#### 27.1.2.0 General

An attribute definition that is directly attached to a lower scope unit will override a general attribute definition in a higher scope and a type-specific attribute inherited from a type reference. Attributes inherited from a type reference will override general attributes from a higher scope unit containing the type reference. Additional overwriting rules for variant attributes are defined in clause 27.1.2.1.

EXAMPLE 1:

 **type** **record** MyRecordA

 {

 :

 } **with** { **encode** "RuleA" }

 // In the following, MyRecordA is encoded according to "RuleA" and not according to

 // "RuleB" because the attribute from the referenced type MyRecordA overrides

 // the attribute from higher scope unit (surrounding MyRecordB type).

 **type record** MyRecordB

 {

 :

 MyRecordA field

 } **with** { **encode** "RuleB" }

A **with** statement that is placed inside the scope of another **with** statement shall override the outermost **with**. This shall also apply to the use of the **with** statement with groups. If multiple attributes of the same type are allowed, all of them are overridden unless specified otherwise.

EXAMPLE 2:

 // Example of the use ofthe overwriting scheme of the **with** statement

 **group** myPDUs

 {

 **type** **record** MyPDU1 { … }

 **type** **record** MyPDU2 { … }

 **group** mySpecialPDUs

 {

 **type** **record** MyPDU3 { … }

 **type** **record** MyPDU4 { … }

 }

 **with** {**extension** "MySpecialRule"} // MyPDU3 and MyPDU4 will have the application

 // specific extension attribute MySpecialRule

 }

 **with**

{

 **display** "PDU"; // All types of group myPDUs will be displayed as PDU and

 **extension** "MyRule"; // (if not overwritten) have the extension attribute MyRule

 }

 // is identical to …

 **group** myPDUs

 {

 **type** **record** MyPDU1 { … } **with** {**display** "PDU"; **extension** "MyRule" }

 **type** **record** MyPDU2 { … } **with** {**display** "PDU"; **extension** "MyRule" }

  **group** mySpecialPDUs {

 **type** **record** MyPDU3 { … } **with** {**display** "PDU"; **extension** "MySpecialRule" }

 **type** **record** MyPDU4 { … } **with** {**display** "PDU"; **extension** "MySpecialRule" }

 }

 }

Attributes defined for a synonym type don't override existing attributes of fields or elements of this synonym type. The attributes are applied to the fields or elements of synonym types only if the fields or elements have no valid attributes.

EXAMPLE 3:

 // Example of the use ofattributes in synonym types

 **type** **record** SourceType1 {

 **integer** field1,

 **integer** field2

 } // neither the record nor its fields have a valid attribute

 **type** **record** SourceType2 {

 **integer** field1,

 **integer** field2

 } **with** { **encode** "Rule1" }

 // the record and its fields have a valid encode attribute "Rule1"

 **type** SourceType1 SynonymType1 **with** { **encode** "Rule2" }

 // SynonymType1 and all its fields will be encoded with Rule2

 **type** SourceType2 SynonymType2 **with** { **encode** "Rule3" }

 // SynonymType2 will be encoded with Rule3, but field1 and field2 will be encoded with

 // Rule1 as SourceType2 definition already specifies the encode attribute of these fields

Attributes with the **@local** modifier **o**verride attributes from higher scope, but they are valid for the associated language element only. They do not affect definitions inside the associated language element as the **@local** attribute is completely transparent to lower scopes. Attributes from higher scope will still affect attributes in lower scopes even if the **@local** attribute is between them.

NOTE: Attributes with the **@local** modifier associated to modules and groups are valid, but do not affect the definitions inside them.

EXAMPLE 4:

 module M {

 **type** **record** MyRec {

 **integer** field1,

 **integer** field1,

 } **with** { **encode @local** "CodecB" }

 // the record type MyRec will be encoded with CodecB, but its fields with CodecA,

 // because the local attribute CodecB doesn't affect fields of the MyRec type.

 } **with** { **encode** "CodecA" }

An attribute definition in a lower scope or those inherited from a referenced type can be overwritten in a higher scope by using the **override** directive.

EXAMPLE 5:

 **type** **record** MyRecordA

 {

 :

 } **with** { **encode** "RuleA" }

 // In the following, fieldA of a MyRecordB instance is encoded according to RuleB

 **type record** MyRecordB

 {

 :

 MyRecordA fieldA

 } **with** { **encode** **override** (fieldA) "RuleB" }

The **override** directive overrides the specified attribute for all declarations at all lower scopes that do not also declare the specified attribute. If the override directive is applied to a type reference, it doesn't affect the attributes of the original referenced type.

An attribute definition directly attached to a field or element of a structured type overrides the corresponding attribute of the structured type, as regards the identified field or element. Override attribute applied to a synonym type (clause 6.4) overrides attributes of all fields or elements of the synonym type unless the synonym type definition contains an explicit attribute definition for the field or element.

EXAMPLE 6:

 // An instance of MyRecordA is encoded according to RuleA.

 **type** **record** MyRecordA

 {

 :

 } **with** { **encode** **override** "RuleA" }

 // In the following, fieldA of a MyRecordB instance (and all its sub-fields) is encoded

 // according to "RuleB".

 **type** **record** MyRecordB

 {

 :

 MyRecordA fieldA

 } **with** { **encode** **override** "RuleB" }

 // The following template will use "RuleA" as the override directive for MyRecordB affects only

 // MyRecordB.fieldA, but not the original MyRecordA.

 **template** MyRecordA mw\_msg;

 // In the following, rule "RuleB" is overridden by "RuleC" for fieldC, but it is

 // not overridden by "RuleA" of the group because the direct attachment to fieldC and

 // MyRecordC override the encode of the outer scope.

 **group** myGroup {

 **type** **record** MyRecordC

 {

 :

 } **with** { **encode** **override** "RuleB" }

 **type** **record** MyRecordD

 {

 :

 MyRecordC fieldC

 } **with** { **encode** **override** (fieldC) "RuleC" }

 } **with** { **encode** **override** "RuleA" }

 // In the following, the template mw\_msg will be encoded with "RuleB", because the

 // override directive doesn't override the encode attribute in references. However,

 // all fields of the mw\_msg template will be encoded with "RuleA", because the attributes

 // from the references have higher precedence than attributes from a higher scope.

 **type** **record** MyRecordE

 {

 :

 } **with** { **encode** **override** "RuleA" }

 **template** MyRecordE mw\_msg **:=**

 {

 :

 } **with** { **encode** "RuleB" }

 // MyRecordG and its "field1" member will be encoded with "RuleB", but its field2 member

 // will be encoded with "RuleA", because there's an encode attribute explicitly declared

 // for this field.

 **type** **record** MyRecordF {

 **integer** field1,

 **integer** field2

 } **with** { **encode** "RuleA" }

 **type** MyRecordF MyRecordG **with** {

 **encode** **override** "RuleB";

 **encode**(field2) "RuleA"

 }