Annex A
(normative)

Syntax and static semantics of TTCN

A.1 Introduction

This annex defines the syntax and the static semantics of TTCN. There are two forms of TTCN, a graphical form (TTCN.GR) and a machine processable form (TTCN.MP). For the human user the graphical form of TTCN, the TTCN.GR, takes advantage of an easily understood visual interpretation. However, TTCN.GR does not readily lend itself to machine processing. The TTCN.MP addresses this problem and serves the following purposes:

a) to provide a formal syntax for TTCN in BNF;

b) to act as a transfer syntax;

c) to ease automated derivation of ETSs from ATSs;

d) other machine processing.

NOTE - Automated derivation of ETSs is outside the scope of this part of ISO/IEC 9646.

This annex also defines the static semantics for both TTCN.GR and TTCN.MP.

A.2 Conventions for the syntax description

A.2.1 Syntactic metanotation

Table 1 defines the metanotation used to specify the extended form of BNF grammar for TTCN (henceforth called BNF):

Table A.1 - The TTCN.MP Syntactic Metanotation

::=	is defined to be
abc xyz	abc followed by xyz
|	alternative
[abc]	0 or 1 instances of abc
{abc}	0 or more instances of abc
{abc}+	1 or more instances of abc
(...)	textual grouping
abc	the non-terminal symbol abc
abc	a terminal symbol abc
"abc"	a terminal symbol abc

In the metanotation, concatenation binds more tightly than the alternative operator. Hence “abc def | ghi jkl” is equivalent to “(abc def) | (ghi jkl)”.

A.2.2 TTCN.MP syntax definitions

A.2.2.1 Complete tables defined in TTCN.GR are represented in TTCN.MP by productions of the kind:

$Begin_KEYWORD .... **$End_KEYWORD**

EXAMPLE A.1 - TS_PARdcls ::= **$Begin_TS_PARdcls** {TS_PARdcl}+ **$End_TS_PARdcls**

Normally, these productions contain at least one mandatory component.

A.2.2.2 Both sets of lines of a table and individual lines (*i.e.*, sets of fields in a table) are represented by productions of the kind:

$KEYWORD .... **$End_KEYWORD**

Begin does not appear in the opening keyword.

EXAMPLE A.2 - TS_PARdcl ::= **$TS_PARdcl** TS_PARid TS_PARtype PICS_PIXIT [Comment] **$End_TS_PARdcl**

A.2.2.3 Individual fields in a line are represented by:

$KEYWORD ....

There is no closing keyword.

EXAMPLE A.3 - TS_ParIid ::= **$TS_ParId** TS_ParIdentifier

EXAMPLE A.4 - TS_ParIidentifier ::= Identifier

A.2.2.4 Sets of tables, up to and including the test suite, are represented by productions of the kind:

$KEYWORD .... **$End_KEYWORD**

EXAMPLE A.5 - ASP_TypeDefs ::= **$ASP_TypeDefs** [TTCN_ASP_TypeDefs] [ASN1_ASP_TypeDefs] **$End_ASP_TypeDefs**

A.2.2.5 All other productions defining non-terminal symbols have no keywords at the beginning or the end of the right-hand expression.

EXAMPLE A.6 - TimerIdentifier ::= Identifier

A.2.2.6 When parsing TTCN.MP, any symbol not allowed within an identifier may denote the end of an identifier. In those cases in which it is necessary to insert a meaningless character at the end of an identifier in order to separate it from another identifier or keyword (e.g. when an identifier is followed by a keyword such as **BY** or **OR**) then the recommended separators are space and tab characters.

A.3 The TTCN.MP syntax productions in BNF

A.3.1 TTCN Specification

1 TTCN_Specification ::= TTCN_Module | Suite

A.3.2 TTCN Module

2 TTCN_Module ::= **$TTCN_Module** TTCN_ModuleId TTCN_ModuleOverviewPart [TTCN_ModuleImportPart] [DeclarationsPart] [ConstraintsPart] [DynamicPart] **$End_TTCN_Module**

3 TTCN_ModuleId ::=**$TTCN_ModuleId** TTCN_ModuleIdentifier

4 TTCN_ModuleIdentifier ::= Identifier

A.3.2.1 TTCN Module Overview Part

5 TTCN_ModuleOverviewPart ::= **$TTCN_ModuleOverviewPart** TTCN_ModuleExports [TTCN_ModuleStructure] [TestCaseIndex] [TestStepIndex] [DefaultIndex] **$End_TTCN_ModuleOverviewPart**

A.3.2.1.1 TTCN Module Exports

6 TTCN_ModuleExports ::= **$Begin_TTCN_ModuleExports** TTCN_ModuleId [TTCN_ModuleRef] [TTCN_ModuleObjective] [StandardsRef] [PICSref] [PIXITref] [TestMethods] [Comment] ExportedObjects [Comment] **$End_TTCN_ModuleExports**

7 TTCN_ModuleRef ::= **$TTCN_ModuleRef** BoundedFreeText

8 TTCN_ModuleObjective ::= **$TTCN_ModuleObjective** BoundedFreeText

9 ExportedObjects ::= **$ExportedObjects** {ExportedObject} **$End_ExportedObjects**

10 ExportedObject ::= **$ExportedObject** ObjectId ObjectType [SourceInfo] [Comment] **$End_ExportedObject**

11 ObjectId ::= **$ObjectId** ObjectIdentifier

12 ObjectIdentifier ::= Identifier | ObjectTypeReference

13 ObjectTypeReference ::= Identifier "[" Identifier "]"

/* STATIC SEMANTICS - The first Identifier is a NamedNumber or an Enumeration and the Identifier contained in brackets is the name of the corresponding type. */

14 ObjectType ::= **$ObjectType** TTCN_ObjectType

15 TTCN_ObjectType ::= **SimpleType_Object** | **StructType_Object** | **ASN1_Type_Object** | **TS_Op_Object** | **TS_Proc_Object** | **TS_Par_Object** | **SelectExpr_Object** | **TS_Const_Object** | **TS_Var_Object** | **TC_Var_Object** | **PCO_Type_Object** | **PCO_Object** | **CP_Object** | **Timer_Object** | **TComp_Object** | **TCompConfig_Object** | **TTCN_ASP_Type_Object** | **ASN1_ASP_Type_Object** | **TTCN_PDU_Type_Object** | **ASN1_PDU_Type_Object** | **TTCN_CM_Type_Object** | **ASN1_CM_Type_Object** | **EncodingRule_Object** | **EncodingVariation_Object** | **InvalidFieldEncoding_Object** | **Alias_Object** | **StructTypeConstraint_Object** | **ASN1_TypeConstraint_Object** | **TTCN_ASP_Constraint_Object** | **ASN1_ASP_Constraint_Object** | **TTCN_PDU_constraint_Object** | **ASN1_PDU_Constraint_Object** | **TTCN_CM_Constraint_Object** | **ASN1_CM_Constraint_Object** | **TestCase_Object** | **TestStep_Object** | **Default_Object** | **NamedNumber_Object** | **Enumeration_Object**

16 SourceInfo ::= **$SourceInfo** (SourceIdentifier | ObjectDirective)

/* STATIC SEMANTICS - The SourceIdentifier is the name of the original source object ~~or the source package~~. */

17 SourceIdentifier ::= SuiteIdentifier | TTCN_ModuleIdentifier

18 ObjectDirective ::= Omit | **EXTERNAL**

A.3.2.1.2 TTCN Module Structure

19 TTCN_ModuleStructure ::= **$Begin_TTCN_ModuleStructure** {Structure&Objective}+ [Comment] **$End_TTCN_ModuleStructure**

A.3.2.2 TTCN Module Import Part

20 TTCN_ModuleImportPart ::= **$TTCN_ModuleImportPart** [ExternalObjects] [ImportDeclarations] **$End_TTCN_ModuleImportPart**

A.3.2.2.1 External Objects

21 ExternalObjects ::= **$Begin_ExternalObjects** [ExternalGroupId] {ExternalObject}+ [Comment] **$End_ExternalObjects**

22 ExternalGroupId ::= **$ExternalGroupId** ExternalGroupIdentifier

23 ExternalObject ::= **$ExternalObject** ExternalObjectId ObjectType [Comment] **$End_ExternalObject**

24 ExternalObjectId ::= **$ExternalObjectId** ExternalObjectIdentifier

25 ExternalObjectIdentifier ::= ObjectIdentifier | TS_OpId&ParList | ConsId&ParList | TestStepId&ParList

A.3.2.2.2 Import Declarations

26 ImportDeclarations ::= **$ImportDeclarations** {Imports}+ **$End_ImportDeclarations**

27 Imports ::= **$Begin_Imports** SourceId [SourceRef] [StandardsRef] [Comment] ImportedObjects [Comment] **$End_Imports**

28 SourceId ::= **$SourceId** SourceIdentifier

29 SourceRef ::= **$SourceRef** BoundedFreeText

30 ImportedObjects ::= **$ImportedObjects** {ImportedObject}+ **$End_ImportedObjects**

31 ImportedObject ::= **$ImportedObject** ObjectId ObjectType [SourceInfo] [Comment] **$End_ImportedObject**

A.3.3 Test suite

32 Suite ::= **$Suite** SuiteId SuiteOverviewPart [ImportPart] DeclarationsPart ConstraintsPart DynamicPart **$End_Suite**
/* STATIC SEMANTICS - SuiteId shall be the same as the SuiteId declared in TestSuiteStructure table (Suite Structure). */

33 SuiteId ::= **$SuiteId** SuiteIdentifier

34 SuiteIdentifier ::= Identifier

A.3.3.1 The Test Suite Overview

35 SuiteOverviewPart::= **$SuiteOverviewPart** [TestSuiteIndex] SuiteStructure TestCaseIndex [TestStepIndex] [DefaultIndex] [TestSuiteExports] **$End_SuiteOverviewPart**

A.3.3.2 Test Suite Index

36 TestSuiteIndex ::= **$Begin_TestSuiteIndex** {ObjectInfo} [Comment] **$End_TestSuiteIndex**

A.3.3.2.1 The Imported Object Info

37 ObjectInfo ::= **$ObjectInfo** ObjectId ObjectType SourceId OrigObjectId [PageNum] [Comment] **$End_ObjectInfo**

38 PageNum ::= **$PageNum** PageNumber

39 PageNumber ::= Number

40 OrigObjectId ::= **$OrigObjectId** ObjectIdentifier

A.3.3.3 Test Suite Structure

41 SuiteStructure ::= **$Begin_SuiteStructure** SuiteId StandardsRef PICSref PIXITref TestMethods [Comment] Structure&Objectives [Comment] **$End_SuiteStructure**

42 StandardsRef ::= **$StandardsRef** BoundedFreeText

43 PICSref ::= **$PICSref** BoundedFreeText

44 PIXITref ::= **$PIXITref** BoundedFreeText

45 TestMethods ::= **$TestMethods** BoundedFreeText

46 Comment ::= **$Comment** [BoundedFreeText]

47 Structure&Objectives ::= **$Structure&Objectives** {Structure&Objective} **$End_Structure&Objectives**

48 Structure&Objective ::= **$Structure&Objective** TestGroupRef SelExprId Objective **$End_Structure&Objective**

49 SelExprId ::=**$SelectExprId** [SelectExprIdentifier]

A.3.3.4 Test Case Index

50 TestCaseIndex ::= **$Begin_TestCaseIndex** {[CollComment] CaseIndex}+ [Comment] **$End_TestCaseIndex**
/* NOTE - Collective comments may be used in this table according to Figure 2. */

51 CollComment ::= **$CollComment** [BoundedFreeText]

52 CaseIndex ::= **$CaseIndex** TestGroupRef TestCaseId SelExprId Description **$End_CaseIndex**
/* STATIC SEMANTICS - Test Cases shall be listed in the order that they exist in the dynamic part. */
/* STATIC SEMANTICS - An explicit TestGroupReference shall be provided for the first TestCase of each TestGroup. */
/* STATIC SEMANTICS - An explicit TestGroupReference shall be provided for each TestCase that immediately follows a TestGroup. */

53 Description ::= **$Description** BoundedFreeText

A.3.3.5 Test Step Index

54 TestStepIndex ::= **$Begin_TestStepIndex** {[CollComment] StepIndex} [Comment] **$End_TestStepIndex**
/* NOTE - Collective comments may be used in this table according to Figure 2. */

55 StepIndex ::= **$StepIndex** TestStepRef TestStepId Description **$End_StepIndex**
/* STATIC SEMANTICS - TestStepId shall not include a formal parameter list. */
/* STATIC SEMANTICS - Test Steps shall be listed in the order that they exist in the dynamic part. */
/* STATIC SEMANTICS - An explicit TestStepGroupReference shall be provided for the first TestStep of each TestStepGroup. */
/* STATIC SEMANTICS - An explicit TestStepGroupReference shall be provided for each TestStep that immediately follows a TestStepGroup. */

A.3.3.6 Default Index

56 DefaultIndex ::= **$Begin_DefaultIndex** {[CollComment] DefIndex} [Comment] **$End_DefaultIndex**
/* NOTE - Collective comments may be used in this table according to Figure 2. */

57 DefIndex ::= **$DefIndex** DefaultRef DefaultId Description **$End_DefIndex**
/* STATIC SEMANTICS - DefaultId shall not include a formal parameter list. */
/* STATIC SEMANTICS - Defaults shall be listed in the order that they exist in the dynamic part. */
/* STATIC SEMANTICS - An explicit DefaultGroupReference shall be provided for the first Default of each DefaultGroup. */
/* STATIC SEMANTICS - An explicit DefaultGroupReference shall be provided for eachDefault that immediately follows a DefaultGroup. */

A.3.3.7 Test Suite Exports

58 TestSuiteExports::= **$Begin_TestSuiteExports** ExportedObjects [Comment] **$End_TestSuiteExports**

A.3.3.8 The Import Part

59 ImportPart ::= **$ImportPart** ImportDeclarations **$End_ImportPart**

A.3.3.9 The Declarations Part

60 DeclarationsPart ::= **$DeclarationsPart** Definitions Parameterization&Selection Declarations ComplexDefinitions **$End_DeclarationsPart**

A.3.3.10 Definitions

A.3.3.10.1 General

61 Definitions ::= [TS_TypeDefs] [EncodingDefs] [TS_OpDefs] [TS_ProcDefs]

A.3.3.10.2 Test Suite Type Definitions

62 TS_TypeDefs ::= **$TS_TypeDefs** [SimpleTypeDefsOrGroup] [StructTypeDefs] [ASN1_TypeDefs] [ASN1_TypeRefsOrGroup] **$End_TS_TypeDefs**

A.3.3.10.3 Simple Type Definitions

63 SimpleTypeDefsOrGroup ::= SimpleTypeDefs | SimpleTypeGroup

64 SimpleTypeGroup ::= **$SimpleTypeGroup** SimpleTypeGroupId {SimpleTypeDefsOrGroup}+ **$End_SimpleTypeGroup**

65 SimpleTypeGroupId ::= **$SimpleTypeGroupId** SimpleTypeGroupIdentifier

66 SimpleTypeDefs ::= **$Begin_SimpleTypeDefs** [SimpleTypeGroupRef] {[CollComment] SimpleTypeDef}+ [Comment] **$End_SimpleTypeDefs**
/* NOTE - Collective comments may be used in this table according to Figure 2. */

67 SimpleTypeGroupRef ::= **$SimpleTypeGroupRef** SimpleTypeGroupReference

68 SimpleTypeGroupReference ::= [(SuiteIdentifier | TTCN_ModuleIdentifier) “/”] {SimpleTypeGroupIdentifier “/”}

69 SimpleTypeGroupIdentifier ::= Identifier

70 SimpleTypeDef ::= **$SimpleTypeDef** SimpleTypeId SimpleTypeDefinition [PDU_FieldEncoding] [Comment] **$End_SimpleTypeDef**

71 SimpleTypeId ::= **$SimpleTypeId** SimpleTypeIdentifier

72 SimpleTypeIdentifier ::= Identifier

73 SimpleTypeDefinition ::= **$SimpleTypeDefinition** Type&Restriction

/* STATIC SEMANTICS - There shall be no recursive references (neither directly nor indirectly) in Type&Restriction. */

74 Type&Restriction ::= Type [Restriction]

/* STATIC SEMANTICS - Type shall be either PredefinedType or SimpleType. */

75 Restriction ::= LengthRestriction | IntegerRange | SimpleValueList

/* STATIC SEMANTICS - The set of values defined by Restriction shall be a true subset of the values of the base type. */

76 LengthRestriction ::= SingleTypeLength | RangeTypeLength

/* STATIC SEMANTICS - LengthRestriction shall be provided only when the base type is a string type (i.e., BITSTRING, HEXSTRING, OCTETSTRING or CharacterString) or derived from a string type. */

77 SingleTypeLength ::= "[" Number "]"

78 RangeTypeLength ::= "[" LowerTypeBound To UpperTypeBound "]"

/* STATIC SEMANTICS - LowerTypeBound shall be a non-negative number. */

/* STATIC SEMANTICS - LowerTypeBound shall be less than UpperTypeBound. */

79 IntegerRange ::= "(" LowerTypeBound To UpperTypeBound ")"

/* STATIC SEMANTICS - LowerTypeBound shall be less than UpperTypeBound. */

80 LowerTypeBound ::= [Minus] Number | Minus **INFINITY**

81 UpperTypeBound ::= [Minus] Number | **INFINITY**

82 To ::= **TO** | ".."

83 SimpleValueList ::= "(" [Minus] LiteralValue {Comma [Minus] LiteralValue} ")"

/* STATIC SEMANTICS - If Minus is used in SimpleValueList then LiteralValue shall be a number. */

/* STATIC SEMANTICS - The LiteralValues shall be of the base type and shall be a true subset of the values defined by the base type. */

