**SLang reference manual.**

Version 0.21

1. **SLang syntax:**

<Compilation> ::= [<CompilationUnit>](#CompilationUnit) {[<CompialtionUnit>](#CompilationUnit)}

<CompialtionUnit> ::= {[<UseDirective>](#UseDirective)} [<AnonymousRoutine>](#AnonymousRoutine)|[<StandaloneRoutine>](#StandaloneRoutine)|[<UnitDeclaration>](#UnitDeclaration)

<UseDirective> ::= “**use**” [<UseElement>](#UseElement) {“**,**” [<UseElement>](#UseElement)}

<UseElement> ::= [<FullUnitName>](#FullUnitName) [“**as**” [<Identifier>](#Identifier)]

<FullUnitName> ::= [<Identifier>](#Identifier) [“**[**” [<UnitType>](#UnitType) {“**,**” [<UnitType>](#UnitType)}“**]**”]

<AnonymousRoutine> ::= [<StatementsList>](#StatementsList)

<StatementsList> ::= [<Statement>](#Statement) { [“**;**”] [<Statement>](#Statement)}

<StandaloneRoutine> ::= [<Identifier>](#Identifier) [[<FormalGenerics>](#FormalGenerics)] [[<Arguments>](#Arguments)] [“**:**” [<Type>](#Type)] “**is**”

[[<RequireBlock>](#RequireBlock)]

[<InternalRoutineBody>](#InternalRoutineBody)|[<ExternalRoutineBody>](#ExternalRoutineBody)

[[<EnsureBlock>](#EnsureBlock)] [[<RescueBlock>](#RescueBlock)] “**end**” [[<Identifier>](#Identifier)]

<Arguments> ::= “**(**” [<Argument>](#Arguemnt) {”**;**” [<Argument>](#Arguemnt)}“**)**”

<Argument> ::= [<Identifier>](#Identifier) {“**,**”[<Identifier>](#Identifier)} “**:**”[<Type>](#Type)

<RescueBlock> ::= “**rescue**” [<StatementsList>](#StatementsList)

<RequireBlock> ::= “**require**” [“**else**”] {[<Predicate>](#Predicate)}

<EnsureBlock> ::= “**ensure**” [“**then**”] {[<Predicate>](#Predicate)}

<Predicate> ::= [[<StringConstant>](#StringConstant) “**:**”][<BooleanExpression>](#BooleanExpression)

<InternalRoutineBody> ::= [<StatementsList>](#StatementsList)

<ExternalRoutineBody> ::= “**external**” <StringConstant>?….

<UnitDeclaration> ::= [“**ref**”|”**val**”|”**concurrent**”] “**unit**”|”***module***”|”***class***” [<Identifier>](#Identifier) [[<FormalGenerics>](#FormalGenerics)]

[[<InheritDirective>](#InheritDirective)] [[<UseDirective>](#UseDirective)] {[<FeatureSelection>](#FeatureSelection)}

{[<FeatureDeclaration>](#FeatureDeclaration)| (“**public**”|”**private**” “**:**” {[<FeatureDeclaration>](#FeatureDeclaration)})}

[“**invariant**” {[<Predicate>](#Predicate)}] “**end**” [[<Identifier>](#Identifier)]

<InheritDirective> ::= “**inherit**”| “***:***” [<Parent>](#Parent) {“,” [<Parent>](#Parent)}

<Parent> ::= [<UnitType>](#UnitType) [“**’**”]

<GenericInstantiation> ::= “**[**”[<FactualGeneric>](#FactualGeneric) {“**,**” [<FactualGeneric>](#FactualGeneric)}“**]**”

<FactualGeneric> ::= [<Type>](#Type)|<IntegerConstant>|<CharacterConstant>|<BooleanConstant>|<EnumConstant>

<FormalGenerics> ::= “**[**” [<FormalGeneric>](#FormalGeneric) {“**,**” [<FormalGeneric>](#FormalGeneric)}“**]**”

<FormalGeneric> ::= [<Identifier>](#Identifier) [“**->**” [<TypeConstraint>](#TypeConstraint)|(“**val**” [<UnitType>](#UnitType))]

