Union internationale des télécommunications

UIT-T Z.141

(03/2006)

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 de test et de commande de test

Notation de test et de commande de test version 3 (TTCN-3): format de présentation tabulaire

Recommandation UIT-T Z.141

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
Langage étendu de définition d'objets	Z.130–Z.139
Notation de test et de commande de test	**Z.140–Z.149**
Notation de prescriptions d'utilisateur	Z.150–Z.159
LANGAGES DE PROGRAMMATION	
CHILL: le langage de haut niveau 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.349
Interfaces homme-machine orientées données	Z.350–Z.359
Interfaces homme-machine pour la gestion des réseaux de télécommunication	Z.360–Z.379
QUALITÉ	
Qualité des logiciels de télécommunication	Z.400–Z.409
Aspects qualité des Recommandations relatives aux protocoles	Z.450–Z.459
MÉTHODES	
Méthodes de validation et d'essai	Z.500–Z.519
INTERGICIELS	
Environnement de traitement réparti	Z.600–Z.609

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

Recommandation UIT-T Z.141

Notation de test et de commande de test version 3 (TTCN-3): format de présentation tabulaire

Résumé

La présente Recommandation définit le format de présentation tabulaire pour la notation de test et de commande de test version 3 (TTCN-3, *testing and test control notation version 3*) (TFT). C'est le format employé pour le langage noyau de la notation TTCN-3, défini dans la Rec. UIT-T Z.140. Il est semblable, de par son aspect et ses fonctionnalités, à celui en notation TTCN-2, qui est défini dans la Rec. UIT-T X.292 pour les tests de conformité. Il permet de présenter différemment le langage noyau en mettant l'accent sur les aspects propres aux spécifications d'une suite de tests normalisés de la conformité. Tandis que le langage noyau peut être employé indépendamment du format de présentation tabulaire, celui-ci ne peut être utilisé sans le langage noyau. L'emploi et l'implémentation du format de présentation tabulaire doivent se faire en fonction du langage noyau. La présente Recommandation définit les formulaires, les mappages syntaxiques, la sémantique statique additionnelle, les restrictions de sémantique opérationnelle, la présentation et autres attributs. L'ensemble de ces caractéristiques constitue le format de présentation tabulaire.

Le format TFT, qui hérite de toutes les propriétés essentielles du langage noyau, est destiné aux spécifications des suites de tests qui sont indépendants des plates-formes, des méthodes de test, des couches de protocole et des protocoles. La notation TTCN-3 peut être employée pour spécifier tous les types de tests réactifs des systèmes sur une gamme de ports de communication. Parmi les domaines d'application types, on peut citer les tests de protocoles (y compris les protocoles régissant les télécommunications mobiles et Internet), les tests de services (y compris les services complémentaires), les tests de modules, les tests de plates-formes fondées sur l'architecture de courtier commun de requête sur des objets (CORBA, *common object request broker architecture*) et les tests d'interfaces de programmation d'application (API, *application programming interface*). La spécification des suites de tests pour les protocoles de la couche Physique sort du cadre de la présente Recommandation.

Source

La Recommandation UIT-T Z.141 a été approuvée le 16 mars 2006 par la Commission d'études 17 (2005-2008) de l'UIT-T selon la procédure définie dans la Recommandation UIT-T A.8.

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.

Le respect de cette Recommandation se fait à titre volontaire. Cependant, il se peut que la Recommandation contienne certaines dispositions obligatoires (pour assurer, par exemple, l'interopérabilité et l'applicabilité) et considère que la Recommandation est respectée lorsque toutes ces dispositions sont observées. Le futur d'obligation et les autres moyens d'expression de l'obligation comme le verbe "devoir" ainsi que leurs formes négatives servent à énoncer des prescriptions. L'utilisation de ces formes ne signifie pas qu'il est obligatoire de respecter la Recommandation.

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 de test et de commande de test version 3 (TTCN-3): format de présentation tabulaire

1 Domaine d'application

La présente Recommandation définit le format de présentation tabulaire de la notation de test et de commande de test version 3 (TTCN-3, *testing and test control notation version 3*). La présente Recommandation est fondée sur le langage noyau TTCN-3 défini dans la Rec 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. La référence à un document figurant dans la présente Recommandation ne donne pas à ce document, en tant que tel, le statut d'une Recommandation.

[1] Recommandation UIT-T Z.140 (2006), *Notation de test et de commande de test version 3 (TTCN-3): langage noyau*.

[2] Recommandation UIT-T Z.143 (2006), *Notation de test et de commande de test version 3 (TTCN-3): sémantique opérationnelle*.

3 Abréviations

Dans la présente Recommandation, les abréviations suivantes s'appliquent:

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

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

BNF formalisme de Backus-Naur (*Backus-Naur form*)

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

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

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

TFT format de présentation tabulaire de la notation TTCN-3 (*tabular presentation format for TTCN-3*)

TTCN notation de test et de commande de test (*testing and test control notation*)

4 Introduction

Le format de présentation tabulaire de la notation de test et de commande de test version 3 (TTCN-3, *testing and test control notation version 3*) (TFT) 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 la Rec. UIT-T Z.140 [1] et spécifie une syntaxe en texte intégral, une sémantique statique ainsi que l'utilisation de cette notation avec l'ASN.1. La sémantique opérationnelle est définie dans la Rec. UIT-T Z.143 [2]. 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.

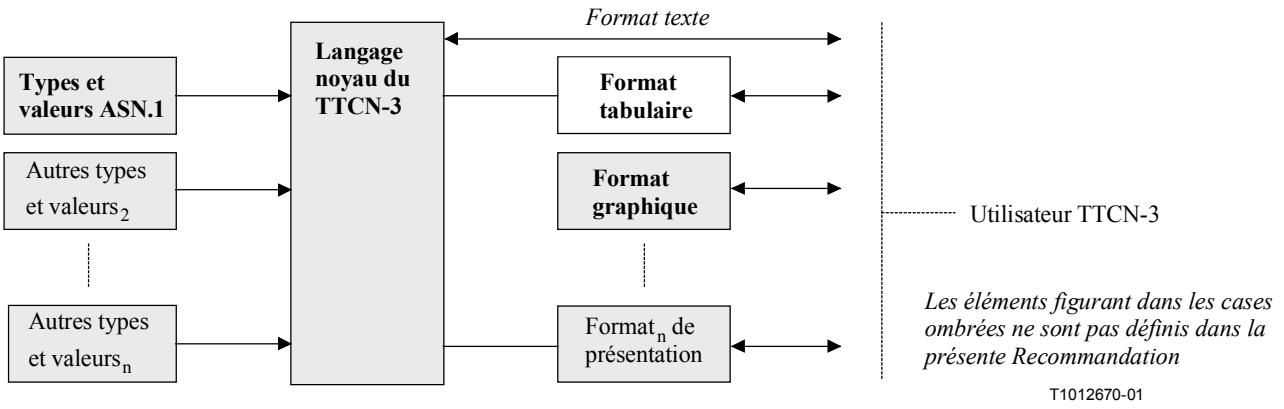

Figure 1/Z.141 – Vue utilisateur du langage noyau et des divers formats de présentation

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 dans l'Annexe A/Z.140 [1].

5.2 Texte de spécification

a) Un **texte en caractères gras** doit être employé pour les références aux champs des formulaires.

b) Un *texte en caractères italiques* doit être employé pour les références aux productions BNF du langage noyau en notation TTCN-3.

c) Un **`texte en caractères gras courier new`** doit être employé pour les mots-clés du langage noyau.

5.3 Formulaire

a) Un **`texte en caractères gras`** doit apparaître dans son intégralité dans chaque tableau rempli d'un module TTCN-3.

b) Un *`texte en caractères italiques`* ne doit pas apparaître dans son intégralité dans un module TTCN-3. 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-3. Les crochets entourant le *`texte italique`* indiquent que l'insertion du texte dans le champ donné du formulaire est facultative.

