UNION INTERNATIONALE DES TÉLÉCOMMUNICATIONS

UIT-T Z.141

(07/2001)

SECTEUR DE LA NORMALISATION
DES TÉLÉCOMMUNICATIONS
DE L'UIT

SÉRIE Z: LANGAGES ET ASPECTS GÉNÉRAUX LOGICIELS DES SYSTÈMES DE TÉLÉCOMMUNICATION

Techniques de description formelle

Notation combinée arborescente et tabulaire version 3 (TTCN-3): format de présentation tabulaire

Recommandation UIT-T Z.141

(Antérieurement Recommandation du CCITT)

RECOMMANDATIONS UIT-T DE LA SÉRIE Z

LANGAGES ET ASPECTS GÉNÉRAUX LOGICIELS DES SYSTÈMES DE TÉLÉCOMMUNICATION

TECHNIQUES DE DESCRIPTION FORMELLE	
Langage de description et de spécification (SDL)	Z.100–Z.109
Application des techniques de description formelle	Z.110–Z.119
Diagrammes des séquences de messages	Z.120–Z.129
LANGAGES DE PROGRAMMATION	
CHILL: le langage de programmation de l'UIT-T	Z.200–Z.209
LANGAGE HOMME-MACHINE	
Principes généraux	Z.300–Z.309
Syntaxe de base et procédures de dialogue	Z.310–Z.319
LHM étendu pour terminaux à écrans de visualisation	Z.320–Z.329
Spécification de l'interface homme-machine	Z.330–Z.399
QUALITÉ DES LOGICIELS DE TÉLÉCOMMUNICATION	Z.400–Z.499
MÉTHODES DE VALIDATION ET D'ESSAI	Z.500–Z.599

Pour plus de détails, voir la Liste des Recommandations de l'UIT-T.

Recommandation UIT-T Z.141

Notation combinée arborescente et tabulaire version 3 (TTCN-3): format de présentation tabulaire

Résumé

La présente Recommandation définit le format de présentation tabulaire de la notation combinée arborescente et tabulaire version 3 (TTCN-3).

Source

La Recommandation Z.141 de l'UIT-T, élaborée par la Commission d'études 10 (2001-2004) de l'UIT-T, a été approuvée le 22 juillet 2001 selon la procédure définie dans la Résolution 1 de l'AMNT.

AVANT-PROPOS

L'UIT (Union internationale des télécommunications) est une institution spécialisée des Nations Unies dans le domaine des télécommunications. L'UIT-T (Secteur de la normalisation des télécommunications) est un organe permanent de l'UIT. Il est chargé de l'étude des questions techniques, d'exploitation et de tarification, et émet à ce sujet des Recommandations en vue de la normalisation des télécommunications à l'échelle mondiale.

L'Assemblée mondiale de normalisation des télécommunications (AMNT), qui se réunit tous les quatre ans, détermine les thèmes d'étude à traiter par les Commissions d'études de l'UIT-T, lesquelles élaborent en retour des Recommandations sur ces thèmes.

L'approbation des Recommandations par les Membres de l'UIT-T s'effectue selon la procédure définie dans la Résolution 1 de l'AMNT.

Dans certains secteurs des technologies de l'information qui correspondent à la sphère de compétence de l'UIT-T, les normes nécessaires se préparent en collaboration avec l'ISO et la CEI.

NOTE

Dans la présente Recommandation, l'expression "Administration" est utilisée pour désigner de façon abrégée aussi bien une administration de télécommunications qu'une exploitation reconnue.

DROITS DE PROPRIÉTÉ INTELLECTUELLE

L'UIT attire l'attention sur la possibilité que l'application ou la mise en œuvre de la présente Recommandation puisse donner lieu à l'utilisation d'un droit de propriété intellectuelle. L'UIT ne prend pas position en ce qui concerne l'existence, la validité ou l'applicabilité des droits de propriété intellectuelle, qu'ils soient revendiqués par un Membre de l'UIT ou par une tierce partie étrangère à la procédure d'élaboration des Recommandations.

A la date d'approbation de la présente Recommandation, l'UIT n'avait pas été avisée de l'existence d'une propriété intellectuelle protégée par des brevets à acquérir pour mettre en œuvre la présente Recommandation. Toutefois, comme il ne s'agit peut-être pas de renseignements les plus récents, il est vivement recommandé aux responsables de la mise en œuvre de consulter la base de données des brevets du TSB.

TABLE DES MATIÈRES

Recommandation UIT-T Z.141

Notation combinée arborescente et tabulaire version 3 (TTCN-3): format de présentation tabulaire

1 Domaine d'application

La présente Recommandation définit le format de présentation tabulaire de la notation TTCN version 3 (TTCN-3, *tree and tabular combined notation version 3*). La présente Recommandation est fondée sur le langage noyau TTCN-3 défini dans l'UIT-T Z.140 [1].

La spécification d'autres formats n'entre pas dans le domaine d'application de la présente Recommandation.

2 Références normatives

La présente Recommandation se réfère à certaines dispositions des Recommandations UIT-T et textes suivants qui, de ce fait, en sont partie intégrante. Les versions indiquées étaient en vigueur au moment de la publication de la présente Recommandation. Toute Recommandation ou tout texte étant sujet à révision, les utilisateurs de la présente Recommandation sont invités à se reporter, si possible, aux versions les plus récentes des références normatives suivantes. La liste des Recommandations de l'UIT-T en vigueur est régulièrement publiée.

[1] UIT-T Z.140 (2001), *Notation combinée arborescente et tabulaire version 3 (TTCN-3): langage noyau*.

3 Abréviations

La présente Recommandation utilise les abréviations suivantes:

ASN.1 notation de syntaxe abstraite numéro un (*abstract syntax notation one*)

ATS suite de tests abstraits (*abstract test suite*)

BNF forme Backus-Naur (*Backus-Naur form*)

IUT implémentation sous test (*implementation under test*)

MTC composante de test maître (*master test component*)

PICS déclaration de conformité d'implémentation de protocole (*protocol implementation conformance statement*)

PIXIT informations complémentaires sur l'implémentation de protocole destinées au test (*protocol implementation extra information for testing*)

TTCN notation combinée arborescente et tabulaire (*tree and tabular combined notation*)

4 Introduction

Le format de présentation tabulaire est un format graphique dont l'aspect et les fonctionnalités sont analogues aux versions précédentes de la notation TTCN, qui sont orientées vers les tests de conformité. Le langage noyau de la notation TTCN-3 est défini dans l'UIT-T Z.140 [1] et spécifie une syntaxe en texte intégral, une sémantique statique et une sémantique opérationnelle ainsi que l'utilisation de cette notation avec l'ASN.1. Le format tabulaire est une autre façon de présenter le

langage noyau et de mettre l'accent sur les aspects qui sont propres aux exigences d'une suite de tests de conformité normalisée.

