

# SAE AADL AS-5506 Errata Sheet

Sept 2006

This document lists a set of errata and corrections to the SAE Architecture Analysis & Design Language (AADL) standard published in Nov 2004 under the SAE publication number AS-5506.

Location	Errata
Section 4.4, Page 38, Syntax, Line starting with “[ <b>properties</b> ”	Syntax rule for component_implementation_extension: “property_association” replaced with (=>) “property_association   contained_property_association” Rationale: component implementations can have contained property associations (see p.37). Component implementation extensions should be permitted to have them as well. Was an editorial oversight.
Section 4.5, Page 43, Legality Rules	Missing legality rule: The classifier of a subcomponent cannot recursively contain subcomponents with the same component classifier. In other words, there cannot be a cyclic containment dependency between components.
Section 4.6, Page 47, Legality Rules, 3 <sup>rd</sup> paragraph	“Annex libraries can only be declared in packages.” Is too restrictive. The syntax rules on Page 29 permit Annex Libraries in AADL Specifications. Correction: “Annex libraries can be declared in packages and directly in AADL specifications.”
Section 5.2, Page 55, first sentence	Confusing wording as it suggests subprogram is related to entrypoints of threads. Correction: A subprogram component represents <del>an execution entrypoint in source text</del> <u>sequentially executed source text that is called with parameters.</u>
Section 5.3, Page 66, Figure 5	For the rounded box labels representing the action “performing” the word thread should be italicized. Correct: performing <i>thread initialization</i> performing <i>thread activation</i> performing <i>thread deactivation</i> performing <i>thread finalize</i>
Section 5.3, Page 69, Figure 6	Clarification/correction of label on arrows: “block on getResource” replaced with “block Get_Resource”. “unblock on releaseResource” replaced with “unblock Get_Resource”. Explanation for both corrections: Use of resource operation name in text. Explanation for second correction: The execution of a

	Release_Resource by another thread causes the unblocking of the Get_Resource operation of the given thread.
Section 5.3, Page 72, Runtime Support, last paragraph	Add a sentence to Processing Requirements and Permissions regarding Raise_Error. “The Raise_Error method is permitted to have an error identification as parameter value. This error identification can be passed through the error port as the data value, since the error port is defined as event data port.”
Section 5.5, Page 77, Standard Properties, first property	“Scheduling_Protocol: list of Supported_Scheduling_Protocols” replaced with “Runtime_Protection: inherit aadlboolean => true” Rationale: Scheduling_Protocol is a processor property. The comment in the line above the property indicates that the intended property is Runtime_Protection.
Section 6.1, Page 83, Standard Properties	Missing property for processor: Add “Scheduling_Protocol: list of Supported_Scheduling_Protocols”
Section 6.1, Page 83, Standard Properties P.213	Missing property for processor. It specifies whether a processor supports runtime address space protection: Add “Runtime_Protection: aadlboolean => true” Also update the property definition Runtime_Protection : <b>inherit aadlboolean =&gt; true</b>  <b>applies to (process, system, processor);</b>
Section 7.1, Page 93, first property; also Page 213	Property ““Scheduling_Protocol” is defined as “applies to (processor);”, thus, not applicable to “system”. Correction for page 213, Scheduling_Protocol: “Scheduling_Protocol list of Supported_Scheduling_Protocols applies to (processor)” replaced with “Scheduling_Protocol <i>inherit</i> list of Supported_Scheduling_Protocols applies to (processor, <i>system</i> )”
Section 8.1, Page 98, Semantics, paragraph 3	Change Compute_Time to Compute_Execution_Time.
Section 8.2, Page 105, Examples	“ <b>port group</b> GPSextended_plug <b>extends</b> GPSbasic_plug” replaced by “ <b>port group</b> GPSextended_plug <b>extends</b> GPSbasic_socket”  A port group extension cannot extend a port group type with an <b>inverse of</b> .
Section 8.3, Page 110-111, Examples	<pre>subprogram print   features     filetoprint: <u>in parameter</u> data file; end print; and thread A <u>annex pierre</u> {**   features     print: <b>requires</b> subprogram Pierre::print; **}; <b>calls</b></pre>