5.4 Langage noyau

a) Un **`texte en caractères gras`** placé entre guillemets simples (par exemple '{') est employé pour les mots-clés et les terminaux réservés en langage noyau.

b) Un *`texte en caractères italiques`* ne doit pas apparaître dans son intégralité dans un module TTCN-3. 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-3.

c) La notation "`...`" correspond à un contenu arbitraire quelconque qui n'est pas explicitement indiqué.

5.5 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. De telles transformations permettent aussi de convertir un quelconque module de langage noyau en une représentation tabulaire.

Ces transformations relèvent de deux classes. La première assure directement la conversion entre un élément tabulaire et une structure de langage noyau de même signification. La deuxième assure la conversion entre un élément tabulaire et 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 le même élément 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 *WithStatement*.

La syntaxe et la sémantique spécifiées dans la présente Recommandation sont propres au format de présentation tabulaire 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:

```
1:  module TTCN3ModuleId    "{"
2:      ...
3:  "}" with "{"
4:      display """ "presentation format" ":=" "ETSI Tabular version"
5:      MajorVersion "." MinorVersion """ ";"
6:      ...
7:  "}"
```

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

Les champs **Groupe** dans les formulaires ne sont jamais convertis en déclarations *WithStatements* mais sont déduits de la structure de groupe effective de la spécification du module.

6 Formulaires

6.1 Commande de la suite de tests

Commande de la suite de tests			
Nom du module	*TTCN3ModuleId*		
Version	*[TabFreeText]*		
Date	*[TabFreeText]*		
Réf. normes de base	*[TabFreeText]*		
Réf. normes de test	*[TabFreeText]*		
Réf. PICS	*[TabFreeText]*		
Réf. PIXIT	*[TabFreeText]*		
Méthode(s) de test	*[TabFreeText]*		
Codage	*[TabFreeText]*		
Commentaires	*[TabFreeText]*		
Nom de la définition locale	**Type**	**Valeur initiale**	**Commentaires**
[VarConstOrTimerIdentifier]	*[ConstTypeOrTimer]*	*[Expression]*	*[TabFreeText]*
...	...	...	...
Comportement			
ModuleControlBody			
Commentaires détaillés	*[TabFreeText]*		

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

6.1.1 Mappage

Le formulaire de commande de la suite de tests est converti en trois parties. La première partie comporte les champs d'en-tête et le champ **Commentaires détaillés**, qui sont convertis en des attributs d'affichage dans la déclaration *WithStatement* associée au module global TTCN-3. Le champ **Nom du module** est mappé en l'identificateur du module.

La deuxième partie comporte des constantes, des variables et des temporisateurs locaux définis dans la partie de commande. Ces définitions peuvent être présentes partout dans la partie de commande du langage noyau, mais, dans le formulaire, elles doivent être séparées du reste du corps des modules de commande et être affichées dans un tableau distinct. Il faut conserver l'ordre des définitions, puisqu'elles peuvent dépendre les unes des autres. La colonne **Type** doit être affectée du mot clé **timer** pour tous les temporisateurs et du type constant précédé du mot clé **const** pour toutes les constantes. Les champs **Commentaires** du tableau des définitions locales sont convertis en des attributs d'affichage dans la déclaration *WithStatement* associée à la partie de commande du module de langage noyau TTCN-3.

La troisième partie est la partie de commande du module de langage noyau TTCN-3, sans constantes, variables et temporisateurs locaux.

```
 1:  module TTCN3ModuleId "{"
 2:      control "{"
 3:   var Type VarIdentifier [":=" Expression] ";"
 4:   timer TimerIdentifier [":=" Expression] ";"
 5:   const Type ConstIdentifier ":=" ConstantExpression;
 6:          ModuleControlBody
 7:      "}" with "{"
 8:          { VarConstOrTimerCommentsAttribute }
 9:      "}"
10:  "}" with "{"
11:      ModuleAttributes
12:   [EncodeAttribute;]
13:  "}"
```

EXEMPLE:

Commande de la suite de tests			
Nom du module	Example1		
Version	1.01		
Date	19 July 2001		
Réf. normes de base	ITU-T Recommendation Q.123		
Réf. normes de test	ITU-T Recommendation Q.123.1		
Réf. PICS	ITU-T Recommendation Q.123.2, Annex A		
Réf. PIXIT	ITU-T Recommendation Q.123.2, Annex B		
Méthode(s) de test	local		
Codage	BER		
Commentaires	ATS written by STF 133		
Nom de la définition locale	**Type**	**Valeur initiale**	**Commentaires**
PI	const float	3.14	the ratio
x	float	PI * 2	double PI
t1	timer	15	a 15 second timer
Comportement			

```
/* group1/ */
   /* group1_1/ */
       execute(test1);
       execute(test2);
   /* group1_2/ */
       execute(test3);
       execute(test4);
/* group2/ */
     execute(test5);
```

Commentaires détaillés	detailed comments

Se mappe en:

```
  1:  module Example1 {
  2:      control {
  3:          const float PI := 3.14;
  4:          var float x := PI * 2;
  5:          timer t1 := 15;
  6:
  7:          /* group1/ */
  8:              /* group1_1/ */
  9:                  execute(test1());
 10:                  execute(test2());
 11:              /* group1_2/ */
 12:                  execute(test3());
 13:                  execute(test4());
 14:              /* group2/ */
 15:                  execute(test5());
 16:      } with {
 17:          display (PI) "comments := the ratio";
 18:          display (x) "comments := double PI";
 19:          display (t1) "comments := a 15 second timer";
 20:      }
 21:  } with {
 22:      display "presentation format := ETSI Tabular version 1.0";
 23:      display "module version := 1.01";
 24:      display "module date := 19 July 2001";
 25:      display "module base standards ref := ITU-T Recommendation Q.123";
 26:      display "module test standards ref := ITU-T Recommendation Q.123";
 27:      display "module pics ref := ITU-T Recommendation Q.123, Annex A";
 28:      display "module pixit ref := ITU-T Recommendation Q.123, Annex A";
 29:      display "module test method := local";
 30:      display "module comments := ATS written by STF 133";
 31:      display "module detailed comments := detailed comments";
 32:      encode "BER";
 33:  }
```

6.2 Paramètres de la suite de tests

Paramètres de la suite de tests				
Nom	**Type**	**Valeur initiale**	**Ref. PICS/PIXIT**	**Commentaires**
. *ModuleParIdentifier* .	. *ModuleParType* .	. *[ConstantExpression]* .	. *[TabFreeText]* .	. *[TabFreeText]* .
Commentaires détaillés	*[TabFreeText]*			

Figure 3/Z.141 – Formulaire des paramètres de la suite de tests

6.2.1 Mappage

Toutes les entrées du formulaire des paramètres de la suite de tests sont mappées en les listes *ModuleParList*s dans les définitions *ModuleParameterDefs* du module TTCN-3 associé. S'il existe plus d'une définition *ModuleParameterDef*, toutes les listes *ModuleParList*s sont collectées et représentées dans un unique formulaire **Paramètres de la suite de tests**.

