

22. Encapsulation constructs 1

This clause contains the syntax and descriptions of the *e* statements used to create packages and modify access control. 5

22.1 package package-name

Purpose	Associates a module with a package 10
Category	Statement
Syntax	package <i>package-name</i> 15
Parameters	<i>package-name</i> A standard <i>e</i> identifier which assigns a unique name to the package. It is legal for a package name to be the same as a module or type name.

[This associates a module with a package.](#) Only one **package** statement can appear in a file and it shall be the first statement in the file. A file with no **package** statement is equivalent to a file beginning with the **package main** statement. 20

Syntax example:

```
package vr_xb; 25
```

22.2 package type-declaration

Purpose	Modifies access to a type or a struct 30
Category	Statement
Syntax	[package] <i>type-declaration</i> 35
Parameters	<i>type-declaration</i> An <i>e</i> type declaration (for a struct, unit, enumerated list, or other type).

The **package** modifier can be used to shield the defined struct member from code outside the package files. This includes declaring a variable of the type, extending, inheriting, casting using the **as_a** operator, and all other contexts in which the name of a type is used. It is equivalent to the default (package) access level for classes in Java. Without the **package** modifier, the type or struct has no access restriction. 40

NOTE—The package type does not determine the visibility of a package, but only its access control. 45

A derived struct (using like inheritance) needs to be explicitly declared as **package** if its base struct is declared **package**. Otherwise, it can be declared **package** (even if its base struct is not).

The definition of a when subtype (using a **when** or **extend** clause) does not allow for an access modifier. A **when** subtype is public unless its base struct or one of its determinant fields is declared **package**. A **when** subtype cannot have a **private** or **protected** determinant field. 50

Any reference to a **when** subtype, even in a context in which the **when** determinant field is accessible, shall result in a compilation error. 55

Syntax example:

```
package type t: int(bits: 16);
```

22.3 package | protected | private struct-member

Purpose	Modifies access to a struct field, method, or event
Category	Keyword
Syntax	package <i>struct-member-definition</i> protected <i>struct-member-definition</i> private <i>struct-member-definition</i>
Parameters	<i>struct-member-definition</i> A struct or unit field, method, or event definition. See Clause 4 for the syntax of struct and unit member definitions.

A struct member declaration can include a **package**, **protected**, or **private** keyword to modify access to the struct member. If no access modifier exists in the declaration of a struct member, the struct member has no access restriction (the default is public).

- The **package** modifier means code outside the package files cannot access the struct member. It is equivalent to the default (package) access level for fields and methods in Java.
- The **protected** modifier means code outside the struct family scope cannot access the struct member. It is similar (although not equivalent) to the *protected* semantics in other object-oriented languages. The *struct family scope* [is the code](#) within the definition of a struct (declaration and extensions), as well as the definition of all **when** and **like** subtypes.
- The **private** modifier means only code within both the package and the struct family scope can access the struct member. This means code within the extension of the same struct in a different package is outside its accessibility scope. It is less restrictive than the *private* attribute of other object-oriented languages in the sense that methods of derived structs or units within the same package can access a private struct member.

Only fields, methods, and events can have access restrictions. There are other named struct members in *e*, namely cover groups and named expects, to which access control does not apply — they are completely public. However, cover groups and expects are defined in terms of fields, methods and events, and can refer to other entities in their definitions according to the accessibility rules.

- a) An extension of a struct member can restate the same access modifier as the declaration has or omit the modifier altogether. If a different modifier appears, the compiler shall issue an error.
- b) All references to a struct member outside its accessibility scope shall result in an error at compile time. Using an enumerated field's value as a **when** determinant is considered such a reference, even if the field name is not explicitly mentioned.
- c) A field shall be declared **package** or **private** if its type is **package**, unless it is a member of struct which is declared **package**. A method shall be declared **package** or **private** if its return type or any of its parameter types are **package**, unless it is a method of a struct which is declared **package**.

Syntax examples:

```
private f: int;
protected m() is {};
package event e;
```

202