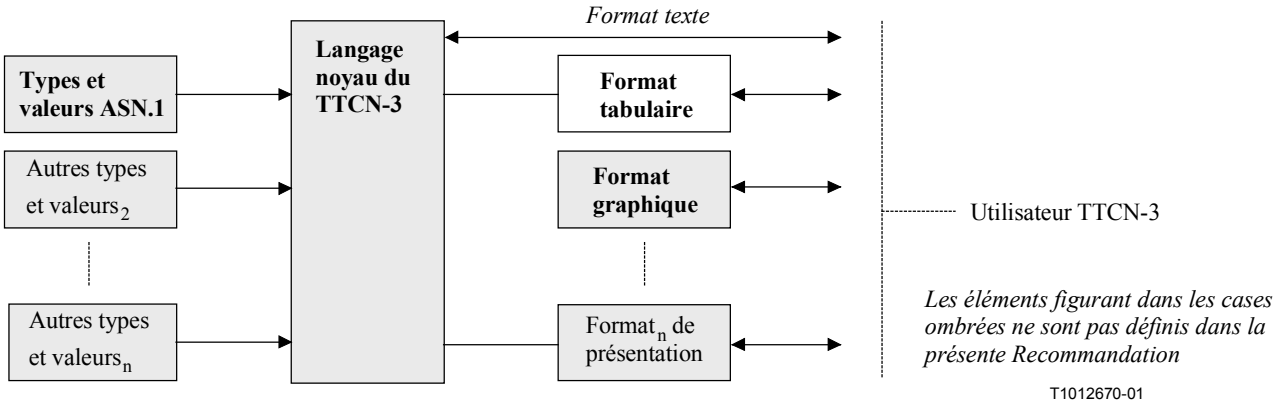

Figure 1/Z.141 – Vue utilisateur du langage noyau et des divers formats de présentation

Le langage noyau peut être utilisé indépendamment du format de présentation tabulaire. Toutefois, le format tabulaire ne peut pas être utilisé sans le langage noyau. Le format de présentation tabulaire doit être utilisé ou implémenté sur la base du langage noyau.

La présente Recommandation définit:

a) les formulaires;

b) les mappages syntaxiques;

c) la sémantique statique additionnelle;

d) les restrictions de sémantique opérationnelle;

e) la présentation et autres attributs.

L'ensemble de ces caractéristiques constitue le format de présentation tabulaire.

5 Conventions

Le présent paragraphe définit les conventions qui ont été utilisées pour définir les formulaires TTCN et la grammaire du langage noyau TTCN.

5.1 Métanotation syntaxique

Le Tableau 1 définit la métanotation utilisée pour spécifier la grammaire BNF étendue pour la notation TTCN (appelée dans ce qui suit BNF).

Tableau 1/Z.141 – Métanotation syntaxique TTCN.MP

::=	défini comme étant
abc xyz	abc suivi de xyz
|	alternative
[abc]	0 ou 1 instance de abc
{abc}	0 ou plusieurs instances de abc
{abc}+	1 ou plusieurs instances de abc
(...)	groupage textuel
abc	symbole non terminal abc
abc	symbole terminal abc
"abc"	symbole terminal abc

Les productions BNF sont définies au paragraphe 8. Celles qui ne sont pas définies au paragraphe 8 se trouvent dans l'UIT-T Z.140 [1].

5.2 Formulaire

a) Un texte en gras (**comme celui-ci**) doit apparaître dans son intégralité dans chaque tableau rempli dans une suite de test TTCN.

b) Un texte en italique (*comme celui-ci*) ne doit pas apparaître dans son intégralité dans une suite de tests TTCN. Cette police est utilisée pour indiquer que le texte réel doit être substitué au texte en italique. Les prescriptions de syntaxe pour le texte réel peuvent se trouver après la définition du formulaire ou dans le formalisme BNF du langage noyau TTCN.

c) Un fond ombré indique que le champ (la rangée, ou la colonne) est facultatif.

6 Règles générales de mappage

Le mappage entre le format de présentation tabulaire et le langage noyau TTCN-3 se compose d'un ensemble de transformations. Une transformation est associée à chaque élément syntaxique de chaque formulaire.

Ces transformations relèvent de deux classes. La première convertit directement un élément tabulaire en une structure de langage noyau de même signification. La deuxième convertit un élément tabulaire en une structure de langage noyau associée qui n'a pas de signification au niveau du langage noyau.

Un exemple type de la première classe de transformations est un champ identificateur. Ce champ peut être directement converti de la forme tabulaire au langage noyau et conserver sa signification, c'est-à-dire identifier certains éléments du langage.

La deuxième classe de transformations est en général une certaine forme de commentaire ou de directive sur la façon dont l'élément de langage doit être présenté dans le format de présentation. Ces éléments n'ont pas de signification directe en langage noyau et sont exprimés avec la déclaration **`with`**.

Ces déclarations **`with`** ont un format commun dont la forme est la suivante:

```
with display "<ProfomaIdentifier> { <ElementIdentifier1> := FreeText;
                                    <ElementIdentifier2> := FreeText;
                                    <ElementIdentifierN> := FreeText }"
```

L'identificateur <*ProfomaIdentifier*> est le nom du tableau associé dans le format tabulaire et <*ElementIdentifier*> est le nom du champ ou de l'élément dans ce tableau qui est défini par la déclaration. La valeur du champ ou de l'élément est spécifiée en texte libre après le caractère ':='. La définition BNF précise de l'interprétation de cette chaîne de présentation est définie dans la présente Recommandation.

La syntaxe et la sémantique spécifiées dans la présente Recommandation sont propres au format de présentation tabulaire de l'ETSI. Afin d'identifier sans ambiguïté à l'intérieur du langage noyau le format de présentation utilisé, la déclaration de présentation spéciale suivante doit être spécifiée comme étant la première déclaration de présentation dans le module de langage noyau TTCN-3:

```
module ModuleName()
{
}
with {   display "PresentationFormatIdentifer" }
```

```
PresentationFormatIdentifer ::= PresentationKeyword FormatKeyword
AssignmentChar "ETSI Tabular     v1.0"
PresentationKeyword ::= presentation
FormatKeyword ::= format
```

NOTE – Toutes les déclarations **with** associées à un formulaire donné doivent être regroupées dans une liste contiguë.

7 Formulaires

7.1 Contrôle de la suite de tests (test suite control)

Contrôle de la suite de tests (Test Suite Control)			
Nom	:	*TTCN3ModuleId*	
Version	:	*VersionIdentifier*	
Date	:	*Texte libre*	
Réf normes de base	:	*Texte libre*	
Réf normes de test	:	*Texte libre*	
Réf PICS	:	*Texte libre*	
Réf PIXIT	:	*Texte libre*	
Méthode(s)de test	:	Texte *libre*	
Commentaires détaillés	:	***[Texte libre]***	

Nom	**Type**	**Valeur initiale**	**Commentaires**
. *[VarConstOrTimerIdentifier]* . .	. *[TypeOrTimer]* . .	. *[ConstantExpression]* . .	. *[Texte libre]* . .