Les champs Réf. **PICS/PIXIT** et **Commentaires** sont mappés en des attributs d'affichage qualifiés par l'identificateur du paramètre dans la déclaration *WithStatement* associée à la définition *ParamDef*. Le champ **Commentaires détaillés** est mappé en un attribut d'affichage dans la déclaration *WithStatement* associée à la définition *ParamDef* englobante.

```
 1:  module TTCN3ModuleId "{"
 2:  parameters "{" ModuleParList "}"
 3:   with "{"
 4:    [ModuleParPicsPixitRefAttribute ";"]
 5:    [ModuleParComments ";"]
 6:    [DetailedComments ";"]
 7:  "}"
 8:  "}"
```

EXEMPLE:

Paramètres de la suite de tests				
Nom	**Type**	**Valeur initiale**	**Réf. PICS/PIXIT**	**Commentaires**
CAP_1	boolean	true	A.1.3	option 1 implemented
Tall	float	600.0	A.1.4	overall module timer
Commentaires détaillés	detailed comments			

Se mappe en:

```
 1:  module MyModule{
 2:   parameters { boolean CAP_1 := true, float Tall := 600.0 }
 3:   with {
 4:   display (CAP_1) "pics/pixit ref := A.1.3";
 5:   display (CAP_1) "comments := option 1 implemented";
 6:   display (Tall) "pics/pixit ref := A.1.4";
 7:   display (Tall) "comments := overall module timer";
 8:   display "detailed comments := detailed comments"
 9:   }
10:  }
```

6.3 Imports de modules

Imports	
Nom de la source	*GlobalModuleId [**recursive**]*
Langage source	*[LanguageSpec]*
Groupe	*[GroupReference]*
Réf. de la source	*[TabFreeText]*
Codage	*[TabFreeText]*
Commentaires	*[TabFreeText]*

Type	**Nom**
. *[ImportType]* .	. *ImportSpecification* .
Commentaires détaillés	*[TabFreeText]*

Figure 4/Z.141 – Formulaire des imports

6.3.1 Mappage

Le formulaire des imports est mappé en une déclaration *ImportDef* en langage noyau TTCN-3. Les champs **Nom de la source**, **Langage source**, **Type** et **Nom** sont directement utilisés dans la déclaration *ImportDef* correspondante en langage noyau. Les champs **Réf. de la source, Commentaires** et **Commentaires détaillés** sont convertis en des attributs d'affichage dans la déclaration *WithStatement* associée à la déclaration *ImportDef*. Le champ **Codage** est converti en un attribut de codage dans la déclaration *WithStatement* associée à la déclaration *ImportDef*.

Si toutes les définitions d'un module sont importées, le type *ImportType* doit être vide et la spécification *ImportSpecification* doit employer le mot clé **`all`**.

```
 1:  module TTCN3ModuleId "{"
 2:   ImportDef
 3:      with "{"
 4:   [ImportsSourceRefAttribute ";"]
 5:   [CommentsAttribute ";"]
 6:   [ImportsSourceDefinitionCommentsAttribute ";"]
 7:   [DetailedCommentsAttribute ";"]
 8:   [EncodeAttribute ";"]
 9:   "}"
10:  "}"
```

EXEMPLE:

Imports		
Nom de la source	ModuleA recursive	
Groupe du langage source	ASN.1:1997	
Réf. de la source		
Codage	EN 800 900 version 2	
Commentaires	BER	
Type	**Nom**	**Commentaires**
constant	all except foobar	
Type	MyType	foobar
Group	AtoU_CTR	
Commentaires détaillés	detailed comments	

Se mappe en:

```
 1:  module MyModule {
 2:   import from ModuleA recursive language "ASN.1997" {
 3:   const all except foobar;
 4:   type MyType;
 5:   Group AtoU_CTR;
 6:   } with {
 7:   display "imports source ref := EN 800 900 version 2";
 8:   display "comments := importing declarations from ATS";
 9:   display "detailed comments := detailed comments";
10:   encode "BER";
11:   }
12:  }
```

6.4 Types simples

Types simples			
Groupe	*[GroupReference]*		
Nom	**Définition**	**Codage**	**Commentaires**
. *SubTypeIdentifier* .	. *Type [ArrayDef] [SubTypeSpec]* .	. *[TabFreeText]* .	. *[TabFreeText]* .
Commentaires détaillés	*[TabFreeText]*		

Figure 5/Z.141 – Formulaire des types simples

6.4.1 Mappage

Le formulaire des types simples est mappé en une série de déclarations de définition des types simples au même niveau de groupe. Les définitions des types simples sont toutes des définitions de type *SubTypeDef*.

Le champ **Commentaires détaillés** est mappé en un attribut d'affichage dans la déclaration *WithStatement* associée au groupe ou au module englobant. Les champs **Codage** et **Commentaires** sont mappés en des attributs de codage et d'affichage respectivement dans la déclaration *WithStatement* associée à la définition des types simples respectifs.

```
 1:  module TTCN3ModuleId "{"
 2:   type Type SubTypeIdentifier [ArrayDef] [SubTypeSpec] with "{"
 3:   [EncodeAttribute ";"]
 4:   [CommentsAttribute ";"]
 5:  "}" with "{"
 6:   [SimpleTypesDetailedCommentsAttribute ";"]
 7:  "}"
```

EXEMPLE:

Types simples			
Groupe	SimpleTypes/		
Nom	**Définition**	**Codage**	**Commentaires**
EQ_NUMBER	integer (1 .. 20)	PER	God knows
Commentaires détaillés	detailed comments		

Se mappe en:

```
 1:  module MyModule {
 2:   group SimpleTypes {
 3:   type integer EQ_NUMBER (1..20) with {
 4:   encode "PER";
 5:   display "comments := God knows";
 6:   }
 7:   } with {
 8:   display "simple types detailed comments := detailed comments";
 9:   }
10:  }
```

6.5 Types structurés

Types structurés			
Nom	*StructTypeIdentifier[StructDefFormalParList]*		
Groupe	*[GroupReference]*		
Structure	*StructureType*		
Codage	*[TabFreeText]*		
Commentaires	*[TabFreeText]*		
Nom du champ	**Type du champ**	**Codage du champ**	**Commentaires**
. *FieldIdentifier* . . .	. *Type [ArrayDef]* *[SubTypeSpec]* *[OptionalKeyword]* .	. *[TabFreeText]* . . .	. *[TabFreeText]* . . .
Commentaires détaillés	*[TabFreeText]*		

Figure 6/Z.141 – Formulaire des types structurés

6.5.1 Mappage

Le formulaire des types structurés est mappé en une déclaration de définition des types structurés en notation TTCN-3. Les types suivants emploieront ce formulaire: *RecordDef, UnionDef* et *SetDef*.

Les champs **Commentaires** et **Commentaires détaillés** sont mappés en des attributs d'affichage dans la déclaration *WithStatement* correspondante. Les champs **Commentaires et Codage de champ** de chacun des éléments de champ sont mappés en un attribut de codage et d'affichage respectivement, qualifié par l'identificateur *FieldIdentifier* dans la déclaration *WithStatement* correspondante.

```
 1:  module TTCN3ModuleId "{"
 2:   type StructureType StructTypeIdentifier [StructDefFormalParList] "{"
 3:   {Type FieldIdentifier [ArrayDef] [SubtypeSpec] [OptionalKeyword]}
 4:   "}" with "{"
 5:   [EncodeAttribute ";"]
 6:   [CommentsAttribute ";"]
 7:   {FieldCommentsAttribute ";"}
 8:   {FieldEncodeAttribute ";"}
 9:   [DetailedCommentsAttribute ";"]
10:   "}"
11:  "}"
```

EXEMPLE:

Types structurés			
Nom	routing_label(SLSel_Type)		
Groupe			
Structure	record		
Codage	BER		
Commentaires	header for routing info		
Nom de l'élément	**Définition du type**	**Codage du champ**	**Commentaires**
DestPC	BIT_14		destination point code
OrigPC	BIT_14		origination point code
SLSel	SLSel_Type	PER	signalling link selection
Commentaires détaillés	overrides previous definitions		

Se mappe en:

```
 1:  module MyModule {
 2:   type record routing_label(SLSel_Type) {
 3:   BIT_14 DestPC,
 4:   BIT_14 OrigPC,
 5:   SLSel_Type SLSel
 6:   } with {
 7:   encode "BER";
 8:   display "comments := header for routing info";
 9:   display (DestPC) "comments := destination point code";
10:   display (OrigPC) "comments := origination point code";
11:   display (SLSel) "comments := signalling link selection";
12:   encode (SLSel) "PER";
13:   display "detailed comments := overrides previous definition";
14:   }
15:  }
```

6.6 Types SequenceOf

Types SequenceOf					
Groupe	*[GroupReference]*				
Nom	**Type**	**Sorte**	**Longueur**	**Codage**	**Commentaires**
. *StructTypeIdentifier* .	. *Type [SubTypeSpec]* .	. *RecordOrSet* .	. *[StringLength]* .	. *[TabFreeText]* .	. *[TabFreeText]* .
Commentaires détaillés	*[TabFreeText]*				

Figure 7/Z.141 – Formulaire des types SequenceOf

6.6.1 Mappage

Le formulaire des types SequenceOf est mappé en une série de déclarations de définition des types Sequenceof au même niveau de groupe. Ce formulaire doit être employé pour les définitions des types *RecordOfDef* et *SetOfDef*.