A.3.3.10.4 Structured Type Definitions

84 StructTypeDefs ::= **$StructTypeDefs** {StructTypeDefOrGroup}+ **$End_StructTypeDefs**

85 StructTypeDefOrGroup ::= StructTypeDef | StructTypeGroup

86 StructTypeGroup ::= **$StructTypeGroup** StructTypeGroupId {StructTypeDefOrGroup}+ **$End_StructTypeGroup**

87 StructTypeGroupId ::= **$StructTypeGroupId** StructTypeGroupIdentifier

88 StructTypeDef ::= **$Begin_StructTypeDef** StructId [StructTypeGroupRef] [EncVariationId] [Comment] ElemDcls [Comment] **$End_StructTypeDef**

89 StructId ::= **$StructId** StructId&FullId

90 StructId&FullId ::= StructIdentifier [FullIdentifier]

91 FullIdentifier ::= "(" BoundedFreeText ")"

/* STATIC SEMANTICS - Some TTCN objects allow names, as given in the appropriate protocol standard to be abbreviated. If an abbreviation is used then FullIdentifier shall be given in the declaration of the object. */

92 StructIdentifier ::= Identifier

93 StructTypeGroupRef ::= **$StructTypeGroupRef** StructTypeGroupReference

94 StructTypeGroupReference ::= [(SuiteIdentifier | TTCN_ModuleIdentifier) “/”] {StructTypeGroupIdentifier “/”}

95 StructTypeGroupIdentifier ::= Identifier

96 ElemDcls ::= **$ElemDcls** {ElemDcl}+ **$End_ElemDcls**

97 ElemDcl ::= **$ElemDcl** ElemId ElemType [PDU_FieldEncoding] [Comment] **$End_ElemDcl**

98 ElemId ::= **$ElemId** ElemId&FullId

99 ElemId&FullId ::= ElemIdentifier [FullIdentifier]

100 ElemIdentifier ::= Identifier

101 ElemType ::= **$ElemType** Type&Attributes

/* STATIC SEMANTICS - There shall be no recursive references (neither directly nor indirectly) in Type&Attributes. */

/* STATIC SEMANTICS - A structure element Type shall be a PredefinedType, TS_TypeIdentifier, PDU_Identifier, or PDU. */

A.3.3.10.5 ASN.1 Type Definitions

102 ASN1_TypeDefs ::= **$ASN1_TypeDefs** {ASN1_TypeDefOrGroup}+ **$End_ASN1_TypeDefs**

103 ASN1_TypeDefOrGroup ::= ASN1_TypeDef | ASN1_TypeGroup

104 ASN1_TypeGroup ::= **$ASN1_TypeGroup** ASN1_TypeGroupId {ASN1_TypeDefOrGroup}+ **$End_ASN1_TypeGroup**

105 ASN1_TypeGroupId ::= **$ASN1_TypeGroupId** ASN1_TypeGroupIdentifier

106 ASN1_TypeDef ::= **$Begin_ASN1_TypeDef** ASN1_TypeId [ASN1_TypeGroupRef] [EncVariationId] [Comment] ASN1_TypeDefinition [Comment] **$End_ASN1_TypeDef**

107 ASN1_TypeId ::= **$ASN1_TypeId** ASN1_TypeId&FullId

108 ASN1_TypeId&FullId ::= ASN1_TypeIdentifier [FullIdentifier]

109 ASN1_TypeIdentifier ::= Identifier

110 ASN1_TypeGroupRef ::= **$ASN1_TypeGroupRef** ASN1_TypeGroupReference

111 ASN1_TypeGroupReference ::= [(SuiteIdentifier | TTCN_ModuleIdentifier) “/”] {ASN1_TypeGroupIdentifier “/”}

112 ASN1_TypeGroupIdentifier ::= Identifier

113 ASN1_TypeDefinition ::= **$ASN1_TypeDefinition** ASN1_Type&LocalTypes **$End_ASN1_TypeDefinition**

114 ASN1_Type&LocalTypes ::= ASN1_Type {ASN1_LocalType}

/* STATIC SEMANTICS - Types referred to from the ASN1_Type definition shall be defined in other ASN.1 type definition tables, be defined by reference in the ASN.1 type reference table or be defined locally (*i.e.,* ASN1_LocalTypes) in the same table, following the first type definition. */

/* STATIC SEMANTICS - ASN1_LocalTypes shall not be used in other parts of the test suite. */

115 ASN1_Type ::= <u>Type</u>

/* REFERENCE - Where Type is a non-terminal defined in ISO/IEC 8824: 1990.
For the purposes of TTCN, the production in ISO/IEC 8824: 1990 which states:

```
        Type ::= BuiltinType | DefinedType | Subtype
```

is redefined to be

```
        Type ::= (BuiltinType | DefinedType | Subtype) [ASN1_Encoding]
```

This means that ASN1_Encoding can be applied to the whole of an ASN1_Type or any ASN.1 Type within the ASN1_Type. */

/* STATIC SEMANTICS - Each terminal type reference used within the Type production shall be one of the following: ASN1_LocalType typereference, TS_TypeIdentifier or PDU_Identifier. */

/* STATIC SEMANTICS - ASN.1 type definitions used within TTCN shall not use external type references as defined in ISO/IEC 8824: 1990. */

116 ASN1_LocalType ::= <u>Typeassignment</u>

/* REFERENCE - Where Typeassignment is a non-terminal defined in ISO/IEC 8824: 1990. */

/* STATIC SEMANTICS - ASN.1 type definitions used within TTCN shall not use external type references as defined in ISO/IEC 8824: 1990. */

A.3.3.10.6 ASN.1 Type Definitions by Reference

117 ASN1_TypeRefsOrGroup ::= ASN1_TypeRefs | ASN1_TypeRefsGroup

118 ASN1_TypeRefsGroup ::= **$ASN1_TypeRefsGroup** ASN1_TypeRefsGroupId {ASN1_TypeRefsOrGroup}+ **$End_ASN1_TypeRefsGroup**

119 ASN1_TypeRefsGroupId ::= **$ASN1_TypeRefsGroupId** ASN1_TypeGroupIdentifier

120 ASN1_TypeRefs ::= **$Begin_ASN1_TypeRefs** [ASN1_TypeRefsGroupRef] {[CollComment] ASN1_TypeRef}+ [Comment] **$End_ASN1_TypeRefs**

/* NOTE - Collective comments may be used in this table according to Figure 2. */

121 ASN1_TypeRefsGroupRef ::= **$ASN1_TypeRefsGroupRef** ASN1_TypeGroupReference

122 ASN1_TypeGroupIdentifier ::= Identifier

123 ASN1_TypeRef ::= **$ASN1_TypeRef** ASN1_TypeId ASN1_TypeReference ASN1_ModuleId [EncVariationId] [Comment] **$End_ASN1_TypeRef**

/* STATIC SEMANTICS - ASN1_TypeId shall not be specified with a FullIdentifier. */

124 ASN1_TypeReference ::= **$ASN1_TypeReference** TypeReference

125 TypeReference ::= typereference

/* REFERENCE - Where typereference is a non-terminal defined in ISO/IEC 8824:1990. */

126 ASN1_ModuleId ::= **$ASN1_ModuleId** ASN1_ModuleIdentifier

127 ASN1_ModuleIdentifier ::= ModuleIdentifier

/* REFERENCE - Where ModuleIdentifier is a non-terminal defined in ISO/IEC 8824: 1990. */

/* STATIC SEMANTICS - ModuleIdentifier shall be unique within the domain of interest. */

A.3.3.10.7 Test Suite Operation Definitions

128 TS_OpDefs ::= **$TS_OpDefs** {TS_OpDefOrGroup}+ **$End_TS_OpDefs**

129 TS_OpDefOrGroup ::= TS_OpDef | TS_OpDefGroup

130 TS_OpDefGroup ::= **$TS_OpDefGroup** TS_OpDefGroupId {TS_OpDefOrGroup}+ **$End_TS_OpDefGroup**

131 TS_OpDefGroupId ::= **$TS_OpDefGroupId** TS_OpDefGroupIdentifier

132 TS_OpDef ::= **$Begin_TS_OpDef** TS_OpId [TS_OpGroupRef] TS_OpResult [Comment] TS_OpDescription [Comment] **$End_TS_OpDef**

133 TS_OpId ::= **$TS_OpId** TS_OpId&ParList

134 TS_OpId&ParList ::= TS_OpIdentifier [FormalParList]

/* STATIC SEMANTICS - A Test Suite Operation formal parameter Type shall be a PredefinedType, TS_TypeIdentifier, PDU_Identifier or ASP_Identifier, or the meta-type **PDU***/

135 TS_OpIdentifier ::= Identifier

136 TS_OpGroupRef ::= **$TS_OpGroupRef** TS_OpGroupReference

137 TS_OpGroupReference ::= [(SuiteIdentifier | TTCN_ModuleIdentifier) "/"] {TS_OpGroupIdentifier "/"}

138 TS_OpGroupIdentifier ::= Identifier

139 TS_OpResult ::= **$TS_OpResult** TypeOrPDU

/* STATIC SEMANTICS - TypeOrPDU shall be a PredefinedType, TS_TypeIdentifier, PDU_Identifier or ASP_Identifier, or the meta-type **PDU**. */

140 TS_OpDescription ::= **$TS_OpDescription** BoundedFreeText

A.3.3.10.8 Test Suite Operation Procedural Definitions

141 TS_ProcDefs ::= **$TS_ProcDefs** {TS_ProcDefOrGroup}+ **$End_TS_ProcDefs**

142 TS_ProcDefOrGroup ::= TS_ProcDef | TS_ProcDefGroup

143 TS_ProcDefGroup ::= **$TS_ProcDefGroup** TS_ProcDefGroupId {TS_ProcDefOrGroup}+ **$End_TS_ProcDefGroup**

144 TS_ProcDefGroupId ::= **$TS_ProcDefGroupId** TS_ProcDefGroupIdentifier

145 TS_ProcDef ::= **$Begin_TS_ProcDef** TS_ProcId [TS_ProcGroupRef] TS_ProcResult [Comment] TS_ProcDescription [Comment] **$End_TS_ProcDef**

/* LEXICAL REQUIREMENT - Comments may be embedded within TS_ProcDescription by enclosing them within "/*" and "*/" but may not be nested. They may be carried within TTCN.MP but shall be removed before parsing the TTCN.MP. */

146 TS_ProcId ::= **$TS_ProcId** TS_ProcId&ParList

147 TS_ProcId&ParList ::= TS_ProcIdentifier [FormalParList]

/* STATIC SEMANTICS - A procedural Test Suite Operation formal parameter Type shall be a PredefinedType, TS_TypeIdentifier, PDU_Identifier or ASP_Identifier, or the meta-type **PDU***/

148 TS_ProcIdentifier ::= Identifier

149 TS_ProcGroupRef ::= **$TS_ProcGroupRef** TS_ProcGroupReference

150 TS_ProcGroupReference ::= [(SuiteIdentifier | TTCN_ModuleIdentifier) "/"] {TS_ProcGroupIdentifier "/"}

151 TS_ProcGroupIdentifier ::= Identifier

152 TS_ProcResult ::= **$TS_ProcResult** TypeOrPDU

/* STATIC SEMANTICS - TypeOrPDU shall be a PredefinedType, TS_TypeIdentifier, PDU_Identifier or ASP_Identifier, or the meta-type **PDU**. */

153 TS_ProcDescription ::= **$TS_ProcDescription** TS_OpProcDef **$End_TS_ProcDescription**

154 TS_OpProcDef ::= [VarBlock] ProcStatement

/* NOTE - Comments are allowed within TS_OpProcDef, starting with "/*" and ending with "*/", but it is assumed that these comments are removed before the syntax is parsed. Hence the BNF does not include the syntax of such embedded comments. */

155 VarBlock ::= **VAR** VarDcls **ENDVAR**

156 VarDcls ::= {VarDcl SemiColon}

157 VarDcl ::= [**STATIC**] VarIdentifiers Colon TypeOrPDU [Colon Value]

158 VarIdentifiers ::= VarIdentifier {Comma VarIdentifier}

159 VarIdentifier ::= Identifier

160 ProcStatement ::= ReturnValueStatement | Assignment | IfStatement | WhileLoop | CaseStatement | ProcBlock

161 ReturnValueStatement ::= **RETURNVALUE** Expression

162 IfStatement ::= **IF** Expression **THEN** {ProcStatement SemiColon}+ [**ELSE** {ProcStatement SemiColon}+] **ENDIF**

163 WhileLoop ::= **WHILE** Expression **DO** {ProcStatement SemiColon}+ **ENDWHILE**

164 CaseStatement ::= **CASE** Expression **OF** {CaseClause SemiColon}+ [**ELSE** {ProcStatement SemiColon}+] **ENDCASE**

165 CaseClause ::= IntegerLabel Colon ProcStatement

166 IntegerLabel ::= Number | TS_ParIdentifier | TS_ConstIdentifier

167 ProcBlock ::= **BEGIN** {ProcStatement SemiColon}+ **END**

A.3.3.11 Parameterization and Selection

A.3.3.11.1 General

168 Parameterization&Selection ::= [TS_ParDclsOrGroup] [SelectExprDefsOrGroup]

A.3.3.11.2 Test Suite Parameter Declarations

169 TS_ParDclsOrGroup ::= TS_ParDcls | TS_ParDclsGroup

170 TS_ParDclsGroup ::= **$TS_ParDclsGroup** TS_ParDclsGroupId {TS_ParDclsOrGroup}+ **$End_TS_ParDclsGroup**

171 TS_ParDclsGroupId ::= **$TS_ParDclsGroupId** TS_ParDclsGroupIdentifier

172 TS_ParDcls ::= **$Begin_TS_ParDcls** [TS_ParGroupRef] {[CollComment] TS_ParDcl}+ [Comment] **$End_TS_ParDcls**
/* NOTE - Collective comments may be used in this table according to Figure 2. */

173 TS_ParGroupRef ::= **$TS_ParGroupRef** TS_ParGroupReference

174 TS_ParGroupReference ::= [(SuiteIdentifier | TTCN_ModuleIdentifier) “/”] {TS_ParGroupIdentifier “/”}

175 TS_ParGroupIdentifier ::= Identifier

176 TS_ParDcl ::= **$TS_ParDcl** TS_ParId TS_ParType PICS_PIXITref [Comment] **$End_TS_ParDcl**

177 TS_ParId ::= **$TS_ParId** TS_ParIdentifier

178 TS_ParIdentifier ::= Identifier

179 TS_ParType ::= **$TS_ParType** TypeOrPDU
/* STATIC SEMANTICS - TypeOrPDU shall be a PredefinedType, TS_TypeIdentifier, PDU_Identifier or ASP_Identifier, or the meta-type **PDU**. */

180 PICS_PIXITref ::= **$PICS_PIXITref** BoundedFreeText

A.3.3.11.3 Test Case Selection Expression Definitions

181 SelectExprDefsOrGroup ::= SelectExprDefs | SelectExprDefsGroup

182 SelectExprDefsGroup ::= **$SelectExprDefsGroup** SelectExprDefsGroupId {SelectExprDefsOrGroup}+ **$End_SelectExprDefsGroup**

183 SelectExprDefsGroupId ::= **$SelectExprDefsGroupId** SelectExprDefsGroupIdentifier

184 SelectExprDefs ::= **$Begin_SelectExprDefs** [SelectExprGroupRef] {[CollComment] SelectExprDef}+ [Comment] **$End_SelectExprDefs**
/* NOTE - Collective comments may be used in this table according to Figure 2. */

185 SelectExprGroupRef ::= **$SelectExprGroupRef** SelectExprGroupReference

186 SelectExprGroupReference ::= [(SuiteIdentifier | TTCN_ModuleIdentifier) “/”] {SelectExprGroupIdentifier “/”}

187 SelectExprGroupIdentifier ::= Identifier

188 SelectExprDef ::= **$SelectExprDef** SelectExprId SelectExpr [Comment] **$End_SelectExprDef**

189 SelectExprId ::= **$SelectExprId** SelectExprIdentifier

190 SelectExprIdentifier ::= Identifier

191 SelectExpr ::= **$SelectExpr** SelectionExpression

192 SelectionExpression ::= Expression

/* STATIC SEMANTICS - SelectionExpression shall only contain LiteralValues, TS_ParIdentifiers, TS_ConstIdentifiers and SelectExprIdentifiers*/

/* OPERATIONAL SEMANTICS - SelectionExpression shall evaluate to a specific BOOLEAN value. */

/* STATIC SEMANTICS - Expression shall not recursively refer (neither directly nor indirectly) to the SelExprIdentifier being defined by that Expression. */

A.3.3.12 Declarations

A.3.3.12.1 General

193 Declarations ::= [TS_ConstDclsOrGroup] [TS_ConstRefsOrGroup] [TS_VarDclsOrGroup] [TC_VarDclsOrGroup] [PCO_TypeDclsOrGroup] [PCO_DclsOrGroup] [CP_DclsOrGroup] [TimerDclsOrGroup] [TCompDclsOrGroup TCompConfigDcls]

/* STATIC SEMANTICS - PCOs shall be optional */

A.3.3.12.2 Test Suite Constant Declarations

194 TS_ConstDclsOrGroup ::= TS_ConstDcls | TS_ConstDclsGroup

195 TS_ConstDclsGroup ::= **$TS_ConstDclsGroup** TS_ConstDclsGroupId {TS_ConstDclsOrGroup}+ **$End_TS_ConstDclsGroup**

196 TS_ConstDclsGroupId ::= **$TS_ConstDclsGroupId** TS_ConstDclsGroupIdentifier

197 TS_ConstDcls ::= **$Begin_TS_ConstDcls** [TS_ConstGroupRef] {[CollComment] TS_ConstDcl}+ [Comment] **$End_TS_ConstDcls**

/* NOTE - Collective comments may be used in this table according to Figure 2. */