Comportement	**Commentaires**
. . *ModuleControlBody*	. . *[Texte libre]* . .
Commentaires détaillés: *[Texte libre]*	

Figure 2/Z.141 – Formulaire de contrôle de la suite de tests

7.1.1 Mappage

Le formulaire de contrôle de suite de tests est traduit en deux parties. La première est la partie contrôle du module en langage noyau TTCN-3. Les champs commentaires sont convertis en une déclaration **with** associés avec la définition de contrôle dans le langage noyau. L'information d'en-tête dans le formulaire de contrôle de suite de tests est convertie en une déclaration **with** associée au module global TTCN-3.

```
module MyModule()
{
     control{
          var Type VarIdentifier [ ":=" ConstantExpression]
          timer TimerIdentifier  [ ":=" ConstantExpression]
          const Type ConstIdentifier ":=" ConstantExpression

          ModuleControlBody

     } with display "ControlProforma"
}
with display "ReferenceProforma"
```

Exemple:

Contrôle de la suite de test (Test Suite Control)		
Nom	:	*MyATS*
Version	:	*1.1*
Date	:	*23 May 1999*
Réf. normes de base	:	
Réf. normes de test	:	
Réf. PICS	:	
Réf. PIXIT	:	
Méthode(s) de test	:	*local*
Commentaires détaillés	:	*ATS written by STF 133*

Nom	Type	Valeur initiale	Commentaires
X	**Integer**	7	
T1	**timer**	15 min	

Comportement	Commentaires
/* group1/ */ /* group1_1/ */ **execute**(test1); **execute**(test2); /* group1_2/ */ **execute**(test3); **execute**(test4); /* group2/ */ **execute**(test5);	basic tests check capability 1
Commentaires détaillés:	

Se mappe en:

```
module MyATS()
{

     control{
          var integer x := 7;
          timer T1 := 15 min;

          /* group1/ */
               /* group1_1/ */
                    execute( test1);
                    execute( test2);
               /* group1_2/ */
                    execute( test3);
                    execute( test4);
               /* group2/ */
               execute( test5);
     } with display " Control { extracomments := "basic tests
                                                 check capability 1"}";

} with
     {
display "presentation format                    :=  "ETSI Tabular v1.0";
display "reference {    version                 :=     "1.1";
                        date                    :=     "23 May 1999";
                        testmethod              :=     "local";
                        detailedcomments        :=     "ATS written by STF 133" }"
     }
```

7.2 Paramètres de suite de tests (test suite parameters)

Paramètres de la suite de tests (Test Suite Parameters)				
Groupe : *[GroupReference]*				
Name	**Type**	**Valeur initiale**	**Réf PICS/PIXIT**	**Commentaires**
. . *ModuleParIdentifier* . .	. . *ModuleParType* . .	. . *[ConstantExpression]*	. . *[Texte libre]*	. . *[Texte libre]* . .
Commentaires détaillés: *[Texte libre]*				

Figure 3/Z.141 – Formulaire de paramètres de suite de test

7.2.1 Mappage

Toutes les entrées du tableau de paramètres sont mappées en une liste de paramètres du module TTCN-3 associé. La PICS/PIXITref et les commentaires sont mappés en déclaration **with** du module TTCN-3.

```
module MyModule(ModuleType ModuleParIdentifier)
{
}
with display "ParametersProforma";
```

Exemple:

Paramètres de la suite de tests (Test Suite Parameters)				
Groupe : PICS/				
Nom	**Type**	**Valeur initiale**	**Réf PICS/PIXIT**	**Commentaires**
CAP_1	**Boolean**	**True**	A.1.3	Option 1 implemented by IUT
Commentaires détaillés:				

Se mappe en:

```
module MyModule(boolean CAP_1 := true)
{
}
with display "parameters {
              group := PICS/;
              pics pixet := { CAP_1 := "A.1.3"}
              comments := { CAP1 := "Option 1 implemented by IUT"}
              }"
```

7.3 Imports de modules

Imports				
Nom de la source :	*ModuleIdentifier [DefinitiveIdentifier]] ModuleIdentifier [DefinitiveIdentifier]*			
Langage source	**:**	*[LanguageSpec]*		
Groupe	**:**	*[GroupReference]*		
Réf. de la source	**:**	*[Texte libre]*		
Codage	**:**	*[Texte libre]*		
Commentaires	**:**	*[Texte libre]*		
Type		**Nom**	**NR**	**Commentaires**
. . *ImportType* . .		. . *ImportIdentifier* . .	. . *Mark* . .	. . *[Texte libre]* . .
Commentaires détaillés:	*[Texte libre]*			

Figure 4/Z.141 – Formulaire d'imports

7.3.1 Mappage

Le formulaire d'imports est mappé en un groupe de déclarations d'import en langage noyau TTCN-3. Le nom de la source, le type d'import, le nom d'import et les éléments tabulaires de récursion sont directement utilisés dans la délaration d'imports correspondante en langage noyau. Le groupe est appelé **`Imports`** avec un numéro univoque associé à la fin de l'identificateur lorsqu'il est nécessaire de lui donner un nom de groupe unique. Tous les autres champs sont traduits avec une déclaration **`with`** associée au groupe englobant.

```
module MyModule()
{
     group Imports1 {
          import ImportType  ImportIdentifier from
ModuleIdentifier[DefinitiveIdentifier] [language LanguageIdentifer];
}
with {    display "ImportsProforma";
          encode "FreeText" }
```

Exemple:

Imports				
Nom de la source	**:**	ModuleA		
Réf. de la source	**:**	EN 800 900 version 2		
Codage	**:**			
Commentaires	**:**	importing declarations from an existing ATS		
Type		**Nom**	**NR**	**Commentaires**
all constant type group		 MyType AtoU_CTR	*	(1)
Commentaires détaillés:	(1) Tick indique: importer de manière récursive ce qui est nécessaire pour la définition de MyType.			

Se mappe en:

```
module MyModule()
{
     group Imports1 {
          import all constant  from ModuleA;
          import type  MyType from ModuleA;
          import group  AtoU_CTR from ModuleA;
     }
     with display "imports { source    := "EN 800 900 version 2";
                            comments := "importing declarations from an existing
                            ATS";
                            extracomments := "(1)";
                            detailedcomments := "(1) asterisk indicates: import
                                                recursively what is needed for
                                                MyType definition"}"
}
```

7.4 Codage

Définitions de codage (Encoding Definitions)			
Groupe : *[GroupReference]*			
Nom	**Référence**	**Valeur par défaut**	**Commentaires**
.	.	.	.
.	.	.	.
EncodingRuleIdentifier	*Texte libre*	*[Boolean Expression]*	*[Texte libre]*
.	.	.	.
.	.	.	.
Commentaires détaillés: *[Texte libre]*			

Figure 5/Z.141 – Formulaire de définitions de codage

7.4.1 Mappage

Le formulaire de codage est mappé en une série de déclarations dans la déclaration **with** associée avec le module noyau TTCN-3. Tous les éléments du tableau sont mappés en déclarations de présentation. En outre, une déclaration de codage est ajoutée à la déclaration **with** pour la règle de codage dont la valeur par défaut est évaluée à **true**.

```
module MyModule()
{
}
with {
     display " EncodingProforma ";
     encode "EncodingRuleIdentifier"
}
```

Exemple:

Définition du codage (Encoding Definitions)			
Nom de la règle de codage	**Référence**	**Valeur par défaut**	**Commentaires**
BER	ISO/IEC 8825-1: 1993	TRUE	Basic Encoding Rules
PER	ISO/IEC 8825-1: 1993		Packed Encoding Rules
DER	ISO/IEC 8825-1: 1993		Distinguished Encoding Rules
Commentaires détaillés:			

Se mappe en:

```
module MyModule()
{
}
with {
display "encoding {     reference := {    BER := "ISO/IEC 8825-1: 1993",
                                          PER := "ISO/IEC 8825-1: 1993",
                                          DER := "ISO/IEC 8825-1: 1993"};
                        default   := {    BER := TRUE};
                        comments  := {    BER := "Basic Encoding Rules",
                                          PER := "Packed Encoding Rules",
                                          DER := "Distinguished Encoding Rules"}";
encode "BER" }
```

7.5 Types simples (simple types)

Types simples (Simple Types)			
Groupe : *[GroupReference]*			
Nom	**Définition**	**Codage**	**Commentaires**
. *SubTypeIdentifier* .	. *Type [SubTypeSpec]* .	. *[Texte libre]* .	. *[Texte libre]* .
Commentaires détaillés: *[Texte libre]*			

Figure 6/Z.141 – Formulaire de types simples

7.5.1 Mappage

Le format de types simples est mappé en un groupe TTCN-3 contenant une série de déclarations de définitions de type. La référence du groupe et les commentaires détaillés sont mappés en déclaration de présentation à l'intérieur de la déclaration **with** associée au groupe. Le codage et les champs commentaires sont mappés en déclaration avec la déclaration **with** associée à des définitions de type distinctes.