Le champ **Commentaires détaillés** est mappé en un attribut d'affichage dans la déclaration *WithStatement* associée au groupe ou au module englobant. Les champs **Codage** et **Commentaires** sont mappés en des attributs de codage et d'affichage respectivement dans la déclaration *WithStatement* associée à la définition des types SequenceOf respectifs.

```
 1:  module TTCN3ModuleId "{"
 2:   type record of [StringLength] Type StructTypeIdentifier [SubTypeSpec]
 3:   with {
 4:   [EncodeAttribute ";"]
 5:   [CommentsAttribute ";"]
 6:   }
 7:   type set of [StringLength] Type StructTypeIdentifier [SubTypeSpec]
 8:   with {
 9:   [EncodeAttribute ";"]
10:   [CommentsAttribute ";"]
11:   }
12:  } with {
13:   [SequenceOfTypesDetailedCommentsAttribute ";"]
14:  }
```

EXEMPLE:

Types SequenceOf					
Groupe	SequenceOfTypes/				
Nom	**Type**	**Sorte**	**Longueur**	**Codage**	**Commentaires**
RecordOfIntegers	integer(1..10)	record	10	BER	ten integers
SetOfBooleans	boolean	set	3	PER	three booleans
Commentaires détaillés	example sequenceof types				

Se mappe en:

```
 1:  module MyModule {
 2:   group SequenceOfTypes {
 3:   type record of length(10) integer RecordOfIntegers(1..10) with {
 4:   encode "BER";
 5:   display "comments := ten integers";
 6:   }
 7:   type set of length(3) boolean SetOfBooleans with {
 8:   encode "PER";
 9:   display "comments := three booleans";
10:   }
11:   } with {
12:   display "sequenceof types detailed comments := example sequenceof types";
13:   }
14:  }
```

6.7 Types énumérés

Types énumérés		
Nom **Groupe** **Codage** **Commentaires**	*EnumTypeIdentifier* *[GroupReference]* *[TabFreeText]* *[TabFreeText]*	
Nom dans l'énumération	**Valeur dans l'énumération**	**Commentaires**
. *EnumerationIdentifier* .	. *[Number]* .	. *[TabFreeText]* .
Commentaires détaillés	*[TabFreeText]*	

Figure 8/Z.141 – Formulaire des types énumérés

6.7.1 Mappage

Le formulaire des types énumérés est mappé en une déclaration de définition des types énumérés en langage noyau TTCN-3. Les champs **Commentaires** et **Commentaires détaillés** sont mappés en des attributs d'affichage dans la déclaration *WithStatement* correspondante, et le champ **Codage** est mappé en un attribut de codage dans la déclaration *WithStatement* correspondante. Les champs **Commentaires** de chaque énumération sont mappés en des attributs d'affichage qualifiés par l'identificateur *EnumerationIdentifier* dans la déclaration *WithStatement* correspondante.

```
 1:  module TTCN3ModuleId "{"
 2:   type enumerated EnumTypeIdentifier "{"
 3:   EnumerationIdentifier ["(" Number ")"]
 4:   {"," EnumerationIdentifier ["(" Number ")"]}
 5:   } with {
 6:   [EncodeAttribute ";"]
 7:   [CommentsAttribute ";"]
 8:   {NamedValueCommentsAttribute ";"}
 9:   [DetailedCommentsAttribute ";"]
10:   }
11:  }
```

EXEMPLE:

Types énumérés		
Nom **Groupe** **Codage** **Commentaires**	Weekdays BER days of the week	
Nom dans l'énumération	**Valeur dans l'énumération**	**Commentaires**
Monday	1	
Tuesday	2	
Wednesday	3	half way there
Thursday	4	
Friday	5	TGIF
Saturday	6	
Sunday	7	
Commentaires détaillés	wish it were Friday	

Se mappe en:

```
 1:  module MyModule {
 2:   type enumerated Weekdays {
 3:   Monday(1), Tuesday(2), Wednesday(3), Thursday(4), Friday(5),
 4:   Saturday(6), Sunday(7)
 5:   } with {
 6:   encode "BER";
 7:   display "comments := days of the week";
 8:   display (Wednesday) "comments := half way there";
 9:   display (Friday) "comments := TGIF";
10:   display "detailed comments := wish it were Friday";
11:   }
12:  }
```

6.8 Types de ports

Types de ports		
Nom **Groupe** **Modèle de communication** **Commentaires**	*PortTypeIdentifier* *[GroupReference]* *PortModelType* *[TabFreeText]*	
Type/Signature	**Direction**	**Commentaires**
. *TypeOrSignature* .	. *InOutOrInout* .	. *[TabFreeText]* .
Commentaires détaillés	*[TabFreeText]*	

Figure 9/Z.141 – Formulaire des types de ports

6.8.1 Mappage

Le formulaire des types de ports est mappé en une définition des types de ports en langage noyau TTCN-3. Les champs **Commentaires** et **Commentaires détaillés** sont mappés en des attributs d'affichage dans la déclaration *WithStatement* correspondante. Les champs **Commentaires** du tableau des types et signatures sont mappés en des attributs d'affichage dans la déclaration *WithStatement* qualifiée par l'identificateur du type ou de la signature. A chaque type ou signature correspondra toujours une ligne.

Le champ **Type/Signature** est affecté du mot clé **all** si tous les types ou toutes les signatures des procédures, définis dans le module, peuvent être transmis à travers ce port de communication.

```
 1:  module TTCN3ModuleId "{"
 2:   type port PortTypeIdentifier PortModelType "{"
 3:   PortTypeDef
 4:   "}" with "{"
 5:   [CommentsAttribute ";"]
 6:   {TypeOrSignatureCommentsAttribute ";"}
 7:   [DetailedCommentsAttribute ";"]
 8:   "}"
 9:  "}"
```

EXEMPLE:

Types de ports		
Nom **Groupe** **Modèle de communication** **Commentaires**	MyPortType message example port type	
Type/Signature	**Direction**	**Commentaires**
MsgType1	in	first comment
MsgType2	in	second comment
MsgType3	out	
Commentaires détaillés	detailed comment	

Se mappe en:

```
 1:  module MyModule {
 2:   type port MyPortType message {
 3:   in MsgType1;
 4:   in MsgType2;
 5:   out MsgType3;
 6:   } with {
 7:   display "comments := example port type";
 8:   display (MsgType1) "comments := first comment";
 9:   display (MsgType2) "comments := second comment";
10:   display "detailed comments := detailed comment";
11:   }
12:  }
```

6.9 Types de composantes

Types de composantes			
Nom	*ComponentTypeIdentifier*		
Groupe	*[GroupReference]*		
Commentaires	*[TabFreeText]*		
Nom de la définition locale	**Type**	**Valeur initiale**	**Commentaires**
. *VarConstOrTimerIdentifier* .	. *TypeOrTimer [ArrayDef]* .	. *[ConstantExpression | Expression]* .	. *[TabFreeText]* .
Nom du port	**Type du port**		**Commentaires**
. *PortIdentifier* .	. *PortType[ArrayDef]* .		. *[TabFreeText]* .
Commentaires détaillés	*[TabFreeText]*		

Figure 10/Z.141 – Formulaire des types de composantes

6.9.1 Mappage

Le formulaire des types de composantes est mappé en une définition des types de composantes en langage noyau TTCN-3. Le formulaire est converti en trois parties.

