## 27.8 Getting attribute values

TTCN‑3 provides a set of operations that can be used for retrieving attribute values associated with a type, template, variable, constant or module parameter.

***Syntactical Structure***

( *Type* | *TemplateInstance* ) "." ( **display** | **encode** | **variant** | **extension** | **optional** )

[ "(" *Expression* ")" ]

***Semantic Description***

The operation returns the actual value of an attribute associate with the type, template, variable, constant or module parameter that precedes the dot symbol. The value preceding the dot symbol may be unitialized. The attribute type is denoted by the keyword following the dot symbol.

The return value of the operations for retrieving attribute values is of a universal charstring type in case of attributes that can be present only once (**display**, **optional**) . If such an attribute is not defined, the operation returns an empty string. If the attribute can occur multiple times (**encode**, **variant**, **extension**), the operation return value is of a record of universal charstring type. If such an attribute is not present, the operation returns an empty record of value.

The operation for getting a variant attribute value may be followed by an optional parameter. If no parameter is present, the operation returns only variants that are not bound to any particular encoding. If the parameter is present, the returned value will containt variants that are bound to the encoding referenced in the parameter.

***Restrictions***

In addition to the general static rules of TTCN‑3 given in clause 5, the following restrictions apply:

1. The optional parameter of the operation shall be used only for getting variant attributes
2. The *Expression* in optional parameter of the operation shall be of the universal charstring type
3. An error shall be produced if the *Expression* in the optional parameters is not one of the valid encode attributes

EXAMPLE 1:

// MyPDU1 will be displayed as PDU

**type** **record** MyPDU1 { ... } **with** {

**display** "blue";

**variant** "CommonRule";

**encode** "Codec1";

**variant** "Codec1"."Rule1A";

**variant** "Codec1"."Rule1B";

**encode** "Codec2";

**variant** "Codec2"."Rule2A";

**variant** "Codec2"."Rule2B";

}

**type** **record** **of** **universal** **charstring** RUC;

**control** {

**var** MyPDU1 v\_pdu;

**var universal charstring** v\_display;

**var** RUC v\_encoding, v\_variants;

v\_display := MyPDU1.**display**; // v\_display will contain "blue"

v\_display := v\_variants.**display**; // v\_display will contain "" as no display attribute is

// defined for v\_variants

v\_encoding := v\_pdu.**encode**; // v\_encoding will contain { "Codec1", "Codec2" }

v\_variants := v\_pdu.**variant**; // v\_variants will contain { "CommonRule" }

// retrieve variants for all defined encodings

**for** (**var** **integer** i := 0; i < **sizeof**(v\_encoding); i := i + 1) {

v\_variants := v\_pdu.**variant**(v\_encoding[i]);

...

}

v\_variants := v\_variants.**encode**; // v\_variants will contain {} as no encode attribute is

// defined for v\_variants

v\_variants := v\_pdu.**variant**("UnknownCodec"); // produces an error as there is no such

// encode attribute as "UnknownCodec"

}

#### A.1.6.8.3 Basic statements

534. OpCall ::= [ConfigurationOps](#TConfigurationOps) |

[GetLocalVerdict](#TGetLocalVerdict) |

[TimerOps](#TTimerOps) |

[TestcaseInstance](#TTestcaseInstance) |

([FunctionInstance](#TFunctionInstance) [[ExtendedFieldReference](#TExtendedFieldReference)]) |

([TemplateOps](#TTemplateOps) [[ExtendedFieldReference](#TExtendedFieldReference)]) |

[ActivateOp](#TActivateOp) |

GetAttributeOp

???. GetAttributeOp ::= ( [Type](#TType) | TemplateInstance ) "." GetAttributeSpec

???. GetAttributeSpec ::= [EncodeKeyword](#TEncodeKeyword) |

[VariantKeyword](#TVariantKeyword) ["(" [FreeText](#TFreeText) ")"] |

[DisplayKeyword](#TDisplayKeyword) |

[ExtensionKeyword](#TExtensionKeyword) |

[OptionalKeyword](#TOptionalKeyword)