5i’

Annexes A, B, C and E

to Recommendation Z.100

FUNCTIONAL SPECIFICATION AND

DESCRIPTION LANGUAGE (SDL)

MONTAGE: PAGE 206 = PAGE BLANCHE

ANNEX A
(to Recommendation Z.100)

SDL Glossary

The Z.100 Recommendation contains the formal definitions of SDL terminology. The SDL Glossary is compiled to help new SDL users when reading the Recommendation and its annexes, giving a brief definition and reference to the defining section of the Recommendation. The definitions in the Glossary may summarize or paraphrase the formal definitions, and thus may be incomplete.

Terms which are in a definition may also be found in the glossary. If an italicized phrase, for example *procedure identifier* , is not in the glossary, then it may be the concatenation of two terms, in this case the term *procedure* followed by the term *identifier* . When a word is in italics but cannot be located in the glossary, it may be a derivative of a glossary term. For example, *exported* is the past tense of *export* .

Except where a term is a synonym for another term, after the definition of the term there is a main reference to the use of the term in the Z.100 Recommendation. These references are shown in square brackets [ ] after definitions. For example, [3.2] indicates that the main reference is in § 3.2.

abstract data type

F **|** *type abstrait de donn´ees*

S **|** *tipo abstracto de datos*

Abstract data type is a synonym for *data type* . All *SDL data types* are *abstract data types* .

abstract grammar

F **|** *grammaire abstraite*

S **|** *gram´atica abstracta*

The *abstract grammar defines the semantics* of *SDL* . The *abstract grammer* is described by the *abstract syntax* and the *well-formedness rules* . [1.2, 1.4.1]

abstract syntax

F **|** *syntaxe abstraite*

S **|** *sintaxis abstracta*

The *abstract syntax* is the means to describe the conceptual structure of an *SDL specification* as compared with the *concrete syntaxes* which exist for each *concrete syntax* of *SDL* , this is *SDL/GR* and *SDL/PR* . [1.2]

access

F **|** *acc`es*

S **|** *acceder*

Access is the operation applied to a *variable* which gives the *value* which was last assigned to it. If a *variable* is *accessed* which has an *undefined value* , then an *error* occurs.

action

F **|** *action*

S **|** *acci´on*

An *action* is an operation which is executed within a *transition string* , e.g., a *task, output, decision, create request* or *procedure call* . [2.7]

active timer

F **|** *temporisateur actif*

S **|** *temporizador activo*

An *active timer* is a *timer* which has a *timer signal* in the *input port* of the owning *procedure* or is scheduled to produce a *timer signal* at some future time. [2.8.2, 5.5.4.5]

actual parameter

F | param`etre r´eel

S | par`ametro efectivo

An *actual parameter* is an *expression* given to a *process* or *procedure* for the corresponding *formal parameter* when the *process* or *procedure* is *created* (or *called*). Note that in certain cases in a procedure call an *actual parameter* must be a *variable* (i.e. a particular type of *expression* ; see *IN/OUT*). [2.7.2, 2.7.3, 4.2.2]

actual parameter list

F | liste de param`etres r´eels

S | lista de par`ametros efectivos

An *actual parameter list* is the list of *actual parameters* . The *actual parameters* are matched by position with the respective elements of the corresponding *formal parameter list* .

area

F | zone

S | ´area; zona

An *area* is a two dimensional region in the *concrete graphical syntax* . *Area* often correspond to *nodes* in the *abstract syntax* and usually contain *common textual syntax* . In *interaction diagrams* areas may be connected by *channels* or *signal routes* . In *control flow diagrams areas* may be connected by *flow lines* .

array

F | tableau (array)

S | matriz

Array is the predefined *generator* used to introduce the concept of arrays, easing the definition of arrays.

assign

F | affectation

S | asignar

Assign is the operation applied to a *variable* which associates a *value* to the *variable* replacing the previous *value* associated with the *variable* . [5.5.3]

assignment statement

F | instruction d'affectation

S | sentencia de asignaci´on

An *assignment statement* is a *statement* which assigns a *value* to a *variable* . [5.5.3]

association area

F **|** *zone d'association*

S **|** *´area de asociaci´on*

An *association area* is a connection between *areas* in an *interaction diagram* by means of an association *symbol* . There are five asociation areas: *channel substraction association area, input association area* , priority *input association area, continuous signal association area* and *save association area* . [2.6.3, 3.2.3, 4.10.2, 4.11]

axiom

F **|** *axiome*

S **|** *axioma*

An *axiom* is a special kind of *equation* with an implied equivalence to the *Boolean literal* True. ‘‘*Axioms* ’’ is used as a synonym for ‘‘*axioms* and *equations* .’’ [5.1.3]

basic SDL

F **|** *LDS de base*

S **|** *LED b´asico*

Basic SDL is the subset of *SDL* defined in § 2 of Recommendation Z.100.

behaviour

F **|** *comportement*

S **|** *comportamiento*

The *behaviour* or *functional behaviour* of a *system* is the set of sequences of responses to sequences of stimuli. [1.1.3]

block

F **|** *bloc*

S **|** *bloque*

A *block* is part of a *system* or parent *block* . When used by itself, *block* is a synonym for a *block instance* . A *block* is a *scope unit* and provides a static interface. [2.4.3]

block area

F **|** *zone de bloc*

S **|** *´area de bloque*

The *block area* is the definition of a *block* or a reference to a *block* in an *interaction diagram* . [2.4.2]

block definition

F **|** *d´efinition de bloc*

S **|** *definici´on de bloque*

A *block definition* is the definition of a *block* in *SDL/PR*

block diagram

F **|** *diagramme de bloc*

S **|** *diagrama de bloque*

The *block diagram* is the definition of a *block* in *SDL/GR*

block substructure

F **|** *sous-structure de bloc*

S **|** *subestructura de bloque*

A *block substructure* is the *partitioning* of the *block* into *subblocks* and new *channels* at a lower *level of abstraction* [3.2.2]

block substructure definition

F **|** *d´efinition de sous-structure de bloc*

S **|** *definici´on de subestructura de bloque*

A *block substructure definition* is the *SDL/PR* representation of a *block substructure* for a partitioned block. [3.2.2]

block substructure diagram

F | diagramme de sous-structure de bloc

S | diagrama de subestructura de bloque

A *block substructure diagram* is the *SLD/GR* representation of a *block substructure* for a *partitioned block* . [3.2.2]

block tree diagram

F | diagramme d'arborescence de bloc

S | diagrama de ´arbol de bloques

A *block tree diagram* is an auxiliary document in *SDL/GR* representing the *partitioning* of a *system* into *blocks* at lower *levels of abstraction* by means of an inverted tree diagram (*i.e.* , parent *block* at the top). [3.22]

BNF (Backus-