Le groupe sera appelé $SimpleTypes_n$ dans lequel "n" est un entier utilisé pour distinguer plusieurs groupes de type simple.

```
module MyModule()
{
     group SimpleTypes1 {
          type Type  SubTypeIdentifier SubTypeSpec
     }
     with {
          encode (SubTypeIdentifier) "FreeText";
          display "SimpleTypesProforma ";
 }
}
```

Exemple:

Types simples (Simple Types)			
Nom	**Définition**	**Codage**	**Commentaires**
EQ_NUMBER	**integer** (1 .. 20)		
Commentaires détaillés:			

Se mappe en:

```
module MyModule()
{
     group SimpleTypes1 {
          type integer EQ_NUMBER (1..20)
     }
     with display "simpletypes {}";
}
```

7.6 Types structurés (structured types)

Type structuré (Structured Type)			
Nom : *StructTypeIdentifier*			
Groupe : *[GroupReference]*			
Structure : *StructureType*			
Codage : *[Texte libre]*			
Commentaires : *[Texte libre]*			
Nom de l'élément	**Définition du type**	**Codage du champ**	**Commentaires**
. . *StructFieldIdentifier* . .	. . *Type [SubTypeSpec]* *[OptionalKeyword]* . .	. . *[Texte libre]* . .	. . *[Texte libre]* . .
Commentaires détaillés: *[Texte libre]*			

Figure 7/Z.141 – Formulaire de type structuré

7.6.1 Mappage

Le formulaire de type structuré est mappé en une déclaration de définition de type en TTCN-3, dont le groupe et les champs commentaires sont mappés en déclaration de présentation dans la déclaration **with** correspondante.

```
module MyModule()
{
          type StructureType  StructTypeIdentifier
          {
               Type FieldIdentifier [ SubtypeSpec][ OptionalKeyword]
          }
          with {
               display "StructuredTypeProforma";
               encode "FreeText";
               encode (StructFieldIdentifier)"FreeText";
     }

}
```

Exemple:

Type structuré (Structured Type)			
Nom : *MaleMind*			
Groupe :			
Structure : *record*			
Codage :			
Commentaires :			
Nom de l'élément	**Définition du type**	**Codage des champs**	**Commentaires**
Car	**integer**		
Money	**integer**		
Football	octetstring		
Commentaires détaillés:			

Se mappe en:

```
module MyModule()
{
        type record MaleMind
        {
             integer       Car,
             integer       Money,
             octetstring   Football
        }
        with display "structuredtype {   comments := "";
                                         comments := {}
                                         detailedcomments := ""}
}
```

7.7 Types de port (port types)

Type de port (Port Type)			
Nom	:	*PortTypeIdentifier*	
Groupe	:	*[GroupReference]*	
Modèle de communication	:	*PortModelType*	
Commentaires	:	*[Texte libre]*	
Définition du type			**Commentaires**
PortTypeDef . ..			. . *[Texte libre]* . .
Commentaires détaillés: *[Texte libre]*			

Figure 8/Z.141 – Formulaire de type de port

7.7.1 Mappage

Le formulaire de type port est mappé en une définition de type port en TTCN-3, avec les champs groupe et commentaire mappés en déclaration de présentation dans la déclaration **`with`** correspondante.

```
module MyModule()
{
          type port   PortTypeIdentifier PortModelType
          {
               PortTypeDef
          }
          with  display "PortTypeProforma";
   }
```

Exemple:

Type de port (Port Type)		
Nom	**:**	*MyPortType*
Groupe	**:**	
Modèle de communication	**:**	*message*
Commentaires	**:**	
Définition du type		**Commentaires**
in MsgType1, MsgType2; **out** MsgType3;		
Commentaires détaillés:		

Se mappe en:

```
module MyModule()
{
          type port MyPortType message
          {
               in MsgType1, MsgType2;
               out MsgType3;
          }
          with display "porttype {}";
}
```

7.8 Types de composante (component types)

Type de composante (Component Type)			
Nom	**:**	*ComponentTypeIdentifier*	
Groupe	**:**	*[GroupReference]*	
Commentaires	**:**	*[Texte libre]*	
Nom	**Type**	**Valeur initiale**	**Commentaires**
. *[VarConstOrTimerIdentifier]* . .	. *[TypeOrTimer]* . .	. *[ConstantExpression]* . .	. *[Texte libre]* . .
Définitions du port			**Commentaires**
PortList . .			*[Texte libre]* . . . .
Commentaires détaillés: *[Texte libre]*			

Figure 9/Z.141 – Formulaire de type de composante

7.8.1 Mappage

Le formulaire de types de composantes est mappé en une définition de type composante en TTCN-3, avec les champs groupe et commentaire mappés en déclarations d'affichage dans la déclaration **with** correspondante.

```
module MyModule()
{
        type component ComponentTypeIdentifier
        {
             var Type VarIdentifier [ ":=" ConstantExpression]
             timer TimerIdentifier  [ ":=" ConstantExpression]
             const Type ConstIdentifier ":=" ConstantExpression
             PortList
        }
        with  display "ComponentTypeProforma";
}
```

Exemple:

Type de composante (Component Type)			
Nom : MyComponentType			
Groupe :			
Commentaires :			
Nom	**Type**	**Valeur initiale**	**Commentaires**
x	**Integer**	7	
T1	**timer**	15 min	
Port Definitions			**Commentaires**
MyMessagePortType PCO1, PCO2; MyOtherPortType PCO3, PCO4;			
Commentaires détaillés:			

Se mappe en:

```
module MyModule()
{
        type component MyComponentType
        {
             var integer x := 7;
             timer T1 := 15 min;

             MyMessagePortType PCO1, PCO2;
             MyOtherPortType PCO3, PCO4;
        }
        with display "componenttype {}";
}
```

7.9 Constantes

Constantes (Constants)			
Groupe : *[GroupReference]*			
Nom	**Type**	**Valeur**	**Commentaires**
. *ConstIdentifier* .	. *Type* .	. *ConstantExpression* .	. *[Texte libre]* .
Commentaires détaillés: *[Texte libre]*			

Figure 10/Z.141 – Formulaire de constantes

7.9.1 Mappage

Le formulaire constantes est mappé en un groupe contenant des déclarations de constantes en langage TTCN-3. La référence du groupe et les commentaires sont mappés en déclarations de présentation intégrées à la déclaration **with** associée. Le groupe est nommé $Constants_n$ où "n" est un entier univoque associé à la fin de l'identificateur.

```
module MyModule()
{
     group Constants1 {
          const Type  ConstIdentifier := ConstantExpression
     }
     with display "ConstantsProforma";
```

Exemple:

Constantes (Constants)			
Groupe : Misc			
Nom	**Type**	**Valeur**	**Commentaires**
sel2	**boolean**	(5 + TOTO) < 10	TOTO est une constante
T1	**integer**	15	
Commentaires détaillés:			