198 TS_ConstGroupRef ::= **$TS_ConstGroupRef** TS_ConstGroupReference

199 TS_ConstGroupReference ::= [(SuiteIdentifier | TTCN_ModuleIdentifier) “/”] {TS_ConstGroupIdentifier “/”}

200 TS_ConstGroupIdentifier ::= Identifier

201 TS_ConstDcl ::= **$TS_ConstDcl** TS_ConstId TS_ConstType TS_ConstValue [Comment] **$End_TS_ConstDcl**

202 TS_ConstId ::= **$TS_ConstId** TS_ConstIdentifier

203 TS_ConstIdentifier ::= Identifier

204 TS_ConstType ::= **$TS_ConstType** Type

/* STATIC SEMANTICS - Type shall be a PredefinedType, TS_TypeIdentifier, PDU_Identifier or ASP_Identifier. */

205 TS_ConstValue ::= **$TS_ConstValue** DeclarationValue

206 DeclarationValue ::= Expression

/* STATIC SEMANTICS - DeclarationValue shall only contain LiteralValues, TS_ParIdentifiers and TS_ConstIdentifiers and operators and operations applying to such constant values. */

/* OPERATIONAL SEMANTICS - DeclarationValue shall evaluate to an element of its declared type. */

A.3.3.12.3 Test Suite Constant Declarations by Reference

207 TS_ConstRefsOrGroup ::= TS_ConstRefs | TS_ConstRefsGroup

208 TS_ConstRefsGroup ::= **$TS_ConstRefsGroup** TS_ConstRefsGroupId {TS_ConstRefsOrGroup}+ **$End_TS_ConstRefsGroup**

209 TS_ConstRefsGroupId ::= **$TS_ConstRefsGroupId** TS_ConstRefsGroupIdentifier

210 TS_ConstRefs ::= **$Begin_TS_ConstRefs** [TS_ConstRefsGroupRef] {[CollComment] TS_ConstRef}+ [Comment] **$End_TS_ConstRefs**

/* NOTE - Collective comments may be used in this table according to Figure 2. */

211 TS_ConstRefsGroupRef ::= **$TS_ConstRefsGroupRef** TS_ConstGroupReference

212 TS_ConstRef ::= **$TS_ConstRef** TS_ConstId TS_ConstType ASN1_ValueReference ASN1_ModuleId [Comment] **$End_TS_ConstRef**

213 ASN1_ValueReference ::= **$ASN1_ValueReference** ValueReference

214 ValueReference ::= <u>valuereference</u>

/* REFERENCE - <u>valuereference</u> is a non-terminal defined in ISO/IEC 8824:1990. */

A.3.3.12.4 Test Suite Variable Declarations

215 TS_VarDclsOrGroup ::= TS_VarDcls | TS_VarDclsGroup

216 TS_VarDclsGroup ::= **$TS_VarDclsGroup** TS_VarDclsGroupId {TS_VarDclsOrGroup}+ **$End_TS_VarDclsGroup**

217 TS_VarDclsGroupId ::= **$TS_VarDclsGroupId** TS_VarDclsGroupIdentifier

218 TS_VarDcls ::= **$Begin_TS_VarDcls** [TS_VarGroupRef] {[CollComment] TS_VarDcl}+ [Comment] **$End_TS_VarDcls**
/* NOTE - Collective comments may be used in this table according to Figure 2. */

219 TS_VarGroupRef ::= **$TS_VarGroupRef** TS_VarGroupReference

220 TS_VarGroupReference ::= [(SuiteIdentifier | TTCN_ModuleIdentifier) "/"] {TS_VarGroupIdentifier "/"}

221 TS_VarGroupIdentifier ::= Identifier

222 TS_VarDcl ::= **$TS_VarDcl** TS_VarId TS_VarType TS_VarValue [Comment] **$End_TS_VarDcl**

223 TS_VarId ::= **$TS_VarId** TS_VarIdentifier

224 TS_VarIdentifier ::= Identifier

225 TS_VarType ::= **$TS_VarType** TypeOrPDU
/* STATIC SEMANTICS - TypeOrPDU shall be a PredefinedType, TS_TypeIdentifier, PDU_Identifier or ASP_Identifier, or the meta-type **PDU**. */

226 TS_VarValue ::= **$TS_VarValue** [DeclarationValue]

A.3.3.12.5 Test Case Variable Declarations

227 TC_VarDclsOrGroup ::= TC_VarDcls | TC_VarDclsGroup

228 TC_VarDclsGroup ::= **$TC_VarDclsGroup** TC_VarDclsGroupId {TC_VarDclsOrGroup}+ **$End_TC_VarDclsGroup**

229 TC_VarDclsGroupId ::= **$TC_VarDclsGroupId** TC_VarDclsGroupIdentifier

230 TC_VarDcls ::= **$Begin_TC_VarDcls** [TC_VarGroupRef] {[CollComment] TC_VarDcl}+ [Comment] **$End_TC_VarDcls**
/* NOTE - Collective comments may be used in this table according to Figure 2. */

231 TC_VarGroupRef ::= **$TC_VarGroupRef** TC_VarGroupReference

232 TC_VarGroupReference ::= [(SuiteIdentifier | TTCN_ModuleIdentifier) "/"] {TC_VarGroupIdentifier "/"}

233 TC_VarGroupIdentifier ::= Identifier

234 TC_VarDcl ::= **$TC_VarDcl** TC_VarId TC_VarType TC_VarValue [Comment] **$End_TC_VarDcl**

235 TC_VarId ::= **$TC_VarId** TC_VarIdentifier

236 TC_VarIdentifier ::= Identifier

237 TC_VarType ::= **$TC_VarType** TypeOrPDU
/* STATIC SEMANTICS - TypeOrPDU shall be a PredefinedType, TS_TypeIdentifier, PDU_Identifier or ASP_Identifier, or the meta-type **PDU**. */

238 TC_VarValue ::= **$TC_VarValue** [DeclarationValue]

A.3.3.12.6 PCO Type Declaration

239 PCO_TypeDclsOrGroup ::= PCO_TypeDcls | PCO_TypeDclsGroup

240 PCO_TypeDclsGroup ::= **$PCO_TypeDclsGroup** PCO_TypeDclsGroupId {PCO_TypeDclsOrGroup}+ **$End_PCO_TypeDclsGroup**

241 PCO_TypeDclsGroupId ::= **$PCO_TypeDclsGroupId** PCO_TypeDclsGroupIdentifier

242 PCO_TypeDcls ::= **$Begin_PCO_TypeDcls** [PCO_TypeGroupRef] {PCO_TypeDcl}+ [Comment] **$End_PCO_TypeDcls**

243 PCO_TypeGroupRef ::= **$PCO_TypeGroupRef** PCO_GroupReference

244 PCO_TypeDcl ::= **$PCO_TypeDcl** PCO_TypeId P_Role [Comment] **$End_PCO_TypeDcl**

245 PCO_TypeId ::= **$PCO_TypeId** PCO_TypeIdentifier

246 PCO_TypeIdentifier ::= Identifier

A.3.3.12.7 PCO Declarations

247 PCO_DclsOrGroup ::= PCO_Dcls | PCO_DclsGroup

248 PCO_DclsGroup ::= **$PCO_DclsGroup** PCO_DclsGroupId {PCO_DclsOrGroup}+ **$End_PCO_DclsGroup**

249 PCO_DclsGroupId ::= **$PCO_DclsGroupId** PCO_DclsGroupIdentifier

250 PCO_Dcls ::= **$Begin_PCO_Dcls** [PCO_GroupRef] {[CollComment] PCO_Dcl}+ [Comment] **$End_PCO_Dcls**
/* NOTE - Collective comments may be used in this table according to Figure 2. */
/* STATIC SEMANTICS - In accordance with ISO/IEC 9646-1 the number of PCOs shall relate to the test method used. */

251 PCO_GroupRef ::= **$PCO_GroupRef** PCO_GroupReference

252 PCO_GroupReference ::= [(SuiteIdentifier | TTCN_ModuleIdentifier) “/”] {PCO_GroupIdentifier “/”}

253 PCO_GroupIdentifier ::= Identifier

254 PCO_Dcl ::= **$PCO_Dcl** PCO_Id PCO_TypeId&MuxValue P_Role [Comment] **$End_PCO_Dcl**

255 PCO_Id ::= **$PCO_Id** PCO_Identifier

256 PCO_Identifier ::= Identifier

257 PCO_TypeId&MuxValue ::= **$PCO_TypeId** PCO_TypeIdentifier ["(" MuxValue ")"]

258 MuxValue ::= TS_ParIdentifier

259 P_Role ::= **$PCO_Role** PCO_Role

260 PCO_Role **::= UT** | **LT**

A.3.3.12.8 CP Declarations

261 CP_DclsOrGroup ::= CP_Dcls | CP_DclsGroup

262 CP_DclsGroup ::= **$CP_DclsGroup** CP_DclsGroupId {CP_DclsOrGroup}+ **$End_CP_DclsGroup**

263 CP_DclsGroupId ::= **$CP_DclsGroupId** CP_DclsGroupIdentifier

264 CP_Dcls ::= **$Begin_CP_Dcls** [CP_GroupRef] {[CollComment] CP_Dcl}+ [Comment] **$End_CP_Dcls**
/* NOTE - Collective comments may be used in this table according to Figure 2. */

265 CP_GroupRef ::= **$CP_GroupRef** CP_GroupReference

266 CP_GroupReference ::= [(SuiteIdentifier | TTCN_ModuleIdentifier) “/”] {CP_GroupIdentifier “/”}

267 CP_GroupIdentifier ::= Identifier

268 CP_Dcl ::= **$CP_Dcl** CP_Id [Comment] **$End_CP_Dcl**

269 CP_Id ::= **$CP_Id** CP_Identifier

270 CP_Identifier ::= Identifier

A.3.3.12.9 Timer Declarations

271 TimerDclsOrGroup ::= TimerDcls | TimerDclsGroup

272 TimerDclsGroup ::= **$TimerDclsGroup** TimerDclsGroupId {TimerDclsOrGroup}+ **$End_TimerDclsGroup**

273 TimerDclsGroupId ::= **$TimerDclsGroupId** TimerDclsGroupIdentifier

274 TimerDcls ::= **$Begin_TimerDcls** [TimerGroupRef] {[CollComment] TimerDcl}+ [Comment] **$End_TimerDcls**
/* NOTE - Collective comments may be used in this table according to Figure 2. */

275 TimerGroupRef ::= **$TimerGroupRef** TimerGroupReference

276 TimerGroupReference ::= [(SuiteIdentifier | TTCN_ModuleIdentifier) “/”] {TimerGroupIdentifier “/”}

277 TimerGroupIdentifier ::= Identifier

278 TimerDcl ::= **$TimerDcl** TimerId Duration Unit [Comment] **$End_TimerDcl**

279 TimerId ::= **$TimerId** TimerIdentifier

280 TimerIdentifier ::= Identifier

281 Duration ::= **$Duration** [DeclarationValue]
/* OPERATIONAL SEMANTICS - DeclarationValue shall evaluate to a non-zero positive INTEGER. */

282 Unit ::= **$Unit** TimeUnit

283 TimeUnit ::= **ps** | **ns** | **us** | **ms** | **s** | **min**
/* STATIC SEMANTICS - If a timer is derived from the PICS/PIXIT then the timer declaration shall specify the same units as the PICS/PIXIT entry. */

A.3.3.12.10 Test Component Declarations

284 TCompDclsOrGroup ::= TCompDcls | TCompDclsGroup

285 TCompDclsGroup ::= **$TCompDclsGroup** TCompDclsGroupId {TCompDclsOrGroup}+ **$End_TCompDclsGroup**

286 TCompDclsGroupId ::= **$TCompDclsGroupId** TCompDclsGroupIdentifier

287 TCompDcls ::= **$Begin_TCompDcls** [TCompGroupRef] {[CollComment] TCompDcl}+ [Comment] **$End_TCompDcls**
/* NOTE - Collective comments may be used in this table according to Figure 2. */

288 TCompGroupRef ::= **$TCompGroupRef** TCompGroupReference

289 TCompGroupReference ::= [(SuiteIdentifier | TTCN_ModuleIdentifier) “/”] {TCompGroupIdentifier “/”}

290 TCompGroupIdentifier ::= Identifier

291 TCompDcl ::= **$TCompDcl** TCompId C_Role NumOf_PCOs NumOf_CPs [Comment] **$End_TCompDcl**

292 TCompId ::= **$TCompId** TCompIdentifier

293 TCompIdentifier::= Identifier

294 C_Role ::= **$TCompRole** TCompRole

295 TCompRole **::= MTC** | **PTC**

296 NumOf_PCOs ::= **$NumOf_PCOs** Num_PCOs

297 Num_PCOs ::= Number

298 NumOf_CPs ::= **$NumOf_CPs** Num_CPs

299 Num_CPs ::= Number

A.3.3.12.11 Test Component Configuration Declarations

300 TCompConfigDcls ::= **$TCompConfigDcls** {TCompConfigDclOrGroup}+ **$End_TCompConfigDcls**

301 TCompConfigDclOrGroup ::= TCompConfigDcl | TCompConfigDclGroup

302 TCompConfigDclGroup ::= **$TCompConfigDclGroup** TCompConfigDclGroupId {TCompConfigDclOrGroup}+ **$End_TCompConfigDclGroup**

303 TCompConfigDclGroupId ::= **$TCompConfigDclGroupId** TCompConfigDclGroupIdentifier

304 TCompConfigDcl ::= **$Begin_TCompConfigDcl** TCompConfigId [TCompConfigGroupRef] [Comment] TCompConfigInfos [Comment] **$End_TCompConfigDcl**

305 TCompConfigId ::= **$TCompConfigId** TCompConfigIdentifier

306 TCompConfigIdentifier ::= Identifier

307 TCompConfigGroupRef ::= **$TCompConfigGroupRef** TCompConfigGroupReference

308 TCompConfigGroupReference ::= [(SuiteIdentifier | TTCN_ModuleIdentifier) “/”] {TCompConfigGroupIdentifier “/”}

309 TCompConfigGroupIdentifier ::= Identifier

310 TCompConfigInfos ::= **$TCompConfigInfos** {TCompConfigInfo}+ **$End_TCompConfigInfos**
/* STATIC SEMANTICS - Exactly one of the TCompConfigInfos shall be for a Test Components which has a TCompRole which is **MTC**. */

311 TCompConfigInfo ::= **$TCompConfigInfo** TCompUsed PCOs_Used CPs_Used [Comment] **$End_TCompConfigInfo**

312 TCompUsed ::= **$TCompUsed** TCompIdentifier

313 PCOs_Used ::= **$PCOs_Used** [PCO_List]

314 PCO_List ::= PCO_Identifier {Comma PCO_Identifier}
/* STATIC SEMANTICS - The number of PCOs in the PCO_List shall be the same as in the Test Component declaration. */
/* STATIC SEMANTICS - A given PCO_Identifier shall not be used more than once in the same Test Component Configuration. */

315 CPs_Used ::= **$CPs_Used** [CP_List]

316 CP_List ::= CP_Identifier {Comma CP_Identifier}
/* STATIC SEMANTICS - For a PTC, the number of CPs in the CP_List shall be the same as in the Test Component declaration. */
/* STATIC SEMANTICS - For an MTC, the number of CPs in the CP_List shall be no more than the number in the Test Component declaration. */
/* STATIC SEMANTICS - A given CP_Identifier shall not appear more than once in a given CP_List. */
/* STATIC SEMANTICS - Each CP_Identifier which is used in a Test Component Configuration shall appear in the CP_List of precisely two Test

Components in that Configuration. */

A.3.3.13 ASP, PDU and CM Type Definitions

A.3.3.13.1 General

317 ComplexDefinitions ::= [ASP_TypeDefs] [PDU_TypeDefs] [CM_TypeDefs] [AliasDefsOrGroup]
/* STATIC SEMANTICS - PDUs shall be optional */

A.3.3.13.2 ASP Type Definitions

318 ASP_TypeDefs ::= **$ASP_TypeDefs** [TTCN_ASP_TypeDefs] [ASN1_ASP_TypeDefs] [ASN1_ASP_TypeDefsByRefOrGroup] **$End_ASP_TypeDefs**

A.3.3.13.3 Tabular ASP Type Definitions

319 TTCN_ASP_TypeDefs ::= **$TTCN_ASP_TypeDefs** {TTCN_ASP_TypeDefOrGroup}+ **$End_TTCN_ASP_TypeDefs**

320 TTCN_ASP_TypeDefOrGroup ::= TTCN_ASP_TypeDef | TTCN_ASP_TypeDefGroup

321 TTCN_ASP_TypeDefGroup ::= **$TTCN_ASP_TypeDefGroup** TTCN_ASP_TypeDefGroupId {TTCN_ASP_TypeDefOrGroup}+ **$End_TTCN_ASP_TypeDefGroup**

322 TTCN_ASP_TypeDefGroupId ::= **$TTCN_ASP_TypeDefGroupId** ASP_GroupIdentifier

323 TTCN_ASP_TypeDef ::= **$Begin_TTCN_ASP_TypeDef** ASP_Id [ASP_GroupRef] PCO_Type [Comment] [ASP_ParDcls] [Comment] **$End_TTCN_ASP_TypeDef**

324 ASP_Id ::= **$ASP_Id** ASP_Id&FullId

325 ASP_Id&FullId ::= ASP_Identifier [FullIdentifier]

326 ASP_Identifier ::= Identifier
/* STATIC SEMANTICS - Identifier may be AliasIdentifier provided that it is being used in the behaviour column of a behaviour table (i.e. in a Behaviour Description). */