La première partie comporte les champs d'en-tête **Commentaires** et **Commentaires détaillés**, qui sont convertis en des attributs d'affichage dans la déclaration *WithStatement* associée à la définition des types de composantes.

La deuxième partie comporte des constantes, des variables et des temporisateurs locaux définis dans le type de composantes. Ces définitions peuvent être présentes partout dans la définition des types de composantes du langage noyau, mais, dans le formulaire, elles doivent être séparées des instances de port et être affichées dans un tableau distinct. Il faut conserver l'ordre des définitions, puisqu'elles peuvent dépendre les unes des autres. La colonne **Type** doit être affectée du mot clé **`timer`** pour tous les temporisateurs et du type constant précédé du mot clé **`const`** pour toutes les constantes. A chaque constante, variable ou temporisateur correspondra toujours une ligne. La colonne **Commentaires** de ce tableau est convertie en des attributs d'affichage qualifiés par l'identificateur de la définition locale dans la déclaration *WithStatement* associée à la définition des types de composantes.

La troisième partie comporte des instances de port définies dans le type de composantes. Toute définition séquentielle peut être jointe au type de ports. A chaque instance de port correspondra toujours une ligne. La colonne **Commentaires** de ce tableau est convertie en des attributs d'affichage qualifiés par l'identificateur *PortIdentifier* dans la déclaration *WithStatement* associée à la définition des types de composantes.

```
 1:  module TTCN3ModuleId "{"
 2:   type component ComponentTypeIdentifier "{"
 3:   var Type VarIdentifier [":=" Expression] ";"
 4:   timer TimerIdentifier [":=" Expression] ";"
 5:   const Type ConstIdentifier ":=" ConstantExpression ";"
 6:   PortList
 7:   "}" with "{"
 8:   [CommentsAttribute ";"]
 9:   {PortCommentsAttribute ";"}
10:   [DetailedCommentsAttribute ";"]
11:   "}"
12:  "}"
```

EXEMPLE:

Types de composantes			
Nom **Groupe** **Commentaires**	MyComponentType an example component type		
Nom de la définition locale	**Type**	**Valeur initiale**	**Commentaires**
PI	const float	3.14	the ratio
x	float	PI * 2	double PI
t1	timer	15 min	a 15 second timer
Nom du port	**Type du port**		**Commentaires**
PCO1	MyMessagePortType		first comment
PCO2	MyProcedurePortType		second comment
Commentaires détailés	detailed comments		

Se mappe en:

```
 1:  module MyModule {
 2:   type component MyComponentType {
 3:   const float PI := 3.14;
 4:   var float x := PI * 2;
 5:   timer t1 := 15;
 6:   port MyMessagePortType PCO1;
 7:   port MyProcedurePortType PCO2;
 8:   } with {
 9:   display "comments := an example component type";
10:   display (PI) "comments := the ratio";
11:   display (x) "comments := double PI";
12:   display (t1) "comments := a 15 second timer";
13:   display (PCO1) "comments := first comment";
14:   display (PCO2) "comments := second comment";
15:   display "detailed comments := detailed comments";
16:   }
17:  }
```

6.10 Constantes

Constantes			
Groupe	*[GroupReference]*		
Nom	**Type**	**Valeur**	**Commentaires**
. *ConstIdentifier* | *ExtConstIdentifier* .	. *Type [ArrayDef]* .	. *ConstantExpression* | **external**	. *[TabFreeText]* .
Commentaires détaillés	*[TabFreeText]*		

Figure 11/Z.141 – Formulaire des constantes

6.10.1 Mappage

Le formulaire des constantes est mappé en une série de déclarations de définition des constantes et des constantes externes au même niveau de groupe. Le champ **Commentaires détaillés** est mappé en un attribut d'affichage dans la déclaration *WithStatement* associée au groupe ou au module englobant. Les champs **Commentaires** sont mappés en des attributs d'affichage dans la déclaration *WithStatement* associée à la définition des constantes respectives. Pour une constante externe, le champ **Valeur** est affecté du mot clé **`external`**.

```
 1:  module TTCN3ModuleId "{"
 2:   const Type ConstIdentifier[ArrayDef] ":=" ConstantExpression with "{"
 3:   [CommentsAttribute ";"]
 4:   "}"
 5:   external const Type ConstIdentifier with "{"
 6:   [CommentsAttribute ";"]
 7:   "}"
 8:  "}" with "{"
 9:   [ConstantsDetailedCommentsAttribute ";"]
10:  "}"
```

EXEMPLE:

Constantes			
Groupe	Constants1		
Nom	**Type**	**Valeur**	**Commentaires**
TOTO	integer	external	defined somewhere else
SEL2	boolean	(5 + TOTO) < 10	TOTO limit reached
T1	integer[1..3]	{1,3,2}	
Commentaires détaillés	detailed comments		

Se mappe en:

```
 1:  module MyModule {
 2:   group Constants1 {
 3:   external const integer TOTO with {
 4:   display "comments := defined somewhere else";
 5:   }
 6:   const boolean SEL2 := (5 + TOTO) < 10 with {
 7:   display "comments := TOTO limit reached";
 8:   }
 9:   const integer T1[1..3] := {1,3,2};
10:   } with {
11:   display "detailed comments := detailed comments";
12:   }
13:  }
```

6.11 Définition des signatures

Définition des signatures		
Nom	*SignatureIdentifier*([*SignatureFormalParList*])	
Groupe	*[GroupReference]*	
Type de retour	*[Type]* | **noblock**	
Commentaires	*[TabFreeText]*	
Type d'exception		**Commentaires**
. *[ExceptionType]* .		. *[TabFreeText]* .
Commentaires détaillés	*[TabFreeText]*	

Figure 12/Z.141 – Formulaire de définition des signatures

6.11.1 Mappage

Le formulaire de définition des signatures est mappé en une définition des signatures en langage noyau TTCN-3. Les champs **Commentaires** et **Commentaires détaillés** sont mappés en des attributs d'affichage dans la déclaration *WithStatement*. Les champs **Commentaires** du tableau des exceptions sont mappés en des attributs d'affichage qualifiés par le type d'exception dans la déclaration *WithStatement* correspondante. Des procédures non bloquantes doivent spécifier le mot clé **noblock** en tant que type de retour.

```
 1:  module TTCN3ModuleId "{"
 2:   signature SignatureIdentifier "(" [SignatureFormalParList] ")"
 3:   [return Type | noblock]
 4:   [exception "(" ExceptionTypeList ")"]
 5:   with "{"
 6:   [CommentsAttribute ";"]
 7:   [ExceptionCommentsAttribute ";"]
 8:   [DetailedCommentsAttribute ";"]
 9:   "}"
10:  "}"
```

EXEMPLE:

Définition des signatures	
Nom	read(integer fields, inout charstring buf, integer nbyte)
Groupe	
Type de retour	integer
Commentaires	reads from a file

Type d'exception	Commentaires
integer	error code
MyException	user defined
Commentaires détaillés	required: unistd.h

Se mappe en:

```
 1:  module MyModule {
 2:   signature read_syscall(in integer fields,
 3:   inout charstring buf,
 4:   in integer nbyte)
 5:   return integer
 6:   exception (integer)
 7:   with {
 8:   display "comments := reads from a file";
 9:   display (integer) "comments := error code of system call";
10:   display "detailed comments := required: unistd.h";
11:   }
12:  }
```

6.12 Modèles simples

Modèles simples					
Groupe	*[GroupReference]*				
Nom	**Type**	**Déduit de**	**Valeur**	**Codage**	**Commentaires**
. *Template Identifier* .	. *BaseTemplate* .	. *[DerivedDef]* .	. *TemplateBody* .	. *[TabFreeText]* .	. *[TabFreeText]* .
Commentaires détaillés	*[TabFreeText]*				

Figure 13/Z.141 – Formulaire des modèles simples

6.12.1 Mappage

Le formulaire des modèles simples est mappé en une série de déclarations de définition des modèles simples au même niveau de groupe. Les définitions des modèles simples sont toutes des définitions de modèles qui contiennent un champ *SimpleSpec* ou *ArrayValueOrAttrib* en tant que champ *TemplateBody*. Les types correspondants sont définis dans un formulaire des types simples, SequenceOf ou énumérés.