Se mappe en:

```
module MyModule()
{
          group Constants1 {
               const boolean sel2 := (5 + TOTO) < 10;
               const integer T1 := 15;
          }
          with display "constants {   group := "Misc";
                                      comments := {sel2 ::= "TOTO is a
                                      constant"}}";
}
```

7.10 Signature

Définition de signature		
Nom	:	*SignatureIdentifier&ParList*
Groupe	:	*[GroupReference]*
Type de retour	:	*[Type]*
Commentaires	:	*[Texte libre]*
Liste d'exceptions		**Commentaires**
ExceptionTypeList . ..		. . *[Texte libre]* . .
Commentaires détaillés:	*[Texte libre]*	

Figure 11/Z.141 – Formulaire de définition de signature

7.10.1 Mappage

Le formulaire signature est mappé en une définition de signature en TTCN-3, dont le groupe et le champ commentaire sont mappés en déclarations de présentation dans la déclaration **`with`** correspondante.

```
signature SignatureIdentifier&ParList [return Type]
{
     ExceptionTypeList
}
with display "SignatureProforma";
```

Exemple:

Définition de signature		
Nom	:	MySignature(in integer Par1, out float Par2)
Groupe	:	
Type de retour	:	boolean
Commentaires	:	
Liste d'exceptions		**Commentaires**
integer, boolean, MyType		
Commentaires détaillés:		

Se mappe en:

```
signature MySignature( in interger Par1, out float Par2) return boolean
{
          integer, boolean, MyType
}
with display "signature {}";
```

7.11 Modèles simples (simple templates)

Modèles simples (Simple Templates)				
Groupe : *[GroupReference]*				
Nom	**Type**	***Valeur***	**Codage**	**Commentaires**
. *TemplateIdentifier* .	. *SimpleType* .	. *SingleValueOrAttrib*	. *[Texte libre]* .	. *[Texte libre]* .
Commentaires détaillés: *[Texte libre]*				

Figure 12/Z.141 – Formulaire de modèles simples

7.11.1 Mappage

Le formulaire modèles simples est mappé en un groupe TTCN-3 contenant une série de déclarations de définition de modèle. La référence du groupe et les commentaires détaillés son mappés en déclaration de présentations intégrée à la déclaration **`with`** associée au groupe. Le codage et les champs commentaires sont mappés en déclaration dont la déclaration **`with`** est associée aux définitions de modèles distincts.

Le groupe sera appelé SimpleTemplates_n où "n" est un entier utilisé pour distinguer plusieurs groupes de modèles simples

```
module MyModule()
{
     group SimpleTemplates1 {
          template SimpleType  TemplateIdentifier := SingleVlaueOrAttrib
     }
     with {
          encode (TemplateIdentifier) "FreeText";
          display "SimpleTemplatesProforma ";
 }
}
```

Exemple:

Modèles simples (Simple Templates)				
Nom	**Type**	***Valeur***	**Codage**	**Commentaires**
AgeField	**integer**	?		
Commentaires détaillés:				

Se mappe en:

```
module MyModule()
{
     group SimpleTemplates1 {
          template integer AgeField := ?;
     }
     with display "simpletemplates {}";
}
```

7.12 Modèle structuré (Structured Template)

Modèle structuré (Structured Template)				
Nom	**:**	*TemplateIdentifier&ParList*		
Groupe	**:**	*[GroupReference]*		
Type	**:**	*TemplateStructIdentifier*		
Codage	**:**	*[Texte libre]*		
Commentaires	**:**	*[Texte libre]*		
Nom de l'élément		**Valeur de l'élément**	**Codage de l'élément**	**Commentaires**
. . *FieldReference* . .		. . *FieldValueOrAttrib* .	. . . *[Texte libre]*	. . *[Texte libre]* . .
Commentaires détaillés:		*[Texte libre]*		

Figure 13/Z.141 – Formulaire de modèle structuré

7.12.1 Mappage

Le formulaire de modèle structuré est mappé en une déclaration de définition de formulaire en TTCN-3, les champs groupes et commentaires étant mappés en déclaration de présentation dans la déclaration **with** correspondante.

```
template [Type | Signature] TemplateIdentifier&ParList [modifies TemplateRef] :=
{
        FieldReference ":=" FieldValueOrAttrib
}
with {   display "StructutedTemplateProforma";
         encode "FreeText";
         encode (FieldReference)"FreeText";
    }
```

Exemple:

Modèle structure (Structured Template)				
Nom	**:**	Setup01		
Groupe	**:**			
Type	**:**	SetupMsgType		
Trajet de dérivation	**:**			
Codage	**:**			
Commentaires	**:**			
Nom d'élément		**Valeur d'élément**	**Codage d'élément**	**Commentaires**
MsgId		34		
CrLength		1		
CrValue		42		
IE1		?		
IE2		?		
Commentaires détaillés:				

Se mappe en:

```
template SetupMsgType Setup01 :=
{
        MsgId :=      34,
        CrLength :=   1,
        CrValue   :=  42,
        IE1 :=        ?,
        IE2 :=        ?
}
with display "structuredtemplate {}";
```

7.13 Fonction (function)

Fonction (Function)				
Nom	:	*FunctionIdentifier&ParList*		
Groupe	:	*[GroupReference]*		
Opère sur	:	*[ComponentType]*		
Type de retour	:	*[Type]*		
Commentaires	:	*[Texte libre]*		
Nom		**Type**	**Valeur initiale**	**Commentaires**
. *[VarConstOrTimerIdentifier]* . .		. *[TypeOrTimer]* . .	. *[ConstantExpression]* . .	. *[Texte libre]* . .
Définition de la fonction				**Commentaires**
TabularBehaviour . .				*[Texte libre].* . . .
Commentaires détaillés: *[Texte libre]*				

Figure 14/Z.141 – Formulaire fonction

7.13.1 Mappage

Le formulaire fonction est mappé en une définition de fonction ou de fonction externe en TTCN-3, dont les champs groupes et commentaires sont mappés en déclaration de présentation dans la déclaration **with** correspondante. Le mot clé **external** avant le nom de fonction indique que la fonction doit être mappée en une fonction externe.

```
[external] function FunctionIdentifier&ParList [return Type]
[Runs On ComponentType]
{
     var Type VarIdentifier [:= ConstantExpression] ;
     timer TimerIdentifier [:= ConstantExpression] ;
     const  Type ConstIdentifier  := ConstantExpression

     TabularBehaviour
}
with display "FunctionProforma";
```

Exemple:

Fonction (Function)			
Nom : MyFunction(**in integer** Par1) **Groupe** : **Opère sur** : **Type de retour** : **boolean** **Commentaires** :			
Nom	**Type**	**Valeur initiale**	**Commentaires**
MyLocalVar T1	**boolean** **timer**	**false** 15 min	
Définition de la fonction			**Commentaires**

if(Par1 = 21) {
  MyLocalVar := **true**;
}
if(MyLocalVar) {
  T1.**start**;
  T1.**timeout**;
}
return(MyLocalVar);			

Commentaires détaillés:			

Se mappe en

```
function MyFunction( in integer Par1) return boolean
{
          var boolean MyLocalVar := false;
          timer T1 := 15 min;

          if(  Par1 = 21 ) {
               MyLocalVar := true;
          }
          if(  MyLocalVar) {
               T1.start;
               T1.timeout;
          }
          return( MyLocalVar);
}
with display "function";
```