327 ASP_GroupRef ::= **$ASP_GroupRef** ASP_GroupReference

328 ASP_GroupReference ::= [(SuiteIdentifier | TTCN_ModuleIdentifier) "/"] {ASP_GroupIdentifier "/"}

329 ASP_GroupIdentifier ::= Identifier

330 PCO_Type ::= **$PCO_Type** [PCO_TypeIdentifier]
/* STATIC SEMANTICS - If there is no PCO_Type declaration table then, PCO_TypeIdentifier shall be one of the PCO types used in the PCO declaration table. */
/* STATIC SEMANTICS - If only a single PCO is defined within a test suite then PCO_TypeIdentifier is optional. */

331 ASP_ParDcls ::= **$ASP_ParDcls** {ASP_ParDcl} **$End_ASP_ParDcls**

332 ASP_ParDcl ::= **$ASP_ParDcl** ASP_ParId ASP_ParType [Comment] **$End_ASP_ParDcl**

333 ASP_ParId ::= **$ASP_ParId** ASP_ParIdOrMacro

334 ASP_ParIdOrMacro ::= ASP_ParId&FullId | MacroSymbol
/* STATIC SEMANTICS - The MacroSymbol shall be used only in combination with a reference to a Structured Type. */

335 ASP_ParId&FullId ::= ASP_ParIdentifier [FullIdentifier]

336 ASP_ParIdentifier ::= Identifier

337 ASP_ParType ::= **$ASP_ParType** Type&Attributes
/* STATIC SEMANTICS - Type shall be a PredefinedType or TS_TypeIdentifier, PDU_Identifier, or **PDU**. */

A.3.3.13.4 ASN.1 ASP Type Definitions

338 ASN1_ASP_TypeDefs ::= **$ASN1_ASP_TypeDefs** {ASN1_ASP_TypeDefOrGroup} **$End_ASN1_ASP_TypeDefs**

339 ASN1_ASP_TypeDefOrGroup ::= ASN1_ASP_TypeDef | ASN1_ASP_TypeDefGroup

340 ASN1_ASP_TypeDefGroup ::= **$ASN1_ASP_TypeDefGroup** ASN1_ASP_TypeDefGroupId {ASN1_ASP_TypeDefOrGroup}+ **$End_ASN1_ASP_TypeDefGroup**

341 ASN1_ASP_TypeDefGroupId ::= **$ASN1_ASP_TypeDefGroupId** ASN1ASP_GroupIdentifier

342 ASN1_ASP_TypeDef ::= **$Begin_ASN1_ASP_TypeDef** ASP_Id [ASN1ASP_GroupRef] PCO_Type [Comment] [ASN1_TypeDefinition] [Comment] **$End_ASN1_ASP_TypeDef**

343 ASN1ASP_GroupRef ::= **$ASN1ASP_GroupRef** ASN1ASP_GroupReference

344 ASN1ASP_GroupReference ::= [(SuiteIdentifier | TTCN_ModuleIdentifier) “/”] {ASN1ASP_GroupIdentifier “/”}

345 ASN1ASP_GroupIdentifier ::= Identifier

A.3.3.13.5 ASN.1 ASP Type Definitions by Reference

346 ASN1_ASP_TypeRefOrGroup ::= ASN1_ASP_TypeRef | ASN1_ASP_TypeRefGroup

347 ASN1_ASP_TypeRefGroup ::= **$ASN1_ASP_TypeRefGroup** ASN1_ASP_TypeRefGroupId {ASN1_ASP_TypeRefOrGroup}+ **$End_ASN1_ASP_TypeRefGroup**

348 ASN1_ASP_TypeRefGroupId ::= **$ASN1_ASP_TypeRefGroupId** ASN1ASP_GroupIdentifier

349 ASN1_ASP_TypeDefsByRef ::= **$Begin_ASN1_ASP_TypeDefsByRef** [ASN1ASP_RefGroupRef] {[CollComment] ASN1_ASP_TypeDefByRef}+ [Comment] **$End_ASN1_ASP_TypeDefsByRef**

/* NOTE - Collective comments may be used in this table according to Figure 2. */

350 ASN1ASP_RefGroupRef ::= **$ASN1ASP_RefGroupRef** ASN1ASP_GroupReference

351 ASN1_ASP_TypeDefByRef ::= **$ASN1_ASP_TypeDefByRef** ASP_Id PCO_Type ASN1_TypeReference ASN1_ModuleId [Comment] **$End_ASN1_ASP_TypeDefByRef**

/* STATIC SEMANTICS - ASP_Id shall not be specified with a FullIdentifier. */

A.3.3.13.6 PDU Type Definitions

352 PDU_TypeDefs ::= **$PDU_TypeDefs** [TTCN_PDU_TypeDefs] [ASN1_PDU_TypeDefs] [ASN1_PDU_TypeDefsByRefOrGroup] **$End_PDU_TypeDefs**

A.3.3.13.7 Tabular PDU Type Definitions

353 TTCN_PDU_TypeDefs ::= **$TTCN_PDU_TypeDefs** {TTCN_PDU_TypeDefOrGroup}+ **$End_TTCN_PDU_TypeDefs**

354 TTCN_PDU_TypeRefOrGroup ::= TTCN_PDU_TypeRef | TTCN_PDU_TypeRefGroup

355 TTCN_PDU_TypeRefGroup ::= **$TTCN_PDU_TypeRefGroup** TTCN_PDU_TypeRefGroupId {TTCN_PDU_TypeRefOrGroup}+ **$End_TTCN_PDU_TypeRefGroup**

356 TTCN_PDU_TypeRefGroupId ::= **$TTCN_PDU_TypeRefGroupId** PDU_GroupIdentifier

357 TTCN_PDU_TypeDef ::= **$Begin_TTCN_PDU_TypeDef** PDU_Id [PDU_GroupRef] PCO_Type [PDU_EncodingId] [EncVariationId] [Comment] [PDU_FieldDcls] [Comment] **$End_TTCN_PDU_TypeDef**

/* STATIC SEMANTICS - If a PDU is sent or received only embedded in ASPs within the whole test suite, then PCO_TypeIdentifier (in PCO_Type) is optional. */

358 PDU_Id ::= **$PDU_Id** PDU_Id&FullId

359 PDU_Id&FullId ::= PDU_Identifier [FullIdentifier]

360 PDU_Identifier ::= Identifier

/* STATIC SEMANTICS - Identifier may be AliasIdentifier provided that it is being used in the behaviour column of a behaviour table (i.e. in a Behaviour Description). */

361 PDU_GroupRef ::= **$PDU_GroupRef** PDU_GroupReference

362 PDU_GroupReference ::= [(SuiteIdentifier | TTCN_ModuleIdentifier) “/”] {PDU_GroupIdentifier “/”}

363 PDU_GroupIdentifier ::= Identifier

364 PDU_EncodingId ::= **$PDU_EncodingId** [EncodingRuleIdentifier]

365 PDU_FieldDcls ::= **$PDU_FieldDcls** {PDU_FieldDcl} **$End_PDU_FieldDcls**

366 PDU_FieldDcl ::= **$PDU_FieldDcl** PDU_FieldId PDU_FieldType [PDU_FieldEncoding] [Comment] **$End_PDU_FieldDcl**

367 PDU_FieldId ::= **$PDU_FieldId** PDU_FieldIdOrMacro

368 PDU_FieldIdOrMacro ::= PDU_FieldId&FullId | MacroSymbol

/* STATIC SEMANTICS - The MacroSymbol shall be used only in combination with a reference to a Structured Type. */

369 MacroSymbol ::= "<-"

370 PDU_FieldId&FullId ::= PDU_FieldIdentifier [FullIdentifier]

371 PDU_FieldIdentifier ::= Identifier

372 PDU_FieldType ::= **$PDU_FieldType** Type&Attributes

/* STATIC SEMANTICS - Type shall be a PredefinedType or TS_TypeIdentifier, PDU_Identifier, or **PDU**. */

373 Type&Attributes ::= (Type [LengthAttribute]) | **PDU**

/* OPERATIONAL SEMANTICS - The set of values defined by LengthAttribute shall be a true subset of the values of the base type. */

/* STATIC SEMANTICS - LengthAttribute shall be provided only when the base type is a string type (i.e., BITSTRING, HEXSTRING, OCTETSTRING or CharacterString) or derived from a string type. */

374 LengthAttribute ::= SingleLength | RangeLength

375 SingleLength ::= "[" Bound "]"

376 Bound ::= Number | TS_ParIdentifier | TS_ConstIdentifier

/* OPERATIONAL SEMANTICS - Bound shall evaluate to a non-negative INTEGER value or INFINITY. */

377 RangeLength ::= "[" LowerBound To UpperBound "]"

/* OPERATIONAL SEMANTICS - LowerBound shall be less than UpperBound. */

378 LowerBound ::= Bound

379 UpperBound ::= Bound | **INFINITY**

A.3.3.13.8 ASN.1 PDU Type Definitions

380 ASN1_PDU_TypeDefs ::= **$ASN1_PDU_TypeDefs** {ASN1_PDU_TypeDefOrGroup} **$End_ASN1_PDU_TypeDefs**

381 ASN1_PDU_TypeDefOrGroup ::= ASN1_PDU_TypeDef | ASN1_PDU_TypeDefGroup

382 ASN1_PDU_TypeDefGroup ::= **$ASN1_PDU_TypeDefGroup** ASN1_PDU_TypeDefGroupId {ASN1_PDU_TypeDefOrGroup}+ **$End_ASN1_PDU_TypeDefGroup**

383 ASN1_PDU_TypeDefGroupId ::= **$ASN1_PDU_TypeDefGroupId** ASN1PDU_GroupIdentifier

384 ASN1_PDU_TypeDef ::= **$Begin_ASN1_PDU_TypeDef** PDU_Id [ASN1PDU_GroupRef] PCO_Type [PDU_EncodingId] [EncVariationId] [Comment] [ASN1_TypeDefinition] [Comment] **$End_ASN1_PDU_TypeDef**

/* STATIC SEMANTICS - If a PDU is sent or received only embedded in ASPs within the whole test suite, then PCO_TypeIdentifier (in PCO_Type) is optional. */

385 ASN1PDU_GroupRef ::= **$ASN1PDU_GroupRef** ASN1PDU_GroupReference

386 ASN1PDU_GroupReference ::= [(SuiteIdentifier | TTCN_ModuleIdentifier) “/”] {ASN1PDU_GroupIdentifier “/”}

387 ASN1PDU_GroupIdentifier ::= Identifier

A.3.3.13.9 ASN.1 PDU Type Definitions by Reference

388 ASN1_PDU_TypeRefOrGroup ::= ASN1_PDU_TypeRef | ASN1_PDU_TypeRefGroup

389 ASN1_PDU_TypeRefGroup ::= **$ASN1_PDU_TypeRefGroup** ASN1_PDU_TypeRefGroupId {ASN1_PDU_TypeRefOrGroup}+ **$End_ASN1_PDU_TypeRefGroup**

390 ASN1_PDU_TypeRefGroupId ::= **$ASN1_PDU_TypeRefGroupId** ASN1PDU_GroupIdentifier

391 ASN1_PDU_TypeDefsByRef ::= **$Begin_ASN1_PDU_TypeDefsByRef** [ASN1PDU_RefGroupRef] {[CollComment] ASN1_PDU_TypeDefByRef}+ [Comment] **$End_ASN1_PDU_TypeDefsByRef**

/* NOTE - Collective comments may be used in this table according to Figure 2. */

392 ASN1PDU_RefGroupRef ::= **$ASN1PDU_RefGroupRef** ASN1PDU_GroupReference

393 ASN1_PDU_TypeDefByRef ::= **$ASN1_PDU_TypeDefByRef** PDU_Id PCO_Type ASN1_TypeReference ASN1_ModuleId [PDU_EncodingId] [EncVariationId] [Comment] **$End_ASN1_PDU_TypeDefByRef**

/* STATIC SEMANTICS - If a PDU is sent or received only embedded in ASPs within the whole test suite, then PCO_TypeIdentifier (in PCO_Type) is optional. */

/* STATIC SEMANTICS - PDU_Id shall not be specified with a FullIdentifier. */

A.3.3.13.10 CM Type Definitions

394 CM_TypeDefs ::= **$CM_TypeDefs** [TTCN_CM_TypeDefs] [ASN1_CM_TypeDefs] **$End_CM_TypeDefs**

A.3.3.13.11 Tabular CM Type Definition

395 TTCN_CM_TypeDefs ::= **$TTCN_CM_TypeDefs** {TTCN_CM_TypeDefOrGroup}+ **$End_TTCN_CM_TypeDefs**

396 TTCN_CM_TypeDefOrGroup ::= TTCN_CM_TypeDef | TTCN_CM_TypeDefGroup

397 TTCN_CM_TypeDefGroup ::= **$TTCN_CM_TypeDefGroup** TTCN_CM_TypeDefGroupId {TTCN_CM_TypeDefOrGroup}+ **$End_TTCN_CM_TypeDefGroup**

398 TTCN_CM_TypeDefGroupId ::= **$TTCN_CM_TypeDefGroupId** CM_GroupIdentifier

399 TTCN_CM_TypeDef ::= **$Begin_TTCN_CM_TypeDef** CM_Id [CM_GroupRef] [Comment] [CM_ParDcls] [Comment] **$End_TTCN_CM_TypeDef**

400 CM_Id ::= **$CM_Id** CM_Identifier

401 CM_Identifier ::= Identifier

402 CM_GroupRef ::= **$CM_GroupRef** CM_GroupReference

403 CM_GroupReference ::= [(SuiteIdentifier | TTCN_ModuleIdentifier) "/"] {CM_GroupIdentifier "/"}

404 CM_GroupIdentifier ::= Identifier

405 CM_ParDcls ::= **$CM_ParDcls** {CM_ParDcl} **$End_CM_ParDcls**

406 CM_ParDcl ::= **$CM_ParDcl** CM_ParId CM_ParType [Comment] **$End_CM_ParDcl**

407 CM_ParId ::= **$CM_ParId** CM_ParIdOrMacro

408 CM_ParIdOrMacro ::= CM_ParIdentifier | MacroSymbol

/* STATIC SEMANTICS - The MacroSymbol shall be used only in combination with a reference to a Structured Type. */

409 CM_ParIdentifier ::= Identifier

410 CM_ParType ::= **$CM_ParType** Type&Attributes

A.3.3.13.12 ASN.1 CM Type Definitions

411 ASN1_CM_TypeDefs ::= **$ASN1_CM_TypeDefs** {ASN1_CM_TypeDefOrGroup}+ **$End_ASN1_CM_TypeDefs**

412 ASN1_CM_TypeDefOrGroup ::= ASN1_CM_TypeDef | ASN1_CM_TypeDefGroup

413 ASN1_CM_TypeDefGroup ::= **$ASN1_CM_TypeDefGroup** ASN1_CM_TypeDefGroupId {ASN1_CM_TypeDefOrGroup}+ **$End_ASN1_CM_TypeDefGroup**

414 ASN1_CM_TypeDefGroupId ::= **$ASN1_CM_TypeDefGroupId** ASN1CM_GroupIdentifier

415 ASN1_CM_TypeDef ::= **$Begin_ASN1_CM_TypeDef** CM_Id [ASN1CM_GroupRef] [Comment] [ASN1_TypeDefinition] [Comment] **$End_ASN1_CM_TypeDef**

416 ASN1CM_GroupRef ::= **$ASN1CM_GroupRef** ASN1CM_GroupReference

417 ASN1CM_GroupReference ::= [(SuiteIdentifier | TTCN_ModuleIdentifier) "/"] {ASN1CM_GroupIdentifier "/"}

418 ASN1CM_GroupIdentifier ::= Identifier

A.3.3.13.13 Varieties of Encoding Definition

419 EncodingDefs ::= **$EncodingDefs** [EncodingDefinitionsOrGroup] [EncodingVariations] [InvalidFieldEncodingDefs] **$End_EncodingDefs**

A.3.3.13.13.1 Encoding Definitions

420 EncodingDefinitionsOrGroup ::= EncodingDefinitions | EncodingDefinitionsGroup

421 EncodingDefinitionsGroup ::= **$EncodingDefinitionsGroup** EncodingDefinitionsGroupId {EncodingDefinitionsOrGroup}+ **$End_EncodingDefinitionsGroup**

422 EncodingDefinitionsGroupId ::= **$EncodingDefinitionsGroupId** EncodingGroupIdentifier

423 EncodingDefinitions ::= **$Begin_EncodingDefinitions** [EncodingGroupRef] {[CollComment] EncodingDefinition}+ [Comment] **$End_EncodingDefinitions**

/* NOTE - Collective comments may be used in this table according to Figure 2. */

424 EncodingGroupRef ::= **$EncodingGroupRef** EncodingGroupReference

425 EncodingGroupReference ::= [(SuiteIdentifier | TTCN_ModuleIdentifier) "/"] {EncodingGroupIdentifier "/"}

426 EncodingGroupIdentifier ::= Identifier

427 EncodingDefinition ::= **$EncodingDefinition** EncodingRuleId EncodingRef EncodingDefault [Comment] **$End_EncodingDefinition**

/* OPERATIONAL SEMANTICS - No more than one EncodingRuleIdentifier shall have an EncodingDefault containing a DefaultExpression which evaluates to TRUE*/

428 EncodingRuleId ::= **$EncodingRuleId** EncodingRuleIdentifier

429 EncodingRuleIdentifier ::= Identifier

430 EncodingRef ::= **$EncodingRef** EncodingReference

431 EncodingReference ::= BoundedFreeText

432 EncodingDefault ::= **$EncodingDefault** [DefaultExpression]

433 DefaultExpression ::= Expression

/* STATIC SEMANTICS - DefaultExpression shall only contain LiteralValues, TS_ParIdentifiers and TS_ConstIdentifiers. */

A.3.3.13.13.2 Encoding Variations

434 EncodingVariations ::= **$EncodingVariations** {EncodingVariationSetOrGroup}+ **$End_EncodingVariations**

435 EncodingVariationSetOrGroup ::= EncodingVariationSet | EncodingVariationSetGroup

436 EncodingVariationSetGroup ::= **$EncodingVariationSetGroup** EncodingVariationSetGroupId {EncodingVariationSetOrGroup}+ **$End_EncodingVariationSetGroup**

