must refer to a named access or feature c	0	0	0
Access Connections	p94n2	(N2) An access reference in an access connection declaration must reference an access feature of a subcomponent, sub	0	0	0
Access Connections	p94l1	(L1) The category of the source and the destination of a access connection declaration must be the same, i.e., they must	0	0	0
Access Connections	p94l2	(L2) In the case of a bidirectional semantic access connection either connection end can be the source.	0	0	0
Access Connections	p94l3	(L3) In the case of a directional data or bus access connection the connection end representing the component being ac	0	0	0
Access Connections	p94l4	(L4) In a partial AADL model the ultimate source or destination may be a provides access feature of a component instead	0	0	0
Access Connections	p94l5	(L5) If the access connection declaration represents an access connection between access features of sibling componen	0	0	0
Access Connections	p94l6	(L6) If the access connection declaration represents a feature mapping up the containment hierarchy, then one connectio	0	0	0
Access Connections	p94l7	(L7) If the access connection declaration represents a feature mapping down the containment hierarchy, then one conne	0	0	0
Access Connections	p94l8	(L8) A requires access cannot be the source or destination feature reference of more than one access connection declar	0	0	0
Access Connections	p94l9	(L9) For access connections the classifier of the provider access must match to the classifier of the requires access acco	0	0	0
Access Connections	p94l10	(L10) If more than one access feature in a semantic access connection has an Access_Right property association, then t	0	0	0
Access Connections	p94l11	(L11) The category of the access connection source and destination must be identical. If the component category is spec	0	0	0
Feature Group Connections	p95n1	(N1) The connection identifier in a feature group connection refinement declaration must refer to a feature group named	0	0	0
Feature Group Connections	p95n2	(N2) A source or destination reference in a feature group connection declaration must reference a feature group declared	0	0	0
Feature Group Connections	p95l1	(L1) If the feature group connection declaration represents a component connection between sibling components, the fea	0	0	0
Feature Group Connections	p95l2	(L2) The Classifier_Matching_Rule property specifies the rule to be applied to match the feature group classifier of a con	0	0	0
Feature Group Connections	p95l3	(L3) The following rules are supported for feature group connection declarations that represent a connection up or down	0	0	0
Feature Group Connections	p95l4	(L4) The following rules are supported for feature group connection declarations that represent a connection between tw	0	0	0
Feature Group Connections	p95l5	(L5) If the feature group connection declaration represents a connection between feature group of sibling components, th	0	0	0
Feature Group Connections	p95l6	(L6) If the feature group connection declaration represents a connection between feature groups up the containment hier	0	0	0
Feature Group Connections	p95l7	(L7) If the feature group connection declaration represents a connection between feature groups down the containment h	0	0	0
Feature Group Connections	p95l8	(L8) A feature group connection must be bidirectional or be consistent with the direction of the source and destination fea	0	0	0