	<del>print: subprogram;</del> end A;
Section 8.5, Page 113, paragraph 1	Provided <u>and required</u> subcomponent access <u>are a feature of a component. Subcomponent access is limited to data components and bus components.</u>
Section 8.5, Page 113, paragraph 3	First sentence: “ ... a subcomponent provides access to a <del>data</del> component contained in the component.”
Section 8.5, Page 115, Legality Rules	There should be a legality rule stating that if the source code of a thread does access shared data, then the thread type declaration must specify a requires data access declaration. In other words, for all components that access shared data their component type declaration must reflect that fact.
Section 9.2.2, Page 140, Legality Rules	Add a legality rule: “Flow implementations may be declared from an in port directly to an out port.” This permits flow implementations to be declared for components without subcomponents and different property values to be attached for different component implementations.
Section 9.2.2, Page 141/142, Example	The comments following the declaration of b: thread baz.basic; should have an additional comment line -- baz has a flow path fs3 from port p1 to p3 This flow path is then referenced in the flow implementation declaration Flow2: <b>flow path</b> signal -> conn1 -> A.fs1 -> conn4 -> C. <del>fs1</del> fs3 -> conn5 -> result2;
Section 9.2.2, Page 142, Example	Flow4: <b>flow source</b> A.fs2 -> <del>connect</del> 6 -> status;
Section 9.2.3, Page 144, Example	The flow path Flow1 requires the following correction: Flow1: <b>flow path</b> signal -> conn1 -> A.fs1 -> conn4 -> C.fs1 -> conn3 -> <del>result2</del> result1;
Section 9.2.3, Page 145, Example	The end-to-end flow ete1 requires the following correction: ETE1: <b>end to end flow</b> A.fs2 -> <del>conn4</del> conn6 -> C.fsink; The connection conn6 must be added to the set of connection declarations on page 143. Conn6: event port A.p3 -> C.reset;
Section 10.1.3, Page 153, Syntax	Add brackets around aadlinteger and aadlreal to make clear that units applies to both. This occurs in both the single_valued_property_constant and multi_valued_property_constant rule. The same correction applies to the syntax summary on page 250. ( ( ( <b>aadlinteger</b>   <b>aadlreal</b> )

	[ <i>units_unique_property_type_identifier</i> ] )   <b>aadlstring</b>   <b>aadlboolean</b>
Section 10.3, Page 156, Syntax/Legality Rules	<p>“Property associations for component types, features, and flow specifications must not have an <b>in modes</b> statement.” The reason is that the modes of a component are only visible in its implementation, not its type.</p> <p>Correction: Add the above quoted sentence as legality rule.</p>
Section 10.3, Page 157, Naming Rules, Paragraph 5	<p>The naming rules do not allow a subprogram call to be named as the AADL model element to which a contained property association <b>applies to</b>. In particular, this restriction does not permit users to specify an <code>Actual_Subprogram_Call</code> property value to bind a subprogram call declaration to a server subprogram in a thread to indicate a remote procedure call. The reason is that the property association has to be declared as a contained property association in a component implementation, one of whose subcomponents contains the thread with the server subprogram to be referenced, thus, the <b>applies to</b> statement of the contained property association has to be able to name the call to be characterized as remote.</p> <p>Addition marked with underscore: The dot-separated identifier sequence following the reserved words <b>applies to</b> of a contained property association identifies a component, feature, flow, connection, <u>call</u>, or mode in the containment hierarchy, for which the property value holds.</p>
Section 10.3, Page 158, Legality Rules	<p>Missing legality rule. It is possible to declare a modal property association as part of a modal subcomponent, connection, call sequence, or flow implementation declaration. For example,</p> <pre>myThread: <b>thread</b> example { Period =&gt; 10 ms <b>in modes</b> (m1, m2); } <b>in modes</b> ( m1 );</pre> <p>As this example shows we need a legality rule that limits the set of modes in the <b>in modes</b> statement of the property association to a subset of the modes in the <b>in modes</b> statement of the subcomponent.</p> <p>Additional Legality Rule:  <u>“If a property association with an <b>in modes</b> statement is declared as part of a subcomponent, connection, flow implementation, or call sequence with an <b>in modes</b> statement, then the set of modes for which the property association applies must be contained in the set of modes for which the subcomponent, connection, flow implementation, or call sequence is active.”</u></p>
Section 10.4, Page 163, Syntax	<p>Add parentheses to clarify that <b>reference</b> applies to all three alternatives.</p> <pre>reference_term ::=     <b>reference</b> (         subcomponent_identifier { .         subcomponent_identifier }*           { subcomponent_identifier. }+</pre>