437 EncodingVariationSetGroupId ::= **$EncodingVariationSetGroupId** EncVariationGroupIdentifier

438 EncodingVariationSet ::= **$Begin_EncodingVariationSet** EncodingRuleId [EncVariationGroupRef] Encoding_TypeList [Comment] EncodingVariationList [Comment] **$End_EncodingVariationSet**

439 EncVariationGroupRef ::= **$EncVariationGroupRef** EncVariationGroupReference

440 EncVariationGroupReference ::= [(SuiteIdentifier | TTCN_ModuleIdentifier) "/"] {EncVariationGroupIdentifier "/"}

441 EncVariationGroupIdentifier ::= Identifier

442 EncodingVariationList ::= **$EncodingVariationList** {EncodingVariation}+ **$End_EncodingVariationList**

443 Encoding_TypeList ::= **$Encoding_TypeList** [TypeList]

444 TypeList ::=Type {Comma Type}

/* STATIC SEMANTICS - Type shall not be an ASP_Identifier, PDU_Identifier or StructIdentifier, since such types may be encoded by encoding rules but not by field encodings. */

445 EncodingVariation ::= **$EncodingVariation** EncodingVariationId VariationRef VariationDefault [Comment] **$End_EncodingVariation**

/* OPERATIONAL SEMANTICS - No more than one EncodingIdentifier shall have a VariationDefault containing a DefaultExpression which evaluates to TRUE. */

446 EncodingVariationId ::= **$EncodingVariationId** EncVariationId&ParList

447 EncVariationId&ParList ::= EncVariationIdentifier [FormalParList]

448 EncVariationIdentifier ::= Identifier

449 VariationRef ::= **$VariationRef** VariationReference

450 VariationReference ::= BoundedFreeText

451 VariationDefault ::= **$VariationDefault** [DefaultExpression]

A.3.3.13.13.3 Invalid Encoding Definitions

452 InvalidFieldEncodingDefs ::= **$InvalidFieldEncodingDefs** {InvalidFieldEncodingDefOrGroup}+ **$End_InvalidFieldEncodingDefs**

453 InvalidFieldEncodingOrGroup ::= InvalidFieldEncoding | InvalidFieldEncodingGroup

454 InvalidFieldEncodingGroup ::= **$InvalidFieldEncodingGroup** InvalidFieldEncodingGroupId {InvalidFieldEncodingOrGroup}+ **$End_InvalidFieldEncodingGroup**

455 InvalidFieldEncodingGroupId ::= **$InvalidFieldEncodingGroupId** InvalidFieldEncodingGroupIdentifier

456 InvalidFieldEncodingDef ::= **$Begin_InvalidFieldEncodingDef** InvalidFieldEncodingId [InvalidFieldEncodingGroupRef] Encoding_TypeList [Comment] InvalidFieldEncodingDefinition [Comment] **$End_InvalidFieldEncodingDef**

457 InvalidFieldEncodingId ::= **$InvalidFieldEncodingId** InvalidFieldEncodingId&ParList

458 InvalidFieldEncodingId&ParList ::= InvalidFieldEncodingIdentifier [FormalParList]

459 InvalidFieldEncodingIdentifier ::= Identifier

460 InvalidFieldEncodingGroupRef ::= **$InvalidFieldEncodingGroupRef** InvalidFieldEncodingGroupReference

461 InvalidFieldEncodingGroupReference ::= [(SuiteIdentifier | TTCN_ModuleIdentifier) "/"] {InvalidFieldEncodingGroupIdentifier "/"}

462 InvalidFieldEncodingGroupIdentifier ::= Identifier

463 InvalidFieldEncodingDefinition ::= **$InvalidFieldEncodingDefinition** TS_OpProcDef **$End_InvalidFieldEncodingDefinition**

/* OPERATIONAL SEMANTICS - TS_OpProcDef shall produce a BitString result, to be interpreted as the encoding to be transmitted high order bit first. */

A.3.3.13.14 Alias Definitions

464 AliasDefsOrGroup ::= AliasDefs | AliasDefsGroup

465 AliasDefsGroup ::= **$AliasDefsGroup** AliasDefsGroupId {AliasDefsOrGroup}+ **$End_AliasDefsGroup**

466 AliasDefsGroupId ::= **$AliasDefsGroupId** AliasDefsGroupIdentifier

467 AliasDefs ::= **$Begin_AliasDefs** [AliasGroupRef] {[CollComment] AliasDef}+ [Comment] **$End_AliasDefs**

/* NOTE - Collective comments may be used in this table according to Figure 2. */

468 AliasGroupRef ::= **$AliasGroupRef** AliasGroupReference

469 AliasGroupReference ::= [(SuiteIdentifier | TTCN_ModuleIdentifier) "/"] {AliasGroupIdentifier "/"}

470 AliasGroupIdentifier ::= Identifier

471 AliasDef ::= **$AliasDef** AliasId ExpandedId [Comment] **$End_AliasDef**

472 AliasId ::= **$AliasId** AliasIdentifier

473 AliasIdentifier ::= Identifier

/* STATIC SEMANTICS - An AliasIdentifier shall be used only in a statement line of a behaviour description. */

/* STATIC SEMANTICS - An AliasIdentifier shall be used only where an ASP_Identifier or PDU_Identifier is valid. */

474 ExpandedId ::= **$ExpandedId** Expansion

475 Expansion ::= ASP_Identifier | PDU_Identifier

A.3.3.14 The Constraints Part

476 ConstraintsPart ::= **$ConstraintsPart** [TS_TypeConstraints] [ASP_Constraints] [PDU_Constraints] [CM_Constraints] **$End_ConstraintsPart**

A.3.3.15 Test Suite Type Constraint Declarations

477 TS_TypeConstraints ::= **$TS_TypeConstraints** [StructTypeConstraints] [ASN1_TypeConstraints] **$End_TS_TypeConstraints**

A.3.3.16 Structured Type Constraint Declarations

478 StructTypeConstraints ::= **$StructTypeConstraints** {StructTypeConstraintOrGroup}+ **$End_StructTypeConstraints**

479 StructTypeConstraintOrGroup ::= StructTypeConstraint | StructTypeConstraintGroup

480 StructTypeConstraintGroup ::= **$StructTypeConstraintGroup** StructTypeConstraintGroupId {StructTypeConstraintOrGroup}+ **$End_StructTypeConstraintGroup**

481 StructTypeConstraintGroupId ::= **$StructTypeConstraintGroupId** StructTypeConstraintGroupIdentifier

482 StructTypeConstraint ::= **$Begin_StructTypeConstraint** ConsId [StructTypeConstraintGroupRef] StructId DerivPath [EncVariationId] [Comment] ElemValues [Comment] **$End_StructTypeConstraint**

/* STATIC SEMANTICS - The FullIdentifier that is part of Struct_Id shall not be used. */

/* STATIC SEMANTICS - A modified constraint shall have the same parameter list as its base constraint. In particular, there shall be no parameters omitted from or added to this list. */

483 StructTypeConstraintGroupRef ::= **$StructTypeConstraintGroupRef** StructTypeConstraintGroupReference

484 StructTypeConstraintGroupReference ::= [(SuiteIdentifier | TTCN_ModuleIdentifier) "/"] {StructTypeConstraintGroupIdentifier "/"}

485 StructTypeConstraintGroupIdentifier ::= Identifier

486 EncVariationId ::= **$EncVariationId** [EncVariationCall]

487 EncVariationCall ::= EncVariationIdentifier [ActualParList]

488 ElemValues ::= **$ElemValues** {ElemValue}+ **$End_ElemValues**

489 ElemValue ::= **$ElemValue** ElemId ConsValue [PDU_FieldEncoding] [Comment] **$End_ElemValue**

/* STATIC SEMANTICS - The FullIdentifier that is part of ElemId shall not be used. */

/* STATIC SEMANTICS - Parameterized Element values in a base constraint shall not be modified or explicitly omitted in a modified constraint. */

490 PDU_FieldEncoding ::= **$PDU_FieldEncoding** [PDU_FieldEncodingCall]

491 PDU_FieldEncodingCall ::= EncVariationCall | InvalidFieldEncodingCall

492 InvalidFieldEncodingCall ::= InvalidFieldEncodingIdentifier (ActualParList | "(" ")")

A.3.3.17 ASN.1 Type Constraint Declarations

493 ASN1_TypeConstraints ::= **$ASN1_TypeConstraints** {ASN1_TypeConstraintOrGroup}+ **$End_ASN1_TypeConstraints**

494 ASN1_TypeConstraintOrGroup ::= ASN1_TypeConstraint | ASN1_TypeConstraintGroup

495 ASN1_TypeConstraintGroup ::= **$ASN1_TypeConstraintGroup** ASN1_TypeConstraintGroupId {ASN1_TypeConstraintOrGroup}+ **$End_ASN1_TypeConstraintGroup**

496 ASN1_TypeConstraintGroupId ::= **$ASN1_TypeConstraintGroupId** ASN1_TypeConstraintGroupIdentifier

497 ASN1_TypeConstraint ::= **$Begin_ASN1_TypeConstraint** ConsId [ASN1_TypeConstraintGroupRef] ASN1_TypeId DerivPath [EncVariationId] [Comment] ASN1_ConsValue [Comment] **$End_ASN1_TypeConstraint**

/* STATIC SEMANTICS - The FullIdentifier that is part of ASN1_TypeId shall not be used. */

/* STATIC SEMANTICS - A modified constraint shall have the same parameter list as its base constraint. In particular, there shall be no parameters omitted from or added to this list. */

498 ASN1_TypeConstraintGroupRef ::= **$ASN1_TypeConstraintGroupRef** ASN1_TypeConstraintGroupReference

499 ASN1_TypeConstraintGroupReference ::= [(SuiteIdentifier | TTCN_ModuleIdentifier) “/”] {ASN1_TypeConstraintGroupIdentifier “/”}

500 ASN1_TypeConstraintGroupIdentifier ::= Identifier

A.3.3.18 ASP Constraint Declarations

501 ASP_Constraints ::= **$ASP_Constraints** [TTCN_ASP_Constraints] [ASN1_ASP_Constraints] **$End_ASP_Constraints**

A.3.3.19 Tabular ASP Constraint Declarations

502 TTCN_ASP_Constraints ::= **$TTCN_ASP_Constraints** {TTCN_ASP_ConstraintOrGroup}+ **$End_TTCN_ASP_Constraints**

503 TTCN_ASP_ConstraintOrGroup ::= TTCN_ASP_Constraint | TTCN_ASP_ConstraintGroup

504 TTCN_ASP_ConstraintGroup ::= **$TTCN_ASP_ConstraintGroup** TTCN_ASP_ConstraintGroupId {TTCN_ASP_ConstraintOrGroup}+ **$End_TTCN_ASP_ConstraintGroup**

505 TTCN_ASP_ConstraintGroupId ::= **$TTCN_ASP_ConstraintGroupId** ASP_ConstraintGroupIdentifier

506 TTCN_ASP_Constraint ::= **$Begin_TTCN_ASP_Constraint** ConsId [ASP_ConstraintGroupRef] ASP_Id DerivPath [Comment] [ASP_ParValues] [Comment] **$End_TTCN_ASP_Constraint**

/* STATIC SEMANTICS - The FullIdentifier that is part of ASP_Id shall not be used. */

/* STATIC SEMANTICS - If an ASP is substructured, then the constraints for ASPs of that type shall have the same structure*/

/* STATIC SEMANTICS - A modified constraint shall have the same parameter list as its base constraint. In particular, there shall be no parameters omitted from or added to this list. */

507 ASP_ConstraintGroupRef ::= **$ASP_ConstraintGroupRef** ASP_ConstraintGroupReference

508 ASP_ConstraintGroupReference ::= [(SuiteIdentifier | TTCN_ModuleIdentifier) “/”] {ASP_ConstraintGroupIdentifier “/”}

509 ASP_ConstraintGroupIdentifier ::= Identifier

510 ASP_ParValues ::= **$ASP_ParValues** {ASP_ParValue}+ **$End_ASP_ParValues**

511 ASP_ParValue ::= **$ASP_ParValue** ASP_ParId ConsValue [Comment] **$End_ASP_ParValue**

/* STATIC SEMANTICS - The FullIdentifier that is part of ASP_ParId shall not be used. */

/* STATIC SEMANTICS - If an ASP definition refers to a Structured Type as a substructure of a parameter (*i.e.*, with a parameter name) then the corresponding constraint shall have the same parameter name in the corresponding position in the parameter name name column of the constraint and the value shall be a reference to a constraint for that parameter (*i.e.*, for that substructure in accordance with the definition of the Structured Type). */

/* STATIC SEMANTICS - If an ASP definition refers to a parameter specified as being of metatype PDU then in a corresponding constraint, the value for that parameter shall be specified as the name of a PDU constraint, or formal parameter. */

/* STATIC SEMANTICS - Use of structured constraints by macro expansion in a constraint shall not be used unless the corresponding ASP definition also references the same Structured Type by macro expansion. */

/* STATIC SEMANTICS - Parameterized ASP parameter values in a base constraint shall not be modified or explicitly omitted in a modified constraint. */

A.3.3.20 ASN.1 ASP Constraint Declarations

512 ASN1_ASP_Constraints ::= **$ASN1_ASP_Constraints** {ASN1_ASP_ConstraintOrGroup}+ **$End_ASN1_ASP_Constraints**

513 ASN1_ASP_ConstraintOrGroup ::= ASN1_ASP_Constraint | ASN1_ASP_ConstraintGroup

514 ASN1_ASP_ConstraintGroup ::= **$ASN1_ASP_ConstraintGroup** ASN1_ASP_ConstraintGroupId {ASN1_ASP_ConstraintOrGroup}+ **$End_ASN1_ASP_ConstraintGroup**

515 ASN1_ASP_ConstraintGroupId ::= **$ASN1_ASP_ConstraintGroupId** ASN1ASP_ConstraintGroupIdentifier

516 ASN1_ASP_Constraint ::= **$Begin_ASN1_ASP_Constraint** ConsId [ASN1ASP_ConstraintGroupRef] ASP_Id DerivPath [Comment] [ASN1_ConsValue] [Comment] **$End_ASN1_ASP_Constraint**

/* STATIC SEMANTICS - The FullIdentifier that is part of ASP_Id shall not be used. */

/* STATIC SEMANTICS - If an ASP is substructured, then the constraints for ASPs of that type shall have a compatible ASN.1 structure (*i.e.*, possibly with some groupings). */

/* STATIC SEMANTICS - A modified constraint shall have the same parameter list as its base constraint. In particular, there shall be no parameters omitted from or added to this list. */

517 ASN1ASP_ConstraintGroupRef ::= **$ASN1ASP_ConstraintGroupRef** ASN1ASP_ConstraintGroupReference

518 ASN1ASP_ConstraintGroupReference ::= [(SuiteIdentifier | TTCN_ModuleIdentifier) “/”] {ASN1ASP_ConstraintGroupIdentifier “/”}

519 ASN1ASP_ConstraintGroupIdentifier ::= Identifier

A.3.3.21 PDU Constraint Declarations

520 PDU_Constraints ::= **$PDU_Constraints** [TTCN_PDU_Constraints] [ASN1_PDU_Constraints] **$End_PDU_Constraints**

A.3.3.22 Tabular PDU Constraint Declarations

521 TTCN_PDU_Constraints ::= **$TTCN_PDU_Constraints** {TTCN_PDU_ConstraintOrGroup}+ **$End_TTCN_PDU_Constraints**

522 TTCN_PDU_ConstraintOrGroup ::= TTCN_PDU_Constraint | TTCN_PDU_ConstraintGroup

523 TTCN_PDU_ConstraintGroup ::= **$TTCN_PDU_ConstraintGroup** TTCN_PDU_ConstraintGroupId {TTCN_PDU_ConstraintOrGroup}+ **$End_TTCN_PDU_ConstraintGroup**

524 TTCN_PDU_ConstraintGroupId ::= **$TTCN_PDU_ConstraintGroupId** PDU_ConstraintGroupIdentifier

525 TTCN_PDU_Constraint ::= **$Begin_TTCN_PDU_Constraint** ConsId [PDU_ConstraintGroupRef] PDU_Id DerivPath [EncRuleId] [EncVariationId] [Comment] [PDU_FieldValues] [Comment] **$End_TTCN_PDU_Constraint**

/* STATIC SEMANTICS - The FullIdentifier that is part of PDU_Id shall not be used. */

/* STATIC SEMANTICS - If a PDU is substructured, then the constraints for PDUs of that type shall have the same structure*/

/* STATIC SEMANTICS - A modified constraint shall have the same parameter list as its base constraint. In particular, there shall be no parameters omitted from or added to this list. */

526 PDU_ConstraintGroupRef ::= **$PDU_ConstraintGroupRef** PDU_ConstraintGroupReference

527 PDU_ConstraintGroupReference ::= [(SuiteIdentifier | TTCN_ModuleIdentifier) “/”] {PDU_ConstraintGroupIdentifier “/”}

528 PDU_ConstraintGroupIdentifier ::= Identifier

529 EncRuleId ::= **$EncRuleId** [EncodingRuleIdentifier]

530 ConsId ::= **$ConsId** ConsId&ParList

531 ConsId&ParList ::= ConstraintIdentifier [FormalParList]

532 ConstraintIdentifier ::= Identifier

533 DerivPath ::= **$DerivPath** [DerivationPath]

534 DerivationPath ::= {ConstraintIdentifier Dot}+

/* STATIC SEMANTICS - If a constraint definition is a modification of an existing constraint, the name of the constraint that is taken as the basis of this modification shall be referenced in the table in the derivation path entry. */

/* STATIC SEMANTICS - The first ConstraintIdentifier in DerivationPath shall be a base constraint identifier. */