7.14 Valeurs par défaut (default)

Définition de valeur par défaut (Default Definition)		
Nom	:	*NamedAltIdentifier&ParList*
Groupe	:	*[GroupReference]*
Objet	:	*[Texte libre]*
Commentaires	:	*[Texte libre]*
Comportement		**Commentaires**
AltGuardList . ..		*[Texte libre].* . . . .
Commentaires détaillés:	*[Texte libre]*	

Figure 15/Z.141 – Formulaire de définition de valeur par défaut

7.14.1 Mappage

Le formulaire de valeur par défaut est mappé en une définition alt dans le module TTCN-3. Les champs groupes et commentaires sont mappés en déclaration de présentation intégrée à la déclaration **with** associée à la définition.

```
named alt NamedAltIdentifier&ParList{
          AltGuardList
}
with display "default {purpose := "";}";
```

Exemple:

Définition de valeur par défaut (Default Definition)		
Nom	:	Default1
Groupe	:	
Objet	:	
Commentaires	:	
Comportement		**Commentaires**
[] PCO2.**receive**(DL_EST_IN); PCO2.**send**(DL_EST_CO); [] PCO2.**receive**(DL_EST_CO); // do nothing		
Commentaires détaillés:		

Se mappe en:

```
named alt Default1{
     []   PCO2.receive( DL_EST_IN){
          PCO2.send( DL_EST_CO)}
     []   PCO2.receive( DL_EST_CO);
}
with display "default{ purpose := "";}";
```

7.15 Alternative nommée (named alternative)

Définition d'une alternative nommée (Named Alternative Definition)		
Nom	:	*NamedAltIdentifier&ParList*
Groupe	:	*[GroupReference]*
Objet	:	*[Texte libre]*
Commentaires	:	*[Texte libre]*
Comportement		**Commentaires**
AltGuardList . ..		. . . . .
Commentaires détaillés:	*[Texte libre]*	

Figure 16/Z.141 – Formulaire de définition d'alternative nommée

7.15.1 Mappage

Le formulaire d'alternative nommée est mappé en une définition alt nommée dans le module TTCN-3. Les champs groupes et commentaires sont mappés en déclaration de présentation intégrée dans la déclaration **with** associée à la définition

```
named alt NamedAltIdentifier&ParList{
          AltGuardList
}
with display "NamedAltProforma ";
```

Exemple:

Définition d'une alternative nommée (Named Alternative Definition)		
Nom	:	TS01
Groupe	:	
Objet	:	
Commentaires	:	
Comportement		**Commentaires**
[] PCO2.**receive**(DL_EST_IN); PCO2.**send**(DL_EST_CO); [] PCO2.**receive**(DL_EST_CO); // do nothing		
Commentaires détaillés:		

Se mappe en:

```
named alt Default1{
     []   PCO2.receive( DL_EST_IN){
          PCO2.send( DL_EST_CO)}
     []   PCO2.receive( DL_EST_CO);
}
with display "namedalt{ purpose := ""; }";
}
```

7.16 Test élémentaire (testcase)

Définition d'un test élémentaire (Test Case Definition)			
Nom : *TestcaseIdentifier&ParList*			
Groupe : *[GroupReference]*			
Objet : *[Texte libre]*			
Interface système : *[ComponentType]*			
Type MCT : *ComponentType*			
Commentaires : *[Texte libre]*			
Nom	**Type**	**Valeur initiale**	**Commentaires**
. *[VarConstOrTimerIdentifier]* . .	. *[TypeOrTimer]* . .	. *[ConstantExpression]* . .	. *[Texte libre]* . .
Comportement			**Commentaires**
TabularBehaviour . ..			. . . . .
Commentaires détaillés: *[Texte libre]*			

Figure 17/Z.141 – Formulaire de définition d'un test élémentaire (Testcase)

7.16.1 Mappage

Le formulaire de test élémentaire est mappé en une définition de test élémentaire en TTCN-3, dont les champs groupes et commentaires sont mappés en déclaration de présentation dans la déclaration **with** correspondante. La liste des paramètres peut seulement contenir que des variables de la suite de test.

```
testcase TestcaseIdentifier&ParList
runs on ComponentType[system ComponentType]
{
         var Type VarIdentifier [":=" ConstantExpression] ;
         timer TimerIdentifier [":=" ConstantExpression] ;
         const Type ConstIdentifier ":=" ConstantExpression ;

         TabularBehaviour

}
with display "TestcaseProforma";
```

Exemple:

Définition d'un test élémentaire (Test Case Definition)

Nom	:	MyTestcase
Groupe	:	
Objet	:	First Example Testcase
Interface système	:	
Type MCT	:	MyComponentType
Commentaires	:	

Noms	Type	Valeur initiale	Commentaires
MyLocalVar	**Integer**	0	
TimerT1	**Timer**	15 min	

Comportement — **Commentaires**

```
default.activate { [expand] OtherwiseFail(); }; /* Default activation */
ISAP1.send( ICONreq {} ); /* Inline template definition */
alt {
        []      MSAP2.receive( Medium_Connection_Request() ); { /* use of a template */
                        MSAP2.send( MDATreq Medium_Connection_Confirmation() );
                        alt {
                            []  ISAP1.receive ( ICONconf {} ); {
                                   ISAP1.send ( Data_Request(TestSuitePar) );
                                   alt {
                                       []  MSAP2.receive (
                                                          Medium_Data_Transfer()
); {
                                          MSAP2.send ( MDATreq
cmi_synch1() );
                                          ISAP1.send ( IDISreq {} );
                                       }
                                   []      ISAP1.receive( IDISind {} ); {
                                           verdict.set(inconclusive);
                                            stop();
                                           }
                                   };
                                }
                            []  MSAP2.receive( MDATind_Connection_Request()); {
                                    verdict.set(inconclusive);
                                    stop();
                                }
                            []  ISAP1.receive( IDISind {} ); {
                                   verdict.set(inconclusive);
                                   stop();
                                }
                        };
          }
        []      ISAP1.receive( IDISind {} ); {
                        verdict.set(inconclusive);
                        stop();
                }
}
```

Commentaires détaillés:

Se mappe en:

```
testcase MyTestcase
runs on MyComponentType
{
          var integer MyLocalVar:= 0;
          timer T1 := 15 min;

          default.activate { [expand] OtherwiseFail(); }; /* Default activation
          */

          ISAP1.send( ICONreq {} ); /* Inline template definition */
          alt {
          []   MSAP2.receive( Medium_Connection_Request() ); { /* use of a
               template */
               MSAP2.send( MDATreq Medium_Connection_Confirmation() );
               alt {
               []   ISAP1.receive ( ICONconf {} ); {
                    ISAP1.send ( Data_Request(TestSuitePar) );
                    alt {
                    []   MSAP2.receive ( Medium_Data_Transfer() ); {
                                          MSAP2.send ( MDATreq cmi_synch1() );
                                          ISAP1.send ( IDISreq {} );
                                        }
                    []   ISAP1.receive( IDISind {} ); {
                         verdict.set(inconclusive);
                                               stop();
                                        }
                              };
                         }
                    []   MSAP2.receive( MDATindConnection_Request()); {
                              verdict.set(inconclusive);
                              stop();
                         }
                    []   ISAP1.receive( IDISind {} ); {
                              verdict.set(inconclusive);
                              stop();
                         }
               };
          }
          []   ISAP1.receive( IDISind {} ); {
               verdict.set(inconclusive);
               stop();
          }
}
}
with display "testcase { purpose "";}";
}
```