	<pre> connection_identifier       { subcomponent_identifier . }+ server_subprogram_identifier ) </pre>
Section 10.4, Page 166, Semantics, Reference property type bullet 1	<p>The bullet states that a property with a reference value that is declared in a component implementation must start with an identifier to a subcomponent. This restriction does not permit a reference to a connection, flow implementation, or call in the same component implementation to be specified. On the other hand, bullet 2 allows the first identifier to be a connection, flow implementation, or call in the component implementation of the subcomponent for which the property association is declared.</p> <p>Correction of bullet 1:</p> <p>For property associations of component implementations, the first identifier in the reference must appear <del>as a subcomponent identifier</del> in the local namespace of the component implementation to which the property association belongs.</p>
Section 10.4, Page 168, Example	<pre> Allowed_Processor_Binding =&gt; <b>reference</b> HW.Host_B     <b>applies to</b> <del>SW.Sampler_A</del> <i>SW.Sampler_B</i>; </pre> <p>Deletion marked with strikethrough and addition marked in italic.</p>
Section 11.1, Page 171, Semantics	<p>Paragraph 2, sentence 3 is a sentence fragment.</p> <p>Delete.</p>
Section 11.1, Page 172, Mode Switch	<p>Paragraph 1, last sentence</p> <p>Mode transitions cannot have properties. The urgency property is associated with ports.</p> <p>Correction:</p> <p>If an Urgency property is associated with each <u>port named in mode transitions</u>, then the mode transition with the highest <u>port urgency</u> takes precedence. <u>If several ports have the same urgency then the mode transition is chosen non-deterministically.</u></p>
Section A.1, Page 199	<pre> Actual_Subprogram_Call_Binding: <b>inherit list of</b> <b>reference (bus, processor, memory, device)</b> </pre> <p>This is useful if subprogram calls can be routed through devices.</p>
Section A.1, Page 200	<pre> Allowed_Connection_Protocol: <b>list of enumeration</b> Data_Connection, Event_Connection, Event_Data_Connection, Data_Access_Connection, Server_Subprogram_Call) <b>applies to (bus, device)</b>; </pre> <p>No reason for limit to data and event connection.</p>
Section A.1, Page 205	<p>In Compute_Execution_Time description:</p> <p>If the property is specified for a subprogram, event port, or event data port feature, then this compute execution time applies to the dispatched thread when the corresponding call, event, or event data</p>

	<u>initiates a dispatch.</u>
Section A.1, Page 206	Deadline: <u>inherit</u> Time => <del>inherit</del> value(Period) This property definition is also referenced on pages 61, 75, 77, 90, 93.
Section A.1, Page 207	Device_Dispatch_Protocol: Supported_Dispatch_Protocols => Aperiodic  <b>applies to (<del>thread,</del> device);</b>  The device thread is not explicitly declared. Thus, this property should only be associated with a device.
	<b>New since July 2005</b>
Section 10.1.1, Page 148, Naming Rules paragraph 3	Clarify the fact that the list of enumeration literals is ordered. “An enumeration type introduces an enumeration namespace. The enumeration literal identifiers in the enumeration list declare <del>a set</del> <u>an ordered list</u> of enumeration literals.”
Various Sections	Classifier refinement. Different sections of the standard that deal with <b>refined to</b> declarations use different words to state that classifiers can only be completed. Updated these sections to use the same wording.
Section 10.3, Page 159, Paragraph 1	Property values for features should be retrieved from their classifier for those that have classifiers.  “For features in a component type or port group type, or flow specifications in a component type, the property value of a property is determined by its property association in the feature or flow specification declaration. If not present and the feature or flow specification is refined, then the property value is determined by a property association in the feature or flow specification declaration being refined; this is done recursively along the refinement sequence. For <del>subprogram, server subprogram, and port group</del> features, if not present and the feature references a <del>subprogram</del> -classifier or port group type reference, the property value is determined by the <del>subprogram-component</del> classifier reference or port group type according to the respective component implementation, component type, or port group type rules described above. <u>If a classifier reference is not present for a feature and the feature is refined, then the classifier or port group type reference of the feature being refined is considered in determining the property value.</u> ”
Section 10.4, Page 161 & Section 10.1.3, Page 152	Support units literals as property values in property expressions.  property_expression ::= boolean_term   real_term

```

| integer_term
| string_term
| enumeration_term
| unit_term
| real_range_term
| integer_range_term
| property_term
| component_classifier_term
| reference_term

unit_term ::=
unit_identifier | unit_property_constant_term

Support for defining property constants of type units.
single_valued_property_constant ::=
    defining_property_constant_identifier : constant
        ( ( ( aadlinteger | aadlreal )
            [
units_unique_property_type_identifier ] )
        | aadlstring | aadlboolean
        | enumeration_unique_property_type_identifier
        | unit_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
        ( ( ( aadlinteger | aadlreal )
            [
units_unique_property_type_identifier ] )
        | aadlstring | aadlboolean
        | enumeration_unique_property_type_identifier
        | unit_unique_property_type_identifier
        | integer_range_unique_property_type_identifier
        | real_range_unique_property_type_identifier