/* STATIC SEMANTICS - The DerivationPath shall be the complete list of constraints in the order in which their modifications to the base constraint are to be applied. */

/* STATIC SEMANTICS - There shall be no white space between ConstraintIdentifier and Dot. */

535 PDU_FieldValues ::= **$PDU_FieldValues** {PDU_FieldValue}+ **$End_PDU_FieldValues**

536 PDU_FieldValue ::= **$PDU_FieldValue** PDU_FieldId ConsValue [PDU_FieldEncoding] [Comment] **$End_PDU_FieldValue**

/* STATIC SEMANTICS - The FullIdentifier that is part of PDU_FieldId shall not be used. */

/* STATIC SEMANTICS - If a PDU definition refers to a Structured Type as a substructure of a field (*i.e.*, with a field name) then the corresponding constraint shall have the same field name in the corresponding position in the field name name column of the constraint and the value shall be a reference to a constraint for that field (*i.e.*, for that substructure in accordance with the definition of the Structured Type). */

/* STATIC SEMANTICS - If a PDU definition refers to a field specified as being of metatype PDU then in a corresponding constraint, the value for that field shall be specified as the name of a PDU constraint, or formal parameter. */

/* STATIC SEMANTICS - Use of structured constraints by macro expansion in a constraint shall not be used unless the corresponding PDU definition also references the same Structured Type by macro expansion. */

/* STATIC SEMANTICS - Parameterized PDU field values in a base constraint shall not be modified or explicitly omitted in a modified constraint. */

537 ConsValue ::= **$ConsValue** ConstraintValue&Attributes

/* OPERATIONAL SEMANTICS - ConsValue shall evaluate to an element of the type specified for the ASP parameter, PDU field or structure element. This may include matching symbols compatible with the specified type. */

538 ConstraintValue&Attributes ::= ConstraintValue ValueAttributes

/* NOTE - ConstraintValue&Attributes can be reached via DefinedValue in the ASN.1 syntax. See the reference on the production 715 for Value. */

/* STATIC SEMANTICS - ConstraintValue shall fulfil all restrictions defined for the ASP parameter, PDU field or structure element type, including value ranges, value lists, alphabet restrictions and/or length restrictions and shall fulfil the restrictions defined by ValueAttributes. */

/* OPERATIONAL SEMANTICS - Any length specifications defined for the ASP parameter or PDU field type in the Test Suite Type declarations shall not conflict with the length specifications in the ASP or PDU type definition. */

/* STATIC SEMANTICS - Neither Test Suite Variables nor Test Case Variables shall be used in constraints, unless passed as actual parameters. In the latter case they shall be bound to a value and shall not be changed. */

539 ConstraintValue ::= ConstraintExpression | Ma tchingSymbol | ConsRef

/* STATIC SEMANTICS - When a ConstraintExpression is used in a Constraint, its terms shall not contain TS_VarIdentifier or TC_VarIdentifier. */

540 ConstraintExpression ::= Expression

/* OPERATIONAL SEMANTICS - ConstraintExpression shall evaluate to an element of the specified type. */

541 MatchingSymbol ::= Complement | Omit | AnyValue | AnyOrOmit | ValueList | ValueRange | SuperSet | SubSet | Permutation

/* NOTE - No matching symbol is considered to be a specific value. */

542 Complement ::= **COMPLEMENT** ValueList

543 Omit ::= Dash | **OMIT**

/* STATIC SEMANTICS - In ASN.1 constraints Omit shall be used only for ASP parameters or PDU fields that are declared OPTIONAL or DEFAULT. */

544 AnyValue ::= "?"

545 AnyOrOmit ::= "*"

546 ValueList ::= "(" ConstraintValue&Attributes {Comma ConstraintValue&Attributes} ")"

/* STATIC SEMANTICS - Each ConstraintValue&Attributes shall be of the type declared for the ASP parameter, PDU field, or structure element in which the ValueList is used. */

547 ValueRange ::= "(" ValRange ")"

/* STATIC SEMANTICS - ValueRange shall be used only on ASP parameter, PDU field, or structure element of type INTEGER. */

/* STATIC SEMANTICS - The set of values defined by ValueRange shall be a true subset of the values allowed by the ASP parameter's, PDU field's or structure element's declared type. */

548 ValRange **::=** (LowerRangeBound To UpperRangeBound)

/* OPERATIONAL SEMANTICS - LowerRangeBound shall be less than UpperRangeBound. */

549 LowerRangeBound ::= ConstraintExpression | Minus **INFINITY**

/* OPERATIONAL SEMANTICS - ConstraintExpression shall evaluate to a specific INTEGER value. */

550 UpperRangeBound ::= ConstraintExpression | **INFINITY**

/* OPERATIONAL SEMANTICS - ConstraintExpression shall evaluate to a specific INTEGER value. */

551 SuperSet ::= **SUPERSET** "(" ConstraintValue&Attributes ")"

/* STATIC SEMANTICS - The argument to SuperSet, *i.e.*, ConstraintValue&Attributes, shall be of type SET OF. */

552 SubSet ::= **SUBSET** "(" ConstraintValue&Attributes ")"

/* STATIC SEMANTICS - The argument to SubSet, *i.e.*, ConstraintValue&Attributes, shall be of type SET OF. */

553 Permutation ::= **PERMUTATION** ValueList

/* STATIC SEMANTICS - The Permutation shall be used only inside a value of type SEQUENCE OF. */

/* STATIC SEMANTICS - The ValueList shall be of the type specified in the SEQUENCE OF. */

554 ValueAttributes ::= [ValueLength] [**IF_PRESENT**] [ASN1_Encoding]

/* STATIC SEMANTICS - In ASN.1 constraints IF_PRESENT shall be used only for ASP parameters or PDU fields that are declared OPTIONAL or DEFAULT. */

/* STATIC SEMANTICS - ASN1_Encoding shall only be used for ValueAttributes in ASN.1 Type Constraints and ASN.1 PDU Constraints. */

555 ASN1_Encoding ::= **ENC** PDU_FieldEncodingCall

556 ValueLength ::= SingleValueLength | RangeValueLength

/* STATIC SEMANTICS - ValueLength shall be used only for ASP parameters, PDU fields or structure element that are declared as BITSTRING, HEXSTRING, OCTETSTRING, CharacterString, SEQUENCE OF or SET OF. */

/* STATIC SEMANTICS - ValueLength shall be used only in combination with the following mechanisms: Specificvalue, Complement, Omit, AnyValue, AnyOrOmit, AnyOrNone and Permutation. */

/* STATIC SEMANTICS - The set of values defined by ValueLength shall be a true subset of the values allowed by the ASP parameter's, PDU field's or structure element's declared type. */

557 SingleValueLength ::= "[" ValueBound "]"

558 ValueBound ::= Number | TS_ParIdentifier | TS_ConstIdentifier | FormalParIdentifier

/* OPERATIONAL SEMANTICS - ValueBound shall evaluate to a specific non-negative INTEGER value. */

559 RangeValueLength ::= "[" LowerValueBound To UpperValueBound "]"

/* OPERATIONAL SEMANTICS - LowerValueBound shall be less than UpperValueBound. */

560 LowerValueBound ::= ValueBound

561 UpperValueBound ::= ValueBound | **INFINITY**

A.3.3.23 ASN.1 PDU Constraint Declarations

562 ASN1_PDU_Constraints ::= **$ASN1_PDU_Constraints** {ASN1_PDU_ConstraintOrGroup}+ **$End_ASN1_PDU_Constraints**

563 ASN1_PDU_ConstraintOrGroup ::= ASN1_PDU_Constraint | ASN1_PDU_ConstraintGroup

564 ASN1_PDU_ConstraintGroup ::= **$ASN1_PDU_ConstraintGroup** ASN1_PDU_ConstraintGroupId {ASN1_PDU_ConstraintOrGroup}+ **$End_ASN1_PDU_ConstraintGroup**

565 ASN1_PDU_ConstraintGroupId ::= **$ASN1_PDU_ConstraintGroupId** ASN1PDU_ConstraintGroupIdentifier

566 ASN1_PDU_Constraint ::= **$Begin_ASN1_PDU_Constraint** ConsId [ASN1PDU_ConstraintGroupRef] PDU_Id DerivPath [EncRuleId] [EncVariationId] [Comment] [ASN1_ConsValue] [Comment] **$End_ASN1_PDU_Constraint**

/* STATIC SEMANTICS - The FullIdentifier that is part of PDU_Id shall not be used. */

/* STATIC SEMANTICS - If a PDU is substructured, then the constraints for PDUs of that type shall have a compatible ASN.1 structure (*i.e.*, possibly with some groupings). */

/* STATIC SEMANTICS - A modified constraint shall have the same parameter list as its base constraint. In particular, there shall be no parameters omitted from or added to this list. */

567 ASN1PDU_ConstraintGroupRef ::= **$ASN1PDU_ConstraintGroupRef** ASN1PDU_ConstraintGroupReference

568 ASN1PDU_ConstraintGroupReference ::= [(SuiteIdentifier | TTCN_ModuleIdentifier) "/"] {ASN1PDU_ConstraintGroupIdentifier "/"}

569 ASN1PDU_ConstraintGroupIdentifier ::= Identifier

570 ASN1_ConsValue ::= **$ASN1_ConsValue** ConstraintValue&AttributesOrReplace **$End_ASN1_ConsValue**

571 ConstraintValue&AttributesOrReplace ::= ConstraintValue&Attributes | Replacement {Comma Replacement}

572 Replacement ::= **REPLACE** ReferenceList **BY** ConstraintValue&Attributes | **OMIT** ReferenceList

/* STATIC SEMANTICS - Replacement shall be used only when DerivPath is specified. */

/* STATIC SEMANTICS - Parameterized replaced values in a base constraint shall not be modified or explicitly omitted in a modified constraint. */

573 ReferenceList ::= (ArrayRef | ComponentIdentifier | ComponentPosition) {ComponentReference}

A.3.3.24 CM Constraint Declarations

574 CM_Constraints ::= **$CM_Constraints** [TTCN_CM_Constraints] [ASN1_CM_Constraints] **$End_CM_Constraints**

A.3.3.25 Tabular CM Constraint Declaration

575 TTCN_CM_Constraints ::= **$TTCN_CM_Constraints** {TTCN_CM_ConstraintOrGroup}+ **$End_TTCN_CM_Constraints**

576 TTCN_CM_ConstraintOrGroup ::= TTCN_CM_Constraint | TTCN_CM_ConstraintGroup

577 TTCN_CM_ConstraintGroup ::= **$TTCN_CM_ConstraintGroup** TTCN_CM_ConstraintGroupId {TTCN_CM_ConstraintOrGroup}+ **$End_TTCN_CM_ConstraintGroup**

578 TTCN_CM_ConstraintGroupId ::= **$TTCN_CM_ConstraintGroupId** CM_ConstraintGroupIdentifier

579 TTCN_CM_Constraint ::= **$Begin_TTCN_CM_Constraint** ConsId [CM_ConstraintGroupRef] CM_Id DerivPath [Comment] [CM_ParValues] [Comment] **$End_TTCN_CM_Constraint**

580 CM_ConstraintGroupRef ::= **$CM_ConstraintGroupRef** CM_ConstraintGroupReference

581 CM_ConstraintGroupReference ::= [(SuiteIdentifier | TTCN_ModuleIdentifier) “/”] {CM_ConstraintGroupIdentifier “/”}

582 CM_ConstraintGroupIdentifier ::= Identifier

583 CM_ParValues ::= **$CM_ParValues** {CM_ParValue} **$End_CM_ParValues**

584 CM_ParValue ::= **$CM_ParValue** CM_ParId ConsValue [Comment] **$End_CM_ParValue**

A.3.3.26 ASN.1 CM Constraint Declaration

585 ASN1_CM_Constraints ::= **$ASN1_CM_Constraints** {ASN1_CM_ConstraintOrGroup}+ **$End_ASN1_CM_Constraints**

586 ASN1_CM_ConstraintOrGroup ::= ASN1_CM_Constraint | ASN1_CM_ConstraintGroup

587 ASN1_CM_ConstraintGroup ::= **$ASN1_CM_ConstraintGroup** ASN1_CM_ConstraintGroupId {ASN1_CM_ConstraintOrGroup}+ **$End_ASN1_CM_ConstraintGroup**

588 ASN1_CM_ConstraintGroupId ::= **$ASN1_CM_ConstraintGroupId** ASN1CM_ConstraintGroupIdentifier

589 ASN1_CM_Constraint ::= **$Begin_ASN1_CM_Constraint** ConsId [ASN1CM_ConstraintGroupRef] CM_Id DerivPath [Comment] [ASN1_ConsValue] [Comment] **$End_ASN1_CM_Constraint**

590 ASN1CM_ConstraintGroupRef ::= **$ASN1CM_ConstraintGroupRef** ASN1CM_ConstraintGroupReference

591 ASN1CM_ConstraintGroupReference ::= [(SuiteIdentifier | TTCN_ModuleIdentifier) “/”] {ASN1CM_ConstraintGroupIdentifier “/”}

592 ASN1CM_ConstraintGroupIdentifier ::= Identifier

A.3.3.27 The Dynamic Part

593 DynamicPart ::= **$DynamicPart** [TestCases] [TestStepLibrary] [DefaultsLibrary] **$End_DynamicPart**

A.3.3.28 Test Cases

594 TestCases ::= **$TestCases** {TestGroup | TestCase}+ **$End_TestCases**

595 TestGroup ::= **$TestGroup** TestGroupId {TestGroup | TestCase}+ **$End_TestGroup**

596 TestGroupId ::= **$TestGroupId** TestGroupIdentifier

597 TestGroupIdentifier ::= Identifier

598 TestCase ::= **$Begin_TestCase** TestCaseId TestGroupRef TestPurpose [Configuration] DefaultsRef [Comment] BehaviourDescription [Comment] **$End_TestCase**

599 TestCaseId ::= **$TestCaseId** TestCaseIdentifier

600 TestCaseIdentifier ::= Identifier

601 TestGroupRef ::= **$TestGroupRef** TestGroupReference

602 TestGroupReference ::= [SuiteIdentifier "/"] {TestGroupIdentifier "/"}
/* STATIC SEMANTICS - There shall be no white space on either side of the "/"s. */

603 TestPurpose ::= **$TestPurpose** BoundedFreeText

604 Configuration ::= **$Configuration** TCompConfigIdentifier

605 DefaultsRef::= **$DefaultsRef** [DefaultRefList]

606 DefaultRefList ::= DefaultReference {Comma DefaultReference}

607 DefaultReference ::= DefaultIdentifier [ActualParList]

A.3.3.29 Test Step Library

608 TestStepLibrary ::= **$TestStepLibrary** {TestStepGroup | TestStep}+ **$End_TestStepLibrary**

609 TestStepGroup ::= **$TestStepGroup** TestStepGroupId {TestStepGroup | TestStep}+ **$End_TestStepGroup**

610 TestStepGroupId ::= **$TestStepGroupId** TestStepGroupIdentifier

611 TestStepGroupIdentifier ::= Identifier

612 TestStep ::= **$Begin_TestStep** TestStepId TestStepRef Objective DefaultsRef [Comment] BehaviourDescription [Comment] **$End_TestStep**

613 TestStepId ::= **$TestStepId** TestStepId&ParList

614 TestStepId&ParList ::= TestStepIdentifier [FormalParList]

615 TestStepIdentifier ::= Identifier

616 TestStepRef ::= **$TestStepRef** TestStepGroupReference

617 TestStepGroupReference ::= [SuiteIdentifier "/"] {TestStepGroupIdentifier "/"}
/* STATIC SEMANTICS - There shall be no white space on either side of the "/"s. */

618 Objective ::= **$Objective** BoundedFreeText

A.3.3.30 Default Library

619 DefaultsLibrary ::= **$DefaultsLibrary** {DefaultGroup | Default}+ **$End_DefaultsLibrary**

620 DefaultGroup ::= **$DefaultGroup** DefaultGroupId {DefaultGroup | Default}+ **$End_DefaultGroup**

621 DefaultGroupId ::= **$DefaultGroupId** DefaultGroupIdentifier

622 Default ::= **$Begin_Default** DefaultId DefaultRef Objective [Comment] BehaviourDescription [Comment] **$End_Default**
/* STATIC SEMANTICS - BehaviourDescription shall not use tree attachment except for attaching local trees (*i.e.*, Default behaviour trees shall not attach Test Steps). */

623 DefaultRef ::= **$DefaultRef** DefaultGroupReference

624 DefaultId ::= **$DefaultId** DefaultId&ParList

625 DefaultId&ParList ::= DefaultIdentifier [FormalParList]

626 DefaultIdentifier ::= Identifier

627 DefaultGroupReference ::= [SuiteIdentifier "/"] {DefaultGroupIdentifier "/"}
/* STATIC SEMANTICS - There shall be no white space on either side of the "/"s. */

628 DefaultGroupIdentifier ::= Identifier

A.3.3.31 Behaviour descriptions

629 BehaviourDescription ::= **$BehaviourDescription** RootTree {LocalTree} **$End_BehaviourDescription**

630 RootTree ::= {BehaviourLine}+

631 LocalTree ::= Header {BehaviourLine}+

632 Header ::= **$Header** TreeHeader

633 TreeHeader ::= TreeIdentifier [FormalParList]

634 TreeIdentifier ::= Identifier

635 FormalParList ::= "(" FormalPar&Type {SemiColon FormalPar&Type} ")"

636 FormalPar&Type ::= FormalParIdentifier {Comma FormalParIdentifier} Colon FormalParType

637 FormalParIdentifier ::= Identifier

638 FormalParType ::= Type | PCO_TypeIdentifier | **PDU** | **CP** | **TIMER**
/* STATIC SEMANTICS - In a test suite operation or an encoding operation FormalParType shall not be a PCO type or the keyword CP*/
/* STATIC SEMANTICS - If a formal parameter is of type **PDU** then that formal parameter shall not be used with a component reference (i.e. specific fields of the PDU cannot be referenced). */