<TypeConstraint> ::= [<UnitType>](#UnitType) [“**init**”|”***ctor***” [[<Identifier>](#Identifier)] [“**(**”[<UnitType>](#UnitType) {“**,**”[<UnitType>](#UnitType)}”**)**”] ]

<FeatureSelection> ::= “**override**” [<Parent>](#Parent) ”**.**”|”**->**” [<Identifier>](#Identifier) {[“**,**”] [<Parent>](#Parent) ”**.**”|”**->**” [<Identifier>](#Identifier)}

<FeatureDeclaration> ::= [“**override**”] [“**public**”|”**private**”] [“**final**”]

[<UnitAttribiteDeclaration>](#UnitAttributeDeclaration)|[<UnitRoutineDeclaration>](#UnitRoutineDeclaration)|[<InitDeclaration>](#InitDeclaration)

<InitDeclaration> ::= “**init**”|”***ctor***” [[<Identifier>](#Identifier)] [[<Arguments>](#Arguments)] “**is**” [[<RequireBlock>](#RequireBlock)]

[<InternalRoutineBody>](#InternalRoutineBody)

[[<EnsureBlock>](#EnsureBlock)] [[<RescueBlock>](#RescueBlock)] “**end**” [“**init**”|”***ctor***”|[<Identifier>](#Identifier)]

<UnitRoutineDeclaration> ::= [“**pure**”|”**safe**”] [<Identifier>](#Identifier) [[<Arguments>](#Arguments)] [“**:**”[<Type>](#Type)] “**is**” [[<RequireBlock>](#RequireBlock)]

“**abstract**”| [<InternalRoutineBody>](#InternalRoutineBody)|[<ExternalRoutineBody>](#ExternalRoutineBody)

[[<EnsureBlock>](#EnsureBlock)] [[<RescueBlock>](#RescueBlock)] “**end**” [[<Identifier>](#Identifier)]

// Abstract routine must not have <RescueBlock>

<Statement> ::=

[<Assignment>](#Assignment)|[<LocalAttributeDeclaration>](#LocalAttributeDeclaration)|([[<Label>](#Label)”**:**”][<If>](#If))|([[<Label>](#Label)”**:**”][<Case>](#Case))|([[<Label>](#Label)”**:**”][<Loop>](#Loop))|”**break**” [[<Label>](#Label)]|[<FeatureCall>](#FeatureCall)|”**retry**”|”**?**” [<Identifier>](#Identifier)| [<Assert>](#Assert) |…..

<Label> ::= [<Identifier>](#Identifier)

<Assert> ::= “**assert**” {[<Predicate>](#Predicate)} “**end**” [“**assert**”]

<Assignment> ::= [<Writable>](#Writable) “**:=**”[<Expression>](#Expression)

<LocalAttributeDeclaration> ::= [<TypedLocalAttributeDeclaration>](#TypedLocalAttributeDeclaration)|([“**var**”] [<Identifier>](#Identifier) [<AttributeInitializer>](#AttributeInitializer))

<UnitAttributeDeclaration> ::= [<TypedUnitAttributeDeclaration>](#TypedUnitAttributeDeclaration)|( [“**const**”] [<Identifier>](#Identifier) [<AttributeInitializer>](#AttributeInitializer) [“**invariant**” {[<Predicate>](#Predicate)}])

<AttributeInitializer> ::= [“**:**”[<Type>](#Type)] “**is**” [<Expression>](#Expression)

<TypedLocalAttributeDeclaration> ::= [“**var**”] [<Identifier>](#Identifier) {“**,**” [“**var**”] [<Identifier>](#Identifier)} “**:**”[<Type>](#Type)

<TypedUnitAttributeDeclaration> ::= [“**const**”] [<Identifier>](#Identifier) {“**,**” [“**const**”] [<Identifier>](#Identifier)} “**:**”[<Type>](#Type)

<Writable> ::= “**result**”|( [<Identifier>](#Identifier) [“**[**” [<ExprList>](#ExprList) “**]**”] {“**.**” [<Identifier>](#Identifier) “**[**”[<ExprList>](#ExprList) “**]**”})

<ExprList> ::= [<Expression>](#Expression) {“**,**” [<Expression>](#Expression)}

<BooleanExpression> ::= [<Expression>](#Expression)

<Expression> ::= [<IfExpression>](#IfExpession)|[<CaseExpression>](#CaseExpression)|….