```

	<pre>   integer_unique_property_type_identifer   real_unique_property_type_identifer )     =&gt; ( [ constant_property_value { , constant_property_value }* ] ) ;  constant_property_value ::=     string_literal       signed_integer       signed_real       boolean_value       enumeration_identifier       <u>unit_identifier</u>       signed_aadlinteger .. signed_aadlinteger [ <b>delta</b> signed_aadlinteger ]       signed_aadlreal .. signed_aadlreal [ <b>delta</b> signed_aadlreal ] </pre>
Section 1.1.1 Page 146	<p>Remove the ability to “rename” a property type, i.e., define a second name for a property type.</p> <pre> property_type_declaration ::=     <i>defining_property_type_identifier</i> : <b>type</b> <del>property_type_designator</del> <u>property_type</u> ;  <del>property_type_designator ::=</del> <del>property_type   unique_property_type_identifier</del> </pre>
Section A.1 Page 197	<p>Wording for Allowed_Connection_Binding and Allowed_Connection_Binding_Class</p> <p>“The Allowed_Connection_Binding property specifies the execution platform resources that are to be used to perform a communication. The property type is a list of component names. <del>The list must contain an odd number of component names.</del> The named components must belong to a processor, device or bus category.</p> <p>The first component named in the list must be either the processor to which the thread containing the ultimate source feature is bound, or else the processor or device containing the ultimate hardware source feature. The last component named in the list must be either the processor to which the thread containing the ultimate destination feature is bound, or else the processor or device containing the ultimate hardware destination feature. The intermediate component names must <del>alternate between</del> <u>be</u> a bus, and a processor or a device. Each pair of</p>

	names for processor or device components that are separated by the name of a <u>one or more</u> bus components <del>must share that be connected through</del> bus component. That is, the sequence of processors, devices and buses must form a connected path through the specified hardware architecture.
Section A.1 Page 196	Actual_Connection_Binding: <b>inherit <u>list of reference</u> (bus, processor, device)</b>
Section A.1 Page 211	Permit Source_Code_Size and Source_Data_Size to be used on system and thread group.  <b>applies to (data, subprogram, thread, <u>thread group</u>, process, <u>system</u>, processor, device);</b>
Section A.1 Page 214	Source_Language: the text describes it as an <b>inherit</b> property. Also it should be applicable to thread group and system.
Section 9.2.1, Page 137, Legality Rules	Last legality rule should refer to flow sink instead of flow source: The direction declared for the source port or parameter of a flow <del>source</del> <u>sink</u> specification declaration must be <b>in</b> or <b>in out</b> .
Section A.1 Page 203	Base_Address should apply to memory, not just data.
	<b>New Since April 2006 (implemented in OSATE 1.3.0)</b>
Section A.1 Page 198	Actual_Memory_Binding should apply to port group such that it can be inherited by the contained data ports and event data ports..
Section A.1 Page 217	Urgency should apply server subprogram feature aka subprogram feature or provides subprogram access.
	<b>New Since July 1, 2006</b>
Section 10.1.1 Page 147	A property type can be declared in terms of another user-named property type. This offers renaming of property types instead of defining the second type in terms of base types. Remove this capability (was done before July 1 but not recorded here). property_type_declaration ::=  <i>defining_property_type_identifier</i> : <b>type</b> <del>property_type_designator</del> <u>property_type</u> ;
Section 8.2 Page 102/104	It says that if a port group extends another port group it cannot have a inverse of statement. Yet, in the syntax on 102 there are inverse of statements in the grammar of a port_group_type_extension. The syntax should only allow it for inverse of with features. The legality rule should state that the inverse of restriction only applies to port group types with inverseof and no features.
Section 9.2.1 Page 137	The legality rules for flow specs only talk about port direction constraints. IT should not be possible to specify a flow from a port group that does not have any outgoing features (and same for the flow target).

	<p>Add text to Legality rules:  <i>If the flow specification refers to a port group then the port group must contain at least one port or parameter that satisfies the above rules.</i></p>
Section A.1	<p>Entrypoint properties for activate, deactivate, recover, initialize, finalize have the following sentence in their description.  <i>This property may have an unspecified value.</i>  The document does not define “unspecified”. The intent of this sentence is that the user does not have to provide a value for this property. This is already stated in the text of Threads. In general incomplete models can be defined without property values for any property.  Action: delete this sentence.</p>
Section Processor & Memory	<p>Processor and memory can contain memory with requires bus access. Therefore, we must support bus access connections in bus or processor.</p>
Section A.1 Page 203	<p>Base_Address should apply to ports.</p>
Section 9.2.2	<p>Add legality rule to ensure that the port names in the flow spec are matched by the port names in the flow implementation.</p>
Section A.1	<p>Allowed and Actual Connection Binding.  Is currently only applicable to port connections. Data may reside in memory that is connected by bus with a processor. Therefore, it is useful to also specify bindings for data access connections.  Allow data access connections to be listed as applies to (p.152).</p>
Section A.1	<p>Actual_Processor_Binding is applicable to device as well.  Allowed binding already includes device.</p>