A.3.3.32 Behaviour lines

639 BehaviourLine ::= **$BehaviourLine** LabelId Line Cref VerdictId [Comment] **$End_BehaviourLine**

640 Line ::= **$Line** Indentation StatementLine

641 Indentation ::= "[" Number "]"

/* STATIC SEMANTICS - Statements in the first level of alternatives in a behaviour description shall have the indentation value zero. */

/* STATIC SEMANTICS - Statements having a predecessor shall have the indentation value of the predecessor plus one as their indentation value. */

642 LabelId ::= **$LabelId** [Label]

643 Label ::= Identifier

644 Cref ::= **$Cref** [ConstraintReference]

645 ConstraintReference ::= ConsRef | FormalParIdentifier | AnyValue

/* STATIC SEMANTICS - ConsRef shall be present in conjunction with SEND, IMPLICIT SEND and RECEIVE and shall have a type which is consistent with (i.e. the same as or a subset of) the type of ASP, PDU or CM specified in the SEND, IMPLICIT_SEND or RECEIVE statement. A ConstraintReference is not needed for ASPs and CMs that have no parameters or PDUs that have no fields. It shall not be present with any other kind of TTCN statement. */

/* STATIC SEMANTICS - FormalParIdentifier shall resolve to a ConsRef. */

/* STATIC SEMANTICS - ConstraintReferences on SEND events shall not include any MatchingSymbol except Omit unless the MatchingSymbol is explicitly assigned specific values on the SEND event line. */

646 ConsRef ::= ConstraintIdentifier [ActualCrefParList]

647 ActualCrefParList ::= "(" ActualCrefPar {Comma ActualCrefPar} ")"

/* STATIC SEMANTICS - See static semantics on production 675. */

648 ActualCrefPar ::= Value

/* NOTE - Through Value, it is possible to reach MatchingSymbol, TS_ParIdentifier, TS_ConstIdentifier, TS_VarIdentifier, TC_VarIdentifier, FormalParIdentifier or ConsRef. */

649 VerdictId ::= **$VerdictId** [Verdict]

650 Verdict ::= Pass | Fail | Inconclusive | Result

/* STATIC SEMANTICS - Verdict shall not occur corresponding to entries in the behaviour tree which are any of the following: empty, an ATTACH construct, a REPEAT construct, a GOTO construct, an IMPLICIT SEND or a RETURN. */

651 Pass ::= **PASS** | **P** | "(" **PASS** ")" | "(" **P** ")"

652 Fail ::= **FAIL** | **F** | "(" **FAIL** ")" | "(" **F** ")"

653 Inconclusive ::= **INCONC** | **I** | "(" **INCONC** ")" | "(" **I** ")"

654 Result ::= **R**

/* STATIC SEMANTICS - R shall not be used on the LHS of an assignment. */

A.3.3.33 TTCN statements

655 StatementLine ::= (Event [Qualifier] [AssignmentList] [TimerOps]) | (Qualifier [AssignmentList] [TimerOps]) | (AssignmentList [TimerOps]) | TimerOps | Construct | ImplicitSend

656 Event ::= Send | Receive | Otherwise | Timeout | Done

/* STATIC SEMANTICS - A Receive, Otherwise or Timeout event shall only be followed by other Receive, Otherwise and Timeout events through the remainder of the set of alternatives in a fully expanded tree. As a consequence, Default trees will contain only Receive, Otherwise and Timeout events on the first level of alternatives. */

657 Qualifier ::= "[" Expression "]"

/* OPERATIONAL SEMANTICS - Qualifier shall evaluate to a specific BOOLEAN value. */

658 Send ::= [PCO_Identifier | CP_Identifier | FormalParIdentifier] "!" (ASP_Identifier | PDU_Identifier | CM_Identifier)

/* STATIC SEMANTICS - PCO_Identifier, CP_Identifier or FormalParIdentifier shall be present unless the test suite uses only one PCO and no CP. */

/* STATIC SEMANTICS - FormalParIdentifier shall resolve to a PCO_Identifier or CP_Identifier.*/

/* STATIC SEMANTICS - Only CMs may be exchanged on CPs and only ASPs and PDUs may be exchanged on PCOs. */

659 ImplicitSend ::= "<" **IUT** "!" (ASP_Identifier | PDU_Identifier) ">"

/* STATIC SEMANTICS - ImplicitSend shall not be used unless the test method being used is one of the Remote Test Methods. */

660 Receive ::= [PCO_Identifier | CP_Identifier | FormalParIdentifier] "?" (ASP_Identifier | PDU_Identifier | CM_Identifier)

/* STATIC SEMANTICS - PCO_Identifier, CP_Identifier or FormalParIdentifier shall be present unless the test suite uses only one PCO and no CP. */

/* STATIC SEMANTICS - Only CMs may be exchanged on CPs and only ASPs and PDUs may be exchanged on PCOs. */

661 Otherwise ::= [PCO_Identifier | CP_Identifier | FormalParIdentifier] "?" **OTHERWISE**

/* STATIC SEMANTICS - PCO_Identifier, CP_Identifier or FormalParIdentifier shall be present unless the test suite uses only one PCO and no CP. */

/* STATIC SEMANTICS - FormalParIdentifier shall only be of PCO type. */

662 Timeout ::= "?" **TIMEOUT** [TimerIdentifier | FormalParIdentifier]

/* STATIC SEMANTICS - FormalParIdentifier shall only be of TIMER type. */

663 Done ::= "?" **DONE** "(" [TCompIdList] ")"

664 TCompIdList ::= TCompIdentifier {Comma TCompIdentifier}

665 Construct ::= GoTo | Attach | Repeat | Return | Activate | Create

666 Activate ::= **ACTIVATE** "(" [DefaultRefList] ")"

/* STATIC SEMANTICS - The ACTIVATE construct shall not be used in Default behaviour tables. */

667 Return ::= **RETURN**

/* STATIC SEMANTICS - The RETURN construct shall not be used except in Default behaviour trees (including any local trees within Default behaviour tables). */

668 Create ::= **CREATE** "(" CreateList ")"

669 CreateList ::= CreateTComp {Comma CreateTComp}

670 CreateTComp ::= TCompIdentifier Colon TreeReference [ActualParList]

/* STATIC SEMANTICS - TCompIdentifier shall not be of Role MTC */

671 GoTo ::= ("->" | **GOTO**) Label

/* STATIC SEMANTICS - The label column shall contain labels referenced from the GoTo. */

/* STATIC SEMANTICS - Label shall be associated with the first of a set of alternatives, one of which is an ancestor node of the point from which the GoTo is to be made. */

/* STATIC SEMANTICS - GoTo shall be used only for jumps within one tree, *i.e.,* within a Test Case root tree, a Test Step tree a Default tree and a local tree; and thus, each label used in a GoTo construct shall be found within the tree in which the GoTo is used. */

/* STATIC SEMANTICS - There shall be no ACTIVATE operation as an ancestor node of the GoTo construct on the branch of the tree between the Label and the GoTo. */

/* STATIC SEMANTICS - No GoTo shall be made to the first level of alternatives of local trees, Test Steps or Defaults. */

672 Attach ::= "+" TreeReference [ActualParList]

/* STATIC SEMANTICS - TreeReference shall not attach itself, either directly or indirectly, at its top level of indentation. */

/* STATIC SEMANTICS - The number of the actual parameters shall be the same as the number of the formal parameters. */

/* STATIC SEMANTICS - Formal and actual parameters of test steps shall be used in such a way that only valid TTCN is created by textual substitution. */

/* STATIC SEMANTICS - LiteralValue, TS_ParIdentifier, TS_ConstIdentifier, TS_VarIdentifier, TC_VarIdentifier, ConsRef, MatchingSymbol, FormalParIdentifier, PCO_Identifier and CP_Identifier may be passed as actual parameters to an attached tree. */

673 Repeat ::= **REPEAT** TreeReference [ActualParList] **UNTIL** Qualifier

/* STATIC SEMANTICS - TreeReference shall not attach itself, either directly or indirectly, at its top level of indentation. */

/* STATIC SEMANTICS - The number of the actual parameters shall be the same as the number of the formal parameters. */

/* STATIC SEMANTICS - LiteralValue, TS_ParIdentifier, TS_ConstIdentifier, TS_VarIdentifier, TC_VarIdentifier, ConsRef, MatchingSymbol, FormalParIdentifier, PCO_Identifier and CP_Identifier may be passed as actual parameters to the tree in a REPEAT statement. */

674 TreeReference ::= TestStepIdentifier | TreeIdentifier

/* STATIC SEMANTICS - TreeIdentifier shall be the name of one of the trees in the current behaviour description, *i.e.,* local trees are not accessible outside the behaviour description in which they are specified. */

675 ActualParList ::= "(" ActualPar {Comma ActualPar} ")"

/* STATIC SEMANTICS - The number of the actual parameters shall be the same as the number of the formal parameters. */

/* OPERATIONAL SEMANTICS - Each actual parameter shall resolve to a specific value compatible with the type of its corresponding formal parameter, or in the case of predefined operations compatible with the types for which the operation is defined. */

/* STATIC SEMANTICS - If a parameter is a parameterized constraint then the constraint shall be passed together with its actual parameter list. */

/* STATIC SEMANTICS - The actual parameters shall be bound. */

/* STATIC SEMANTICS - If the type of the formal parameter is PDU, then the actual parameter's type shall be declared as PDU or as a specific **PDU** type. */

676 ActualPar ::= Value | PCO_Identifier | CP_Identifier | TimerIdentifier

/* NOTE - Through Value, it is possible to reach MatchingSymbol, TS_ParIdentifier, TS_ConstIdentifier, TS_VarIdentifier, TC_VarIdentifier, FormalParIdentifier or ConsRef. */

A.3.3.34 Expressions

677 AssignmentList ::= "(" Assignment {Comma Assignment} ")"

678 Assignment ::= DataObjectReference ":=" Expression

/* STATIC SEMANTICS - Except within a Procedural Definition or an Encoding Definition, the LHS of Assignment shall only resolve to: TS_VarIdentifier, TC_VarIdentifier, reference to the field of a variable or reference to an ASP parameter or PDU field that is to be sent. */

/* STATIC SEMANTICS - Within a procedure definition of a TSOp or EncodingOp, the DataObject Identifier on the left-hand side of an assignment shall be a VarIdentifier. */

/* STATIC SEMANTICS - The expression shall contain no unbound variables. */

/* OPERATIONAL SEMANTICS - The Expression on the RHS of Assignment shall evaluate to an explicit value of the type of the LHS. */

679 Expression ::= SimpleExpression [RelOp SimpleExpression]

/* OPERATIONAL SEMANTICS - If both SimpleExpressions and the RelOp exist then the SimpleExpressions shall evaluate to specific values of compatible types. */

/* OPERATIONAL SEMANTICS - If RelOp is "<" | ">" | ">=" | "<=" then each SimpleExpression shall evaluate to a specific INTEGER value. */

/* STATIC SEMANTICS - ASN.1 Named Values shall not be used within arithmetic expressions as operands of operations. */

680 SimpleExpression ::= Term {AddOp Term}

/* OPERATIONAL SEMANTICS - Each Term shall resolve to a specific value. If more than one Term exists and if AddOp is "OR" then the Terms shall resolve to type BOOLEAN; if AddOp is "+" or "-" then the Terms shall resolve to type INTEGER. */

681 Term ::= Factor {MultiplyOp Factor}

/* OPERATIONAL SEMANTICS - Each Factor shall resolve to a specific value. If more than one Factor exists and if MultiplyOp is "AND" then the Factors shall resolve to type BOOLEAN; if MultiplyOp is "*" or "/" then the Factors shall resolve to type INTEGER. */

682 Factor ::= [UnaryOp] Primary

/* OPERATIONAL SEMANTICS - The Primary shall resolve to a specific value. If UnaryOp exists and is "NOT" then Primary shall resolve to type BOOLEAN; if the UnaryOp is "+" or "-" then Primary shall resolve to type INTEGER. */

683 Primary ::= Value | DataObjectReference | OpCall | SelectExprIdentifier | "(" Expression ")"

/* STATIC SEMANTICS - SelectExprIdentifier shall only be used within selection expressions. */

/* NOTE - Through Value, it is possible to reach MatchingSymbol, TS_ParIdentifier, TS_ConstIdentifier, TS_VarIdentifier, TC_VarIdentifier, FormalParIdentifier or ConsRef. */

684 DataObjectReference ::= DataObjectIdentifier {ComponentReference}

/* STATIC SEMANTICS - Identifiers of ASP parameters and PDU fields associated with SEND and RECEIVE shall be used only to reference ASP parameter and PDU field values on the statement line itself. */

/* STATIC SEMANTICS - Each ComponentReference shall only reference an ASP parameter, PDU field, structure element or ASN.1 value explicitly declared in the object that immediately precedes in the DataObjectReference. */

/* STATIC SEMANTICS - DataObjectIdentifier shall not be a VarIdentifier except within a procedure definition of a TestSuiteOperation or EncodingOperation. */

685 DataObjectIdentifier ::=TS_ParIdentifier |TS_ConstIdentifier |TS_VarIdentifier |TC_VarIdentifier |FormalParIdentifier | ASP_Identifier | PDU_Identifier | CM_Identifier | VarIdentifier

686 ComponentReference ::= RecordRef | ArrayRef | BitRef

/* STATIC SEMANTICS - RecordRef shall be used to reference ASN.1 SEQUENCE, SET and CHOICE components. It shall not be used to reference components of any other ASN.1 type. */

/* STATIC SEMANTICS - RecordRef shall be used to reference ASP parameters, PDU fields and structure elements in the tabular form. */

/* STATIC SEMANTICS - ArrayRef shall be used to reference ASN.1 SEQUENCE OF and SET OF components. It shall not be used to reference components of any other ASN.1 type. */

687 RecordRef ::= Dot (ComponentIdentifier | ComponentPosition)

/* STATIC SEMANTICS - The ComponentIdentifier form of RecordRef shall always be used to reference ASN.1 SEQUENCE, SET and CHOICE components when an identifier is declared for the component. */

/* STATIC SEMANTICS - The ComponentIdentifier form of RecordRef shall always be used to reference ASP parameters, PDU fields and structure elements declared in the tabular form. */

/* STATIC SEMANTICS - The ComponentPosition form of RecordRef shall always be used to reference ASN.1 SEQUENCE, SET and CHOICE components when an identifier is not declared for the component. */

/* STATIC SEMANTICS - StructIdentifier shall not be used if the relevant structure is used as a macro. StructIdentifiers and PDU_Identifiers shall not be included in a RecordRef when a parameter, field or element is chained to a PDU or structure and the RecordRef is to identify a component of that PDU or structure. */

/* STATIC SEMANTICS - Where a structure is used as a macro expansion, the elements in the structure shall be referred to as if it was expanded into the ASP or PDU referring to it. */

/* STATIC SEMANTICS - If a parameter, field or element is defined to be of metatype PDU no reference shall be made to fields of that substructure. */

688 ComponentIdentifier ::= ASP_ParIdentifier | PDU_FieldIdentifier | CM_ParIdentifier | ElemIdentifier | ASN1_Identifier

689 ASN1_Identifier ::= Identifier

/* NOTE - ASN1_Identifier identifies a field within ASN.1 SEQUENCE, SET or CHOICE type. */

/* STATIC SEMANTICS - An ASN1_Identifier associated with a NamedValue shall not be used unless the value is within a SEQUENCE, SET or CHOICE type. */

/* STATIC SEMANTICS - An ASN1_Identifier shall be provided to identify the variant in a CHOICE type. */

/* STATIC SEMANTICS - An ASN1_Identifier shall be provided whenever the value definition becomes ambiguous because of omitted OPTIONAL values in a SEQUENCE type. */

690 ComponentPosition ::= "(" Number ")"

691 ArrayRef ::= Dot "[" ComponentNumber "]"

692 ComponentNumber ::= Expression

/* OPERATIONAL SEMANTICS - ComponentNumber shall evaluate to a non-negative specific INTEGER value. */

693 BitRef ::= Dot (BitIdentifier | "[" BitNumber "]")

694 BitIdentifier ::= Identifier

/* NOTE - BitIdentifier identifies a particular bit within an ASN.1 BIT STRING. */

695 BitNumber ::= Expression

/* OPERATIONAL SEMANTICS - BitNumber shall evaluate to a non-negative specific INTEGER value. */

696 OpCall ::= OpIdentifier (ActualParList | "(" ")")

/* STATIC SEMANTICS - See static semantics on production 675. */

697 OpIdentifier ::= TS_OpIdentifier | TS_ProcIdentifier | PredefinedOpIdentifier

698 PredefinedOpIdentifier ::= **BIT_TO_INT** | **HEX_TO_INT** | **INT_TO_BIT** | **INT_TO_HEX** | **IS_CHOSEN** | **IS_PRESENT** | **LENGTH_OF** | **NUMBER_OF_ELEMENTS**

699 AddOp ::= "+" | "-" | **OR**

/* OPERATIONAL SEMANTICS - Operands of the “+”, “-” operators shall be of type INTEGER (*i.e.,* TTCN or ASN.1 predefined) or derivations of INTEGER (*i.e.,* subrange). Operands of the OR operator shall be of type BOOLEAN (TTCN or ASN.1 predefined) or derivatives of BOOLEAN. */

700 MultiplyOp ::= "*" | "/" | **MOD** | **AND**

/* OPERATIONAL SEMANTICS - Operands of the “*”, “/” and MOD operators shall be of type INTEGER (*i.e.,* TTCN or ASN.1 predefined) or derivations of INTEGER (*i.e.,* subrange). Operands of the AND operator shall be of type BOOLEAN (TTCN or ASN.1 predefined) or derivatives of BOOLEAN. */

