Living Systems® Process Suite **Expression Language**

Types and value creation expressions

Simple	String	`"Hello" #"unlocalized string" $localized_string(argument_1, argument_2, …)`
	Boolean	`true false`
	Integer	`1 -100`
	Decimal Decimal (scale, rounding)	`6.63E-34`

	Date	`date(year, month, day)`
		`date(year, month, day, hours, minutes, seconds, millis)`
		`date(string)`
		`date(string, pattern)`

	Object	*any_expression*
	Null	`null`
Containers	Set<T>	`{ element_1, ... }`
	List<T>	`[ element_1, ... ]`
	Map<K,V>	`[ key_1 -> value_1, ... ]` `empty map: [ -> ]`
Other	Record	**new** `record_name(field_1 -> value_1,…)`
	Reference	`&variable`
	Closure	`no arguments: { -> expression }` `two arguments: { x, y -> expression }`

Special constructs

Local variable	**def** `type name`
Assignment	`variable := expression`
Expression chaining	`expression_1 ; expression_2 ; expression_3`
Error throw	**error**`(string_expression)`
Try-catch	**try** `expression` **catch** `exception_1,… -> expression_1` **… end**

Operators

Boolean		Arithmetic	
Equality	`= != <>`	Addition	`+`
Comparison	`< <= > >= <=>`	Subtraction	`-`
Negation	`not`	Multiplication	`*`
Conjunction	`and`	Division	`/`
Disjunction	`or`	Modulo	`%`
Exclusive disjunction	`xor`	Exponentiation	`**`
Pattern matching	`like`	String concatenation	`+`

Access			
Set element	`set[index]`	Enumeration literal	`enum.literal`
List element	`list[index]`	Dereferencing	`*reference`
Map element	`map[key]`	Namespace separator	`::`
Record field	`record.field`	Record safe field	`record?.field`
Record reference field	`reference.field`	Record ref. safe field	`reference?.field`
Function call	`function|T_1,…|(argument_1, argument_2,…)` `function|T_1,…|(name_1->value_1, name_2->value_2,…)`		
Closure invocation	`closure(argument_1, argument_2,…)`		

Control flow

If-then If-then-else If-then-elsif-then If-then-elsif-then-else	**if** `condition_1` **then** `expression_1` **elsif** `condition_2` **then** … **else** `expression_N` **end**
Ternary conditional	`condition ? expression_1 : expression_2`
Null-coalescing	`expression_1 ?? expression_2`
Switch	**switch** `arg_expression` **case** `c1,…` -> `expression_1` … **default** -> `defaultExpression` **end**

While looping	**while** `boolean_expression` **do**
	`expression`
	end

Collection iteration	**foreach** `type iterator_name` **in** `collection` **do** `expression` **en**

WHITESTEIN Technologies

Living Systems® Process Suite **Expression Language**