8 Représentation en BNF du format de présentation tabulaire

```
TabularPresentationFormat ::=    ReferenceProforma |
                                 ParametersProforma |
                                 ControlProforma |
                                 ImportsProforma |
                                 EncodingProforma |
                                 SimpleTypesProforma |
                                 StructuredTypesProforma |
                                 PortTypeProforma |
                                 ComponentTypeProforma |
                                 ConstantsProforma |
                                 SignatureProforma |
```

```
                                SimpleTemplatesProforma |
                                StructuredTemplatesProforma |
                                FunctionProforma |
                                DefaultsProforma |
                                TeststepProforma |
                                TestcaseProforma
```

8.1 Formulaire de référence (ReferenceProforma)

```
ReferenceProforma ::=   ReferenceKeyword
                        BeginChar
                        ReferenceFieldList
                        EndChar

ReferenceFieldList ::= VersionKeyword AssignmentChar VersionIdentifier  SemiColon
                       DateKeyword    AssignmentChar FreeText           SemiColon
                       BaseKeyword    AssignmentChar FreeText           SemiColon
                       TestKeyword    AssignmentChar FreeText           SemiColon
                       PicsKeyword    AssignmentChar FreeText           SemiColon
                       PixitKeyword   AssignmentChar FreeText           SemiColon
                       MethodKeyword  AssignmentChar FreeText           SemiColon
                       [DetailedComments]

VersionIdentifier ::= Number { "." Number }

VersionKeyword ::=      version
DateKeyword ::=         date
BaseKeyword ::=         basestandard
TestKeyword ::=         teststandard
PicsKeyword ::=         pics
PixitKeyword ::=        pixit
MethodKeyword ::=       testmethod
DCommentsKeyword ::=    detailedcomments

DetailedComments ::= DCommentsKeyword     AssignmentChar  FreeText
```

8.2 Formulaire de paramètres (ParametersProforma)

```
ParametersProforma ::=  ParametersKeyword BeginChar
                        ParametersFieldList
                        EndChar

ParameterKeyword ::= parameters

ParameterFieldList ::=  [GroupDef]
                        PICSRefList
                        [ParameterCommentsList]
                        [DetailedComments]

GroupDef ::= GroupKeyword             AssignmentChar  GroupReference    SemiColon
GroupKeyword ::= group
CommentListKeyword ::= commentlist

PICSRefList ::= PicsKeyword PixitKeyword  AssignmentChar   BeginChar
                [PicsRef {"," PicsRef}]
                EndChar [SemiColon]

PicsRef ::= ModuleParIdentifier AssignmentChar FreeText

ParameterCommentsList ::=   CommentListKeyword       AssignmentChar    BeginChar
                            [ParComment {"," ParComment}]
                            EndChar [SemiColon]

ParComment ::= ModuleParIdentifier AssignmentChar FreeText
```

8.3 Formulaire de commande (ControlProforma)

```
ControlProforma ::= ControlKeyword BeginChar ControlFieldList EndChar

ControlFieldList ::= [VarConstOrTimerCommentList]
                     [ExtraComments]
                     [DetailedComments]
```

8.4 Formulaire d'import (ImportsProforma)

```
ImportsProforma ::= ImportsKeyword BeginChar ImportsFieldList EndChar

ImportsKeyword ::= imports

ImportType ::= AllKeyword [DefKeyword] | DefKeyword
ImportIdentifier ::= [TypeDefIdentifier | TemplateIdentifier | ConstIdentifier |
                     TestcaseIdentifier | FunctionIdentifier |
                     NamedAltIdentifier]

Mark ::= ["*"]

ImportsFieldList ::= [GroupDef]
                     SourceKeyword AssignmentChar FreeText SemiColon
                     [SingleComment]
                     [ExtraCommentList]
                     [DetailedComments]

SourceKeyword ::= source

SingleComment ::= CommentsKeyword AssignmentChar FreeText SemiColon
CommentsKeyword ::= comments
```

8.5 Formulaire de codage (EncodingProforma)

```
EncodingProforma ::= EncodingKeyword BeginChar EncodingFieldList EndChar
EncodingKeyword ::= encoding

EncodingRuleIdentifier ::= identifier

EncodingFieldList ::=   [GroupDef]
                        EncodingRefList
                        EncodingDefaultList
                        [EncodingCommentList]
                        [DetailedComments]

RefKeyword ::= reference
DefaultKeyword ::= default

EncodingRefList ::= RefKeyword AssignmentChar  BeginChar
                    [EncodingRef {"," EncodingRef}]
                    EndChar [SemiColon]

EncodingRef ::= EncodingRuleIdentifier AssignmentChar FreeText

EncodingDefaultList ::= DefaultKeyword AssignmentChar   BeginChar
                        [EncodingDefault {"," EncodingDefault}]
                        EndChar [SemiColon]]
```

```
EncodingDefault ::= EncodingRuleIdentifier AssignmentChar BooleanExpression

EncodingCommentList ::= CommentListKeyword AssignmentChar BeginChar
                        [EncodingComment {"," EncodingComment}]
                        EndChar [SemiColon]

EncodingComment ::= EncodingRuleIdentifier AssignmentChar FreeText
```

8.6 Formulaire de types simple (SimpleTypesProforma)

```
SimpleTypesProforma ::= SimpleTypeKeyword BeginChar SimpleTypeFieldList EndChar
SimpleTypesKeyword ::= simpletypes

SimpleTypeFieldList ::=     [GroupDef]
                            [SimpleTypeCommentList]
                            [DetailedComments]

SimpleTypeCommentList ::=   CommentListKeyword AssignmentChar BeginChar
                            [SimpleTypeComment {"," SimpleTypeComment}]
                            EndChar [SemiColon]

SimpleTypeComment ::= SubTypeIdentifier AssignmentChar FreeText
```

8.7 Formulaire de type structuré (StructuredTypesProforma)

```
StructuredTypeProforma ::=  StructTypeKeyword BeginChar StructTypeFieldList
                            EndChar
StructTypeKeyword ::= structuredtype

StructTypeFieldList ::=     [GroupDef]
                            [SingleComment]
                            [StructTypeCommentList]
                            [DetailedComments]

StructTypeCommentList ::=   CommentListKeyword AssignmentChar BeginChar
                            [StructTypeComment {"," StructTypeComment}]
                            EndChar [SemiColon]

StructTypeComment ::= StructFieldIdentifier AssignmentChar FreeText

StructureType ::=  record | set | union
```

8.8 Formulaire de type de port (PortTypeProforma)

```
PortTypeProforma ::= PortTypeKeyword BeginChar PortTypeFieldList EndChar
PortTypeKeyword ::= porttype

PortTypeFieldList ::=   [GroupDef]
                        [SingleComment]
                        [PortTypeCommentList]
                        [DetailedComments]

PortTypeCommentList ::= CommentListKeyword AssignmentChar BeginChar
                        [PortTypeComment {"," PortTypeComment}]
                        EndChar [SemiColon]

PortTypeComment ::= PortTypeIdentifier AssignmentChar FreeText

PortModelType ::= MessageKeyword | ProcedureKeyword | MixedKeyword
PortTypeDef ::= BeginChar MixedList {SemiColon MixedList} [SemiColon] EndChar
```