Le champ **Commentaires détaillés** est mappé en un attribut d'affichage dans la déclaration *WithStatement* associée au groupe ou au module englobant. Les champs **Commentaires** et **Codage** sont mappés en des attributs d'affichage et de codage, qualifiés par l'identificateur *TemplateIdentifier* dans la déclaration *WithStatement* associée à la déclaration de définition des modèles simples respectifs.

```
1:  module TTCN3ModuleId "{"
2:   template BaseTemplate[DerivedDef] := TemplateBody with "{"
3:   [EncodeAttribute ";"]
4:   [CommentsAttribute ";"]
5:   "}"
6:  "}" with "{"
7:   [SimpleTemplatesDetailedCommentsAttribute ";"]
8:  "}"
```

EXEMPLE:

Modèles simples					
Groupe	SimpleTemplates1				
Nom	**Type**	**Déduit de**	**Valeur**	**Codage**	**Commentaires**
MyTemplate1	MyType1		3	BER	foobar
MyTemplate11(integer index)	MyType1	MyTemplate1	3*index	PER	the current index
Commentaires détaillés	an example				

Se mappe en:

```
 1:  module MyModule {
 2:   group SimpleTemplates {
 3:   template MyType1 MyTemplate1 with {
 4:   encode "BER";
 5:   display "comments := foobar";
 6:   }
 7:   template MyType1 MyTemplate11(integer index)
 8:   modifies MyTemplate1 := 3 * index
 9:   with {
10:   encode "PER";
11:   display "comments := the current index";
12:   }
13:   } with {
14:   display "simple templates detailed comments := an example";
15:   }
16:  }
```

6.13 Modèles structurés

Modèles structurés			
Nom	*TemplateIdentifier[(TemplateFormalParList)]*		
Groupe	*[GroupReference]*		
Type/Signature	*TypeIdentifier | SignatureIdentifier*		
Déduit de	*[TemplateRef]*		
Codage	*[TabFreeText]*		
Commentaires	*[TabFreeText]*		
Nom de l'élément	**Valeur de l'élément**	**Codage de l'élément**	**Commentaires**
. *FieldReference* .	. *FieldValueOrAttrib* .	. *[TabFreeText]* .	. *[TabFreeText]* .
Commentaires détaillés	*[TabFreeText]*		

Figure 14/Z.141 – Formulaire des modèles structurés

6.13.1 Mappage

Le formulaire des modèles structurés est mappé en une déclaration de définition des modèles structurés en notation TTCN-3. Les définitions des modèles structurés sont toutes des définitions de modèles qui contiennent les champs *FieldSpecList* en tant que corps des modèles. Les types correspondants sont définis dans un formulaire des types structurés.

Les champs **Commentaires** et **Commentaires détaillés** sont mappés en des attributs d'affichage dans la déclaration *WithStatement* associée à la définition des modèles structurés. Le champ **Codage** est mappé en un attribut de codage dans la déclaration *WithStatement* associée à la définition des modèles structurés.

Les champs **Commentaires** du tableau des éléments sont mappés en des attributs d'affichage qualifiés par la référence au champ dans la déclaration *WithStatement* associée à la définition des modèles structurés. Les champs **Codage d'élément** sont mappés en des attributs de codage qualifiés par la référence au champ dans la déclaration *WithStatement* associée à la définition des modèles structurés.

```
module TTCN3ModuleId "{"
 template BaseTemplate [DerivedDef] ":=" TemplateBody with "{"
 [EncodeAttribute ";"]
 [CommentsAttribute ";"]
 [FieldEncodeAttribute ";"]
 [FieldCommentsAttribute ";"]
 [DetailedCommentsAttribute ";"]
 "}"
"}"
```

EXEMPLE:

Modèles structurés			
Nom	MyStructuredTemplate11(integer para1, boolean para2)		
Groupe			
Type/Signature	MyStructuredType		
Déduit de	MyStructuredTemplate1		
Codage	BER		
Commentaires	example structured template		
Nom de l'élément	**Valeur de l'élément**	**Codage de l'élément**	**Commentaires**
field1	13		first field
field2	para2	PER	second field
field3	para1		third field
Commentaires détaillés	detailed comments		

Se mappe en:

```
module MyModule {
 template MyStructuredType MyStructuredTemplate11(integer para1,
 boolean para2)
 modifies MyStructuredTemplate1 := {
 field1 := 13,
 field2 := para2,
 field3 := para1
 } with {
 encode "BER";
 display "comments := example structured template";
 display (field1) "comments := first field";
 encode (field2) "PER";
 display (field2) "comments := second field";
 display (field3) "comments := third field";
 display "detailed comments := detailed comments";
 }
}
```

6.14 Fonctions

Fonctions			
Nom	*FunctionIdentifier([FunctionFormalParList])*		
Groupe	*[GroupReference]*		
Opère sur	*[ComponentType]*		
Type de retour	*[Type]*		
Commentaires	*[TabFreeText]*		
Nom de la définition locale	**Type**	**Valeur initiale**	**Commentaires**
. *VarConstOrTimerIdentifier* .	. *TypeOrTimer* .	. *[Expression | ConstantExpression]*	. *[TabFreeText]* .
Comportement			
. *FunctionStatement* | **external** .			
Commentaires détaillés	*[TabFreeText]*		

Figure 15/Z.141 – Formulaires des fonctions

6.14.1 Mappage

Le formulaire des fonctions est mappé en une déclaration de définition des fonctions en notation TTCN-3 ou de définition des fonctions externes. Il est converti en trois parties.

La première partie comporte les champs d'en-tête. Les champs **Commentaires** et **Commentaires détaillés** sont mappés en des attributs d'affichage dans la déclaration *WithStatement* associée à la définition des fonctions.

La deuxième partie comporte des constantes, des variables et des temporisateurs locaux définis dans la définition des fonctions. Ces définitions peuvent être présentes partout dans le corps des fonctions du langage noyau, mais, dans le formulaire, elles doivent être séparées du reste du corps des fonctions et être affichées dans un tableau distinct. Il faut conserver l'ordre des définitions, puisqu'elles peuvent dépendre les unes des autres. La colonne **Type** doit être affectée du mot clé **`timer`** pour tous les temporisateurs et du type constant précédé du mot clé **`const`** pour toutes les constantes. Les champs **Commentaires** sont convertis en des attributs d'affichage qualifiés par l'identificateur local dans la déclaration *WithStatement* associée à la définition des fonctions.

La troisième partie comporte le corps des fonctions du langage noyau TTCN-3, sans les constantes, variables et temporisateurs locaux.