701 UnaryOp ::= "+" | "-" | **NOT**

/* OPERATIONAL SEMANTICS - Operands of the “+”, “-” operators shall be of type INTEGER (*i.e.,* TTCN or ASN.1 predefined) or derivations of INTEGER (*i.e.,* subrange). Operands of the NOT operator shall be of type BOOLEAN (TTCN or ASN.1 predefined) or derivatives of BOOLEAN. */

702 RelOp ::= "=" | "<" | ">" | "<>" | ">=" | "<="

A.3.3.35 Timer operations

703 TimerOps ::= TimerOp {Comma TimerOp}

704 TimerOp ::= StartTimer | CancelTimer | ReadTimer

705 StartTimer ::= **START** (TimerIdentifier | FormalParIdentifier) ["(" TimerValue ")"]

/* STATIC SEMANTICS - FormalParIdentifier shall only be of TIMER type. */

706 CancelTimer ::= **CANCEL** [TimerIdentifier | FormalParIdentifier]

/* STATIC SEMANTICS - FormalParIdentifier shall only be of TIMER type. */

707 TimerValue ::= Expression

/* OPERATIONAL SEMANTICS - Timervalue shall evaluate to a non-zero positive INTEGER. */

708 ReadTimer ::= **READTIMER** (TimerIdentifier | FormalParIdentifier) "(" DataObjectReference ")"

/* STATIC SEMANTICS - FormalParIdentifier shall only be of TIMER type. */

/* STATIC SEMANTICS - The DataObjectReference shall only resolve to TS_VarIdentifier, TC_VarIdentifier, or reference to the field of a variable. */

/* OPERATIONAL SEMANTICS - The DataObjectReference shall resolve to type INTEGER. */

A.3.3.36 Types

709 TypeOrPDU ::= Type | **PDU**

710 Type ::= PredefinedType | ReferenceType

A.3.3.36.1 Predefined types

711 PredefinedType ::= **INTEGER** | **BOOLEAN** | **BITSTRING** | **HEXSTRING** | **OCTETSTRING** | **OBJECTIDENTIFIER** | **R_Type** | CharacterString

712 CharacterString ::= **NumericString** | **PrintableString** | **TeletexString** | **VideotexString** | **VisibleString** | **IA5String** | **GraphicString** | **GeneralString** | **T61String** | **ISO646String**

A.3.3.36.2 Referenced types

713 ReferenceType ::= TS_TypeIdentifier | ASP_Identifier | PDU_Identifier | CM_Identifier

/* STATIC SEMANTICS - All types, other than the predefined types, used in a test suite shall be declared in the Test Suite Type definitions, ASP type definitions, PDU type definitions or CM type definitions, and referenced by name. */

714 TS_TypeIdentifier ::= SimpleTypeIdentifier | StructIdentifier | ASN1_TypeIdentifier

A.3.3.37 Values

715 Value ::= LiteralValue | ASN1_Value [ASN1_Encoding]

/* REFERENCE - Where ASN1_Value is the non-terminal Value as defined in ISO/IEC 8824: 1990. For the purposes of TTCN, the following production defined in ISO/IEC 8824: 1990:

DefinedValue ::= Externalvaluereference | valuereference

is redefined to be:

DefinedValue ::= ConstraintValue&Attributes | valuereference

This means that ASN.1 external references are not allowed in TTCN, but the full possibilities of ConstraintValue&Attributes as defined in production 538 are allowed within ASN.1 values in TTCN. This means that expressions, matching symbols, constraint references, value lengths, IF_PRESENT, and ASN.1 field encoding operations are all included . */

/* STATIC SEMANTICS - ASN.1 Named Values shall not be used within arithmetic expressions as operands of operations. */

716 LiteralValue ::= Number | BooleanValue | Bstring | Hstring | Ostring | Cstring | R_Value

717 Number ::= (NonZeroNum {Num}) | **0**

718 NonZeroNum ::= **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9**

719 Num ::= **0** | NonZeroNum

720 BooleanValue ::= **TRUE** | **FALSE**

721 Bstring ::= "’" {Bin | Wildcard} "’" **B**

722 Bin ::= **0** | **1**

723 Hstring ::= "’" {Hex | Wildcard} "’" **H**

724 Hex ::= Num | **A** | **B** | **C** | **D** | **E** | **F**

725 Ostring ::= "’" {Oct | Wildcard} "’" **O**

726 Oct ::= Hex Hex

727 Cstring ::= """ {Char | Wildcard | "\"} """

728 Char ::= */* REFERENCE - A character defined by the relevant CharacterString type. */*

/* LEXICAL REQUIREMENT - If the CharacterString type includes the character " (double quote), this character shall be represented by a pair of " (double quote) in the denotation of any value. */

729 Wildcard ::= AnyOne | AnyOrNone

730 AnyOne ::= "?"

/* STATIC SEMANTICS - AnyOne shall be used only within values of string types, SEQUENCE OF and SET OF. */

731 AnyOrNone ::= "*"

/* STATIC SEMANTICS - AnyOrNone shall be used only within values of string types, SEQUENCE OF and SET OF. */

732 R_Value ::= **pass** | **fail** | **inconc** | **none**

733 Identifier ::= Alpha{AlphaNum | Underscore}

/* STATIC SEMANTICS - All Identifiers referenced in a TTCN test suite shall be explicitly declared in the test suite, explicitly declared in an ASN.1 type definition referenced by the test suite or be a TTCN predefined identifier. */

734 Alpha ::= UpperAlpha | LowerAlpha

735 AlphaNum ::= Alpha | Num

736 UpperAlpha ::= **A** | **B** | **C** | **D** | **E** | **F** | **G** | **H** | **I** | **J** | **K** | **L** | **M** | **N** | **O** | **P** | **Q** | **R** | **S** | **T** | **U** | **V** | **W** | **X** | **Y** | **Z**

737 LowerAlpha ::= **a** | **b** | **c** | **d** | **e** | **f** | **g** | **h** | **i** | **j** | **k** | **l** | **m** | **n** | **o** | **p** | **q** | **r** | **s** | **t** | **u** | **v** | **w** | **x** | **y** | **z**

738 ExtendedAlphaNum ::= */* REFERENCE - A character from any character set defined in ISO/IEC 10646. */*

739 BoundedFreeText ::= *"/*"* FreeText *"*/"*

740 FreeText ::= {ExtendedAlphaNum}
/* LEXICAL REQUIREMENT - Free Text shall not contain the string "*/" unless preceded by backslash ("\"). */

A.3.3.38 Miscellaneous productions

741 Comma ::= ","

742 Dot ::= "."

743 Dash ::= "-"

744 Minus ::= "-"

745 SemiColon ::= ";"

746 Colon ::= ":"

747 Underscore ::= "_"

A.4 General static semantics requirements

A.4.1 Introduction

Static semantics requirements that are related to specific BNF productions are specified as comments on the relevant productions, in the following format:

/* STATIC SEMANTICS - ... */

All other static semantic requirements that are common to both TTCN.GR and TTCN.MP are specified in the remainder of A.4. Additional static semantics in the TTCN.MP are specified in A.5.2.

A.4.2 Uniqueness of identifiers

A.4.2.1 In some cases test suites may make references to items defined in other OSI standards. In particular, references to ASN.1 type definition modules according to ISO/IEC 8824: 1990 may be made in the type definitions. Names from those modules (such as identifiers of subfields within structured ASN.1 type definitions) may be used throughout the test suite.

Since the rules for identifiers in ASN.1 and TTCN conflict, the following conventions apply:

a) type references, module identifiers and value references made within the various ASN.1 type definitions tables shall comply to the requirements for identifiers defined in ISO/IEC 8824: 1990;

b) for identifiers used within the other parts of a test suite dash (-) characters shall be replaced with underscores (_).

Within some TTCN tables part of the ASN.1syntax can be used to define types.In that case, ASN.1 rules shall be followed for identifiers, with the exception that dash (-) characters shall not be used. Underscores (_) may be used instead. All other requirements defined by ISO/IEC 8824: 1990 (*e.g.,* Type identifiers shall start with an upper case letter, and field identifiers within structured ASN.1 definitions shall start with a lower case letter) apply to TTCN test suites wherever ASN.1 is used.

A.4.2.2 All identifiers of the following TTCN objects shall be unique throughout the test suite:

a) Test Suite Types;

b) Test Suite Operations;

c) Test Suite Parameters;

d) Test Case Selection Expressions;

e) Test Suite Constants;

f) Test Suite Variables;

g) Test Case Variables;

h) PCO types;

NOTE - If there is no PCO type declaration table, then PCO types are implicitly declared in the PCO declaration table, in which case the uniqueness refers to the meaning of the PCO type - the same PCO type may occur several times in the PCO declaration table with the same meaning.

i) PCOs;

j) CPs;

k) Timers;

l) Test Components;

m) Test Component Configurations;

n) ASP types;

o) PDU types;

p) CM types;

q) Structured Types;

r) Encoding Rules;

s) Encoding Variations;

t) Invalid Field Encodings;

u) Aliases;

v) ASP constraints;

w) PDU constraints;

x) CM constraints;

y) Structure constraints;

z) Test Cases;

aa) Test Steps;

ab) Defaults;

ac) Encoding Rule Names;

ad) Encoding Variation Names;

ae) Invalid Field Encoding Names.

A.4.2.3 All the following TTCN object references shall be unique throughout the test suite:

a) Test Group References;

b) Test Step Group References;

c) Default Group References.

A.4.2.4 TTCN reserved words are listed in table A.2 These reserved words shall not be used as identifiers in a TTCN test suite. All TTCN reserved words and TTCN identifiers are case sensitive.

Table A.2 - TTCN Reserved Words

ACTIVATE	IA5String	pass
AND	IF	PDU
BEGIN	IF_PRESENT	PERMUTATION
BITSTRING	INCONC	PrintableString
BIT_TO_INT	inconc	ps
BOOLEAN	INFINITY	PTC
BY	INTEGER	R
CANCEL	INT_TO_BIT	READTIMER
CASE	INT_TO_HEX	REPEAT
COMPLEMENT	IS_CHOSEN	REPLACE
CP	IS_PRESENT	RETURN
CREATE	IUT	RETURNVALUE
DO	LT	R_Type
DONE	min	s
ELSE	MOD	START
ENC	ms	STATIC
END	MTC	SUPERSET
ENDCASE	NOT	SUBSET
ENDIF	ns	TeletexString
ENDVAR	OF	THEN
ENDWHILE	OMIT	TIMEOUT
F	OR	TIMER
FAIL	OTHERWISE	TO
fail	P	TRUE
FALSE	LENGTH_OF	UNTIL
GeneralString	none	us
GOTO	NUMBER_OF_ELEMENTS	UT
GraphicString	NumericString	VAR
HEXSTRING	OCTETSTRING	VideotexString
HEX_TO_INT	OBJECTIDENTIFIER	VisibleString
I	PASS	WHILE

A.4.2.5 The ASN.1 reserved words are listed in table A.3. These reserved words shall not be used as identifiers in a TTCN test suite.

Table A.3 - - ASN.1 Reserved Words

ABSENT	FROM	OPTIONAL
ANY	GeneralString	PRESENT
APPLICATION	GeneralizedTime	PRIVATE
BEGIN	GraphicString	PrintableString
BIT	IA5String	REAL
BOOLEAN	IDENTIFIER	SEQUENCE
CHOICE	IMPLICIT	SET
COMPONENT	IMPORT	SIZE
COMPONENTS	INCLUDES	STRING
DEFAULT	INTEGER	T61String
DEFINED	ISO646String	TRUE
DEFINITIONS	MAX	TeletexString
END	MIN	UNIVERSAL
ENUMERATED	NULL	UTCTime
EXPLICIT	NumericString	VideotexString
EXPORT	OBJECT	VisibleString
EXTERNAL	OCTET	WITH
FALSE	OF	

A.4.2.6 When ASN.1 is used in a TTCN test suite, ASN.1 identifiers from the following list shall be unique throughout the test suite, regardless of whether the ASN.1 definition is explicit or implicit by reference:

a) *TypeIdentifiers* of an ASN.1 Type Definition;

b) identifiers occurring in an ASN.1 ENUMERATED type as distinguished values;

c) identifiers occurring in a *NamedNumberList* of an ASN.1 INTEGER type.

A.4.2.7 The names of ASP parameters shall be unique within the ASP in which they are declared. The names of PDU fields shall be unique within the PDU in which they are declared. The names of CM parameters shall be unique within the CM in which they are declared.

A.4.2.8 If a Structured Type is used as a macro expansion, then the names of the elements within the Structured Type shall be unique within each ASP, PDU or CM where it will be expanded.

A.4.2.9 Labels used within a tree shall be unique within a tree (*i.e.,* Test Case root tree, Test Step tree, Default tree, local tree).

A.4.2.10 The tree header identifier used for local trees shall be unique within the dynamic behaviour description in which they appear, and shall not be the same as any identifier having a unique meaning throughout the test suite.

NOTE - This means that a local tree identifier may have the same name as a local tree identifier in another behaviour description, but not the same as another Test Step in the Test Step Library.

A.4.2.11 The formal parameter names which may optionally appear as part of the following shall be unique within that formal parameter list, and shall not be the same as any identifier having a unique meaning throughout the test suite**:**

a) Test suite operations definition;

b) Tree header of a local tree;

c) Test Step Identifier;

d) Default Identifier;

e) Parameterized constraint declaration.

A.4.2.12 A formal parameter name contained in the formal parameter list of a local tree header shall take precedence over a formal parameter name contained in the formal parameter list of the Test Step in which it is defined, within the scope of that local formal parameter list.

A.4.2.13 In concurrent TTCN, PCOs and CPs used in a Test Case shall only be those determined by the Test Component configuration for that Test Case.

A.4.2.14 Each identifier used in the procedural definition of a test suite operation shall be on of the following:

a) locally declared variable name;

b) a type name, used in a variable declaration;

c) a formal parameter name declared in a formal parameter list of the operation;

d) a test suite operation name.

The scope of formal parameter names and locally declared variable names is the procedural definition of the test suite operation.. Thus, the values of all other types of identifier are not directly accessible within the procedural definition of a test suite operation. To access such values they shall be passed as actual parameters to the test suite operation.

A.4.2.15 The constraints for TTCN Structured Types, TTCN ASPs,TTCN PDUs and TTCN CMs shall not be specified using ASN.1 tables (i.e., ASN.1 Type Constraints, ASN.1 ASP Constraints, ASN.1 PDU Constraints or ASN.1 CM Constraints). Conversely, the constraints for ASN.1 Types, ASN.1 ASPs, ASN.1 PDUs and ASN.1 CMs shall not be specified using TTCN tables (i.e., Structured Type Constraints, TTCN ASP Constraints, TTCN PDU Constraints or TTCN CM Constraints).

NOTE - However, when ASPs or PDUs are chained to other PDUs, the enclosing ASP or PDU may, for example, be specifiied in tabular TTCN, whereas the enclosed PDU may be specified in ASN.1.

A.5 Differences between TTCN.GR and TTCN.MP

A.5.1 Differences in syntax

The following is a list of syntax differences between TTCN.MP and TTCN.GR:

a) TTCN.MP uses keywords as delimiters between entries, while TTCN.GR uses boxes;

b) TTCN.MP uses an explicit denotation of indentation levels for test events, while indentation is indicated visually in TTCN.GR;

c) TTCN.MP contains an extra occurrence of the suite identifier, which is used to facilitate identification of the ATS in an automated method;

d) in TTCN.MP the Test Case behaviour descriptions are explicitly grouped by the inclusion of appropriate Test Group Identifiers in sequence before the Test Case behaviour descriptions belonging to each group; this information duplicates information contained in the Test Case Index and in the Test Group References of the Test Case behaviour descriptions;

e) the Test Suite Structure, Test Case Index, Test Step Index and Default Index tables require a page number for each entry; since page numbers are not relevant in the machine processable form they are not reflected in the TTCN.MP;

f) TTCN.GR supports both single and compact proformas for ASP and PDU constraints and Test Cases; the TTCN only supports BNF for the single table format and the presentation of a number of single tables in TTCN.GR compact format is a display issue; when mapping a compact constraints table to TTCN.MP (*i.e.,* single format), blank fields due to modification shall be omitted;

g) the symbols “/*” and “*/” which open and close BoundedFreeText strings in the TTCN.MP shall not appear in the TTCN.GR;

h) there are two alternative positions for the labels column in behaviour description tables in TTCN.GR, whereas there is a fixed position for the labels in TTCN.MP;

i) page and line continuation are TTCN.GR features which are not represented in the TTCN.MP;

j) page and line numbering are TTCN.GR features which are not represented in the TTCN.MP.

k) if in TTCN.GR group references are used with definitions, declarations or constraints to indicate an hierarchical grouping of objects, then in TTCN.MP each relevant group identifier is inserted before the syntax for the group of tables which share that group identifier and the syntax for the group identifier and following group of tables are enclosed in the appropriate TTCN.MP keywords, relevant to the type of object.

A.5.2 Additional static semantics in the TTCN.MP

The following is a list of the additional static semantics in the TTCN.MP:

a) in the TTCN.MP, statements in the first level of alternatives having no predecessor in the root or local tree they belong to have the indentation value of zero; statements having a predecessor shall have the indentation value of the predecessor plus one as their indentation value;

b) in the TTCN.MP, the Test Suite Structure information is in the form of Test Group Identifiers preceding Test Case behaviour descriptions shall be the same structure as defined by the part of the Test Suite Structure relevant to Test Groups and that defined by the Test Case Index.

List of BNF production numbers

A.6 Introduction

This section presents an alphabetical index of the BNF productions that appear in annex A. For each production the index gives a reference in terms of the production number (not page number).

EDITOR'S NOTE 1 - This BNF production index needs updating - please ignore it in this version.

~~A.7 The production index~~