8.9 Formulaire de type de composante (ComponentTypeProforma)

```
ComponentTypeProforma ::= ComponentTypeKeyword BeginChar ComponentTypeFieldList
                          EndChar
ComponentTypeKeyword ::= componenttype

ComponentTypeFieldList ::=  [GroupDef]
                            [SingleComment]
                            [VarConstOrTimerCommentList]
                            [ExtraComments]
                            [DetailedComments]

ExtraComments ::= ECommentsKeyword AssignmentChar FreeText SemiColon
ECommentsKeyword ::= extracomments

PortList ::= {PortInstance}

TypeOrTimer ::= Type | TimerKeyword
```

8.10 Formulaire de constante (ConstantsProforma)

```
ConstantsProforma ::= ConstantsKeyword BeginChar ConstantsFieldList EndChar
ConstantsKeyword ::= constants

ConstantsFieldList ::=  [GroupDef]
                        [ConstantsCommentList]
                        [DetailedComments]

ConstantsCommentList ::=    CommentListKeyword AssignmentChar BeginChar
                            [ConstantsComment {"," ConstantsComment}]
                            EndChar [SemiColon]

ConstantsComment ::= ConstIdentifier AssignmentChar FreeText
```

8.11 Formulaire de signature (SignatureProforma)

```
SignatureProforma ::= SignatureKeyword BeginChar SignatureFieldList EndChar

SignatureFieldList ::=  [GroupDef]
                        [SingleComment]
                        [ExtraComments]
                        [DetailedComments]

SignatureIdentifer&ParList ::=   SignatureIdentifier "("
                                 [SignatureFormalParList] ")"
```

8.12 Formulaire de modèle simple (SimpleTemplatesProforma)

```
SimpleTemplatesProforma ::= SimpleTemplatesKeyword
BeginCharSimpleTemplatesFieldList EndChar
SimpleTemplatesKeyword ::= simpleTemplates

SimpleTypeFieldList ::=     [GroupDef]
                            [SimpleTemplatesCommentList]
                            [DetailedComments]

SimpleTemplatesCommentList ::=   CommentListKeyword AssignmentChar BeginChar
                                 [SimpleTemplateComment {","
                                 SimpleTemplateComment}]
                                 EndChar [SemiColon]
```

```
SimpleTemplateComment ::= TemplateIdentifier AssignmentChar FreeText

SimpleType ::= Type | DerivedDef
/* STATIC SEMANTICS – The referenced type or base template shall not be of a
constructed type */
```

8.13 Formulaire de modèle structuré (StructuredTemplatesProforma)

```
StructuredTemplateProforma ::= StructuredTemplateKeyword BeginChar
TemplateFieldList EndChar
StructuredTemplateKeyword ::= structuredtemplate

TemplateFieldList ::=   [GroupDef]
                        [SingleComment]
                        [TemplateCommentList]
                        [DetailedComments]

TemplateCommentList ::=     CommentListKeyword AssignmentChar BeginChar
                            [TemplateComment {"," TemplateComment}]
                            EndChar [SemiColon]

TemplateComment ::= FieldReference AssignmentChar FreeText

TemplateIdentifier&ParList ::= TemplateIdentifier ["(" TemplateFormalParList")" ]
TemplateStructIdentifier ::= Type | Signature | DerivedDef
```

8.14 Formulaire de fonction (FunctionProforma)

```
FunctionProforma ::= FunctionKeyword BeginChar FunctionFieldList EndChar

FunctionFieldList ::=   [GroupDef ]
                        [SingleComment]
                        [VarConstOrTimerCommentList]
                        [ExtraComments]
                        [DetailedComments]

FunctionIdentifier&ParList ::= FunctionIdentifier "(" [FunctionFormalParList]")"

FunctionCommentList ::= FunctionComment {"," FunctionComment}
FunctionComment ::= VarConstOrTimerRef AssignmentChar FreeText

VarConstOrTimerCommentList ::= CommentListKeyword AssignmentChar  BeginChar
                                                                  [FunctionCommentList]
                                                                  EndChar [SemiColon]

VarConstOrTimerIdentifier ::=    ConstKeyword ConstIdentifier |
                                 VarIdentifier [ArraySpec] |
                                 TimerIdentifier [ArraySpec]

VarConstOrTimerRef ::=           ConstIdentifier | VarIdentifier |
                                 TimerIdentifier

TabularBehaviour ::= FunctionBody
/* STATIC SEMANTICS – The FunctionBody production shall not contain any variable,
timer or constant definitions */
```

8.15 Formulaire de valeurs par défaut (DefaultsProforma)

```
DefaultProforma ::= DefaultKeyword BeginChar DefaultFieldList EndChar
DefaultKeyword ::= default

DefaultFieldList ::=    [GroupDef]
                        PurposeDef
                        [SingleComment]
                        [DetailedComments]

PurposeDef ::= PurposeKeyword AssignmentChar FreeText SemiColon
PurposeKeyword ::= purpose
NamedAltIdentifier&ParList ::= NamedAltIdentifier ["(" FunctionFormalParList")"]
```

8.16 Formulaire d'alternative nommé (NamedAltProforma)

```
NamedAltProforma ::= NamedAltKeyword BeginChar DefaultFieldList EndChar
NamedAltKeyword ::= namedalt
```

8.17 Formulaire de test élémentaire (TestcaseProforma)

```
TestcaseProforma ::= TestcaseKeyword BeginChar TestcaseFieldList EndChar

TestcaseFieldList ::=       [GroupDef]
                            PurposeDef
                            [SingleComment]
                            [VarOrTimerCommentList]
                            [ExtraComments]
                            [DetailedComments]


TestcaseIdentifier&ParList ::= TestcaseIdentifier "("
[TestcaseRestrictedFormalParList]")"
TestcaseRestrictedFormalParList ::= FormalVarValuePar {"," FormalVarValuePar}
```

SÉRIES DES RECOMMANDATIONS UIT-T

Série A	Organisation du travail de l'UIT-T
Série B	Moyens d'expression: définitions, symboles, classification
Série C	Statistiques générales des télécommunications
Série D	Principes généraux de tarification
Série E	Exploitation générale du réseau, service téléphonique, exploitation des services et facteurs humains
Série F	Services de télécommunication non téléphoniques
Série G	Systèmes et supports de transmission, systèmes et réseaux numériques
Série H	Systèmes audiovisuels et multimédias
Série I	Réseau numérique à intégration de services
Série J	Réseaux câblés et transmission des signaux radiophoniques, télévisuels et autres signaux multimédias
Série K	Protection contre les perturbations
Série L	Construction, installation et protection des câbles et autres éléments des installations extérieures
Série M	RGT et maintenance des réseaux: systèmes de transmission, circuits téléphoniques, télégraphie, télécopie et circuits loués internationaux
Série N	Maintenance: circuits internationaux de transmission radiophonique et télévisuelle
Série O	Spécifications des appareils de mesure
Série P	Qualité de transmission téléphonique, installations téléphoniques et réseaux locaux
Série Q	Commutation et signalisation
Série R	Transmission télégraphique
Série S	Equipements terminaux de télégraphie
Série T	Terminaux des services télématiques
Série U	Commutation télégraphique
Série V	Communications de données sur le réseau téléphonique
Série X	Réseaux de données et communication entre systèmes ouverts
Série Y	Infrastructure mondiale de l'information et protocole Internet
Série Z	**Langages et aspects généraux logiciels des systèmes de télécommunication**

20815

Imprimé en Suisse
Genève, 2001