Pour une fonction externe, seul le comportement contient le mot clé **`external`**.

```
 1:  module TTCN3ModuleId "{"
 2:   function FunctionIdentifier "(" [FunctionFormalParList] ")"
 3:   [runs on ComponentType]
 4:   [return Type] "{"
 5:   var Type VarIdentifier [":=" Expression] ";"
 6:   timer TimerIdentifier [":=" Expression]";"
 7:   const Type ConstIdentifier ":=" ConstantExpression ";"
 8:   {FunctionStatement}
 9:   "}" with "{"
10:   [CommentsAttribute ";"]
11:   [VarConstOrTimerCommentsAttribute ";"]
12:   [DetailedCommentsAttribute ";"]
13:   "}"
14:  "}"
```

EXEMPLE:

Fonctions			
Nom **Groupe** **Opère sur** **Type de retour** **Commentaires**	MyFunction(integer para1) MyComponentType boolean example function definition		
Nom de la définition locale	**Type**	**Valeur initiale**	**Commentaires**
MyLocalVar	boolean	false	local variable
MyLocalConst	const float	60	local constant
MyLocalTimer	timer	15 * MyLocalConst	local timer
Comportement			

if (para1 == 21) {
 MyLocalVar := true;
}
if (MyLocalVar) {
 MyLocalTimer.start;
 MyLocalTimer.timeout;
}
return (MyLocalVar);			

Commentaires détaillés	detailed comments		

Se mappe en:

```
 1:  module MyModule {
 2:   function MyFunction(in integer para1)
 3:   runs on MyComponentType
 4:   return boolean {
 5:   var boolean MyLocalVar := false;
 6:   const float MyLocalConst := 60;
 7:   timer MyLocalTimer := 15 * MyLocalConst;
 8:
 9:   if (para1 == 21) {
10:   MyLocalVar := true;
11:   }
12:   if (MyLocalVar) {
13:   MyLocalTimer.start;
14:   MyLocalTimer.timeout;
15:   }
16:   return (MyLocalVar);
17:   } with {
18:   display "comments := example function definition";
19:   display (MyLocalVar) "comments := local variable";
20:   display (MyLocalConst) "comments := local constant";
21:   display (MyLocalTimer) "comments := local timer";
22:   display "detailed comments := detailed comments";
23:   }
24:  }
```

6.15 Etapes Altstep

Etapes Altstep			
Nom	*AltstepIdentifier([AltstepFormalParList])*		
Groupe	*[GroupReference]*		
Objet	*[TabFreeText]*		
Opère sur	*[ComponentType]*		
Commentaires	*[TabFreeText]*		
Nom de la définition locale	**Type**	**Valeur initiale**	**Commentaires**
. *VarConstOrTimerIdentifier* .	. *TypeOrTimer [ArrayDef]* .	. *[Expression | ConstantExpression]* .	. *[TabFreeText]* .
Comportement			
. *AltGuardList* .			
Commentaires détaillés	*[TabFreeText]*		

Figure 16/Z.141 – Formulaire des étapes Altstep

6.15.1 Mappage

Le formulaire des étapes Altstep est mappé en une déclaration de définition des étapes Altstep en notation TTCN-3. Il est converti en trois parties.

La première partie comporte les champs d'en-tête. Les champs **Objet, Commentaires** et **Commentaires détaillés** sont mappés en des attributs d'affichage dans la déclaration *WithStatement* associée à la définition des étapes Altstep.

La deuxième partie comporte des constantes, des variables et des temporisateurs locaux définis dans la définition des étapes Altstep. Ces définitions peuvent être présentes partout dans le corps des étapes Altstep du langage noyau, mais, dans le formulaire, elles doivent être séparées du reste du corps des étapes Altstep et être affichées dans un tableau distinct. Il faut conserver l'ordre des définitions, puisqu'elles peuvent dépendre les unes des autres. La colonne **Type** doit être affectée du mot clé **`timer`** pour tous les temporisateurs et du type constant précédé du mot clé **`const`** pour toutes les constantes. Les champs **Commentaires** sont convertis en des attributs d'affichage qualifiés par l'identificateur local dans la déclaration *WithStatement* associée à la définition des étapes Alstep.

La troisième partie comporte la liste *AltGuardList* des étapes Altstep du langage noyau TTCN-3.

```
 1:  module TTCN3ModuleId "{"
 2:   teststep AltstepIdentifier "(" [AltstepFormalParList] ")"
 3:   [runs on ComponentType] "{"
 4:   AltGuardList
 5:   "}" with "{"
 6:   [PurposeAttribute ";"]
 7:   [CommentsAttribute ";"]
 8:   [VarConstOrTimerCommentsAttribute ";"]
 9:   [DetailedCommentsAttribute ";"]
10:   "}"
11:  "}"
```

EXEMPLE:

Etapes Altstep			
Nom	MyAltstep(integer para1)		
Groupe			
Opère sur	MyComponentType		
Objet	to do something		
Commentaires	example altstep definition		
Nom de la définition locale	**Type**	**Valeur initiale**	**Commentaires**
MyLocalVar	boolean	false	local variable
MyLocalConst	const float	60	local constant
MyLocalTimer	timer	15 * MyLocalConst	local timer
Comportement			

[] PCO1.receive(MyTemplate(para1, CompVar) {
verdict.set(inconc);
}
[] PCO2.receive {
repeat;
}
[] CompTimer.timeout {
verdict.set(fail);
stop;
}			

Commentaires détaillés	detailed comments		

Se mappe en:

```
 1:  module MyModule {
 2:   altstep MyTeststep(integer para1) runs on MyComponentType {
 3:   var boolean MyLocalVar := false;
 4:   const float MyLocalConst := 60;
 5:   timer MyLocalTimer := 15 * MyLocalConst;
 6:
 7:   [] PCO1.receive(MyTemplate(para1, CompVar)) {
 8:   verdict.set(inconc);
 9:   }
10:   [] PCO2.receive {
11:   repeat;
12:   }
13:   [] CompTimer.timeout {
14:   verdict.set(fail);
15:   stop;
16:   }
17:   } with {
18:   display "purpose := to do something";
19:   display "comments := example altstep definition";
20:   display "detailed comments := detailed comments";
21:   }
22:  }
```

6.16 Jeux Testcase

Jeux Testcase			
Nom **Groupe** **Objet** **Interface système** **Type MTC** **Commentaires**	*TestcaseIdentifier*(*[TestcaseFormalParList]*) *[GroupReference]* *[TabFreeText]* *[ComponentType]* *ComponentType* *[TabFreeText]*		
Nom de la définition locale	**Type**	**Valeur initiale**	**Commentaires**
. *VarConstOrTimerIdentifier* .	. *TypeOrTimer* .	. *[Expression | ConstantExpression]*	. *[TabFreeText]* .
Comportement			
. *FunctionStatement* .			
Commentaires détaillés	*[TabFreeText]*		

Figure 17/Z.141 – Formulaire des jeux Testcase

6.16.1 Mappage

Le formulaire des jeux Testcase est mappé en une déclaration de définition des jeux Testcase en notation TTCN-3. Il est converti en trois parties.

La première partie comporte les champs d'en-tête. Les champs **Objet, Commentaires** et **Commentaires détaillés** sont mappés en des attributs d'affichage dans la déclaration *WithStatement* associée à la définition des jeux Testcase.

La deuxième partie comporte des constantes, des variables et des temporisateurs locaux définis dans la définition des jeux Testcase. Ces définitions peuvent être présentes partout dans le corps des jeux Testcase du langage noyau, mais, dans le formulaire, elles doivent être séparées du reste du corps des jeux Testcase et être affichées dans un tableau distinct. Il faut conserver l'ordre des définitions, puisqu'elles peuvent dépendre les unes des autres. La colonne **Type** doit être affectée du mot clé **`timer`** pour tous les temporisateurs et du type constant précédé du mot clé **`const`** pour toutes les constantes. Les champs **Commentaires** sont convertis en des attributs d'affichage qualifiés par l'identificateur local dans la déclaration *WithStatement* associée à la définition des jeux Testcase.