<IfExpession> ::= “**if**” [<BooleanExpression>](#BooleanExpression) “**then**” [<Statement>](#Statement)

{“**elsif**”|”**elseif**” [<BooleanExpression>](#BooleanExpression) “**then**” [<Statement>](#Statement) }

[“**else**” [<Statement>](#Statement)] “**end**”

<CaseExpression> ::= “**case**” [“**typeof**”] [<Expression>](#Expression)

“**when**” [<WhenExpression>](#WhenExpression) {“**,**” [<WhenExpression>](#WhenExpression)} “**then**” [<Statement>](#Statement)

{“**when**” [<WhenExpression>](#WhenExpression) {“**,**” [<WhenExpression>](#WhenExpression)} “**then**” [<Statement>](#Statement)}

[“**else**” [<Statement>](#Statement)]

“**end**” [“**case**”]

<FeatureCall> ::= [<Identifier>](#Identifier)|”**this**”|”**super**” {[<CallChain>](#CallChain)}

<CallChain> ::= (“**.**”[<Identifier>](#Identifier) {[<CallChain>](#CallChain)})|([[<GenericInstantiation>](#GenericInstantiation)] “**(**”[<Expression>](#Expression){“**,**” [<Expression>](#Expression)}”**)**”)

<If> ::= “**if**” [<BooleanExpression>](#BooleanExpression) “**then**” [<StatementsList>](#StatementsList)

{“**elsif**”|”**elseif**” [<BooleanExpression>](#BooleanExpression) “**then**” [<StatementsList>](#StatementsList)}

[“**else**” [<StatementsList>](#StatementsList)] “**end**” [“**if**”]

<Case> ::= “**case**” [“**typeof**”] [<Expression>](#Expression)

“**when**” [<WhenExpression>](#WhenExpression) {“**,**” [<WhenExpression>](#WhenExpression)} “**then**” [<StatementsList>](#StatementsList)

{“**when**” [<WhenExpression>](#WhenExpression) {“**,**” [<WhenExpression>](#WhenExpression)} “**then**” [<StatementsList>](#StatementsList)}

[“**else**” [<StatementsList>](#StatementsList)]

“**end**” [“**case**”]

<WhenExpression> ::= [<Expression>](#Expression) [“**..**” [<Expression>](#Expression)]

<Loop> ::= [“**while**” [<BooleanExpression>](#BooleanExpression)] “**loop**” [“**invariant**” {[<Predicate>](#Predicate)}]

[<StatementsList>](#StatementsList)

[“**variant**” {[<Predicate>](#Predicate)}]

[“**while**” [<BooleanExpression>](#BooleanExpression)]

“**end**” [“**loop**”]

<Type> ::= [<UnitType>](#UnitType)|[<AnchorType>](#AnchorType)|[<MultiType>](#MultiType)|”**?**”[<Type>](#Type)| ….

<AnchorType> ::= “**like**” “**this**”| [<Identifier>](#Identifier)

<MultiType> ::= [<UnitType>](#UnitType) {“**|**” [<UnitType>](#UnitType)}

<UnitType> ::= [“**ref**”|”**val**”|”**concurrent**”] [<Identifier>](#Identifier) [[<GenericInstantiation>](#GenericInstantiation)]

<Identifier> ::= [<Letter>](#Letter){[<Letter>](#Letter)|[<Digit>](#Digit)}

<StringConstant> ::= “**”**” {[<Character>](#Character)} “**”**”

<CharacterConstant> ::= “**’**” [<Character>](#Character) “**’**”

<Character> ::= [<Letter>](#Letter)|[<Digit>](#Digit)|’+’…….

<Letter> ::= ‘A’|..’Z’|’a’|..’z’

<Digit> ::= ’0’|..’9’

1. **SLang semantics:**

SEM001: <Compilation> is valid is and only if all <CompilationUnit> are valid

SEM002: <CompilationUnit> is valid is and only if

….