La troisième partie comporte le corps des jeux Testcase du langage noyau TTCN-3, sans les constantes, variables et temporisateurs locaux.

```
 1:  module TTCN3ModuleId "{"
 2:   testcase TestcaseIdentifier [TestcaseFormalParList]
 3:   [runs on ComponentType]
 4:   [system ComponentType] "{"
 5:   var Type VarIdentifier [":=" Expression] ";"
 6:   timer TimerIdentifier [":=" Expression] ";"
 7:   const Type ConstIdentifier ":=" ConstantExpression;
 8:   {FunctionStatement}
 9:   "}" with "{"
10:   [CommentsAttribute ";"]
11:   [PurposeAttribute ";"]
12:   [VarConstOrTimerCommentsAttribute ";"]
13:   [DetailedCommentsAttribute ";"]
14:   "}"
15:  "}"
```

EXEMPLE:

Jeux Testcase			
Nom	MyTestcase(integer para1)		
Groupe			
Objet	do something useful		
Interface système	MyComponentType		
Type MTC	MyComponentType		
Commentaires	example testcase definition		
Nom de la définition locale	**Type**	**Valeur initiale**	**Commentaires**
MyLocalVar	boolean	false	local variable
MyLocalConst	const float	60	local constant
MyLocalTimer	timer	15 * MyLocalConst	local timer

Comportement

```
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	detailed comments

Se mappe en:

```
 1: module MyModule {
 2:     testcase MyTestcase(in integer para1)
 3:         runs on MyComponentType
 4:         system MyComponentType {
 5:         var boolean MyLocalVar := false;
 6:         const float MyLocalConst := 60;
 7:         timer MyLocalTimer := 15 * MyLocalConst;
 8:         var default MyDefault := activate(OtherwiseFail());
 9:
10:         ISAP1.send(ICONreq:{}); /* Inline template definition */
11:         alt {
12:             /* use of a template */
13:             [] MSAP2.receive(Medium_Connection_Request()) {
14:     MSAP2.send(MDATreq:Medium_Connection_Confirmation());
15:                 alt {
16:                     [] ISAP1.receive(ICONconf:{}) {
17:                         ISAP1.send(Data_Request(TestSuitePar));
18:                         alt {
19:                             [] MSAP2.receive(Medium_Data_Transfer()) {
20:                                 MSAP2.send(MDATreq:cmi_synch1());
21:                                 ISAP1.send(IDISreq:{});
22:                             }
23:                             [] ISAP1.receive(IDISind:{}) {
24:                                 verdict.set(inconc);
25:                                 stop;
26:                             }
27:                         }
28:                     }
29:                     [] MSAP2.receive(MDATind_Connection_Request()) {
30:                         verdict.set(inconc);
31:                         stop;
32:                     }
33:                     [] ISAP1.receive(IDISind:{}) {
34:                         verdict.set(inconc);
35:                         stop;
36:                     }
37:                 }
38:             }
39:             [] ISAP1.receive(IDISind:{}) {
40:                 verdict.set(inconc);
41:                 stop;
42:             }
43:         }
44:     } with {
45:         display "purpose := do something useful";
46:         display "comments := example testcase definition";
47:         display (MyLocalVar) "comments := local variable";
48:         display (MyLocalConst) "comments := local constant";
49:         display (MyLocalTimer) "comments := local timer";
50:         display "detailed comments := detailed comments";
51:     }
    }
```

7 Productions BNF

1. TabFreeText ::= [ExtendedAlphaNum]

2. GroupReference ::= {GroupIdentifier "/"}+

3. EncRuleIdentifier ::= Identifier

4. CommentsAttribute :: = **display** """ "comments" ":=" TabFreeText """

5. DetailedCommentsAttribute :: = **display** """ "detailed comments" ":=" TabFreeText """

6. TTCN3ModuleId ::= ModuleIdentifier [DefinitiveIdentifier]

```
7.  ModuleAttributes ::= TabularPresentationFormatAttribute ";"
    ModuleVersionAttribute ";"
    ModuleDateAttribute ";"
    ModuleBaseStandardRefAttribute ";"
    ModuleTestStandardRefAttribute ";"
    ModulePICSRefAttribute ";"
    ModulePIXITRefAttribute ";"
    ModuleTestMethodAttribute ";"
    ModuleCommentsAttribute ";"
    ModuleDetailedCommentsAttribute ";"

8.  TabularPresentationFormatAttribute ::=
    display """ "presentation format := ETSI Tabular version" MajorVersion "." MinorVersion """

9.  MajorVersion ::= Number

10. MinorVersion ::= Number

11. ModuleVersionAttribute ::=
    display """ "module version" ":=" TabFreeText """

12. ModuleDateAttribute ::=
    display """ "module date" ":=" TabFreeText """

13. ModuleBaseStandardRefAttribute ::=
    display """ "module base standards ref" ":=" TabFreeText """

14. ModuleTestStandardRefAttribute ::=
    display """ "module test standards ref" ":=" TabFreeText """

15. ModulePICSRefAttribute ::=
    display """ "module pics ref" ":=" TabFreeText """

16. ModulePIXITRefAttribute ::=
    display """ "module pixit ref" ":=" TabFreeText """

17. ModuleTestMethodAttribute ::=
    display """ "module test method" ":=" TabFreeText """

18. ModuleCommentsAttribute ::=
    display """ "module comments" ":=" TabFreeText """

19. ModuleDetailedCommentsAttribute ::=
    display """ "module detailed comments" ":=" TabFreeText """

20. ModuleParPicsPixitRefAttribute ::=
    display "(" ModuleParIdentifier ")"
    """ "pics/pixit ref" ":=" TabFreeText """

21. ModuleParComments ::=
    display "(" ModuleParIdentifier ")"
    """ "comments" ":=" TabFreeText """

22. ImportsSourceRefAttribute ::=
    display """ "imports source ref" ":=" TabFreeText """

23. ImportsSourceDefinitionCommentsAttribute ::=
    display "(" ImportIdentifier ")"
    """ "comments" ":=" TabFreeText """

24. ImportSpecification ::= ( (Identifier | FullGroupIdentifier) | AllKeyword ) [ ExceptionsDef ]
    /* STATIC SEMANTIC: FullGroupIdentifier shall only be used for group imports. */

25. EncodeAttribute ::= encode """ TabFreeText """

26. SimpleTypesDetailedCommentsAttribute ::=
    display """ "simple types detailed comments" ":=" TabFreeText """

27. StructureType ::= record | union | set

28. FieldCommentsAttribute ::=
    display "(" FieldIdentifier ")" """ "comments" ":=" TabFreeText """

29. FieldEncodeAttribute ::=
    encode "(" FieldIdentifier ")" """ TabFreeText """

30. SequenceOfTypesDetailedCommentsAttribute ::=
    display """ "sequenceof types detailed comments" ":=" TabFreeText """
```

```
31. NamedValueCommentsAttribute ::=
    display "(" NamedValueIdentifier ")"
    """ "comments" ":=" TabFreeText """

32. TypeOrSignatureCommentsAttribute ::=
    display "(" TypeOrSignatureIdentifier ")"
    """ "comments" ":=" TabFreeText """

33. PortCommentsAttribute ::=
    display "(" PortIdentifier ")"
    """ "comments" ":=" TabFreeText """

34. ConstantsDetailedCommentsAttribute ::=
    display """ "simple types detailed comments" ":=" TabFreeText """

35. ExceptionCommentsAttribute ::=
    display "(" Type ")"
    """ "comments" ":=" TabFreeText """

36. VarConstOrTimerCommentsAttribute ::=
    display "(" VarConstOrTimerIdentifier ")"
    """ "comments" ":=" TabFreeText """

37. PurposeAttribute ::= display """ "purpose" ":=" TabFreeText """

38. SimpleTemplatesDetailedCommentsAttribute ::= display """ "simple templates detailed comments"
    ":=" TabFreeText """
```

SÉRIES DES RECOMMANDATIONS UIT-T

Série A	Organisation du travail de l'UIT-T
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	Gestion des télécommunications y compris le RGT et maintenance des réseaux
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, communication entre systèmes ouverts et sécurité
Série Y	Infrastructure mondiale de l'information, protocole Internet et réseaux de prochaine génération
Série Z	**Langages et aspects généraux logiciels des systèmes de télécommunication**

Imprimé en Suisse
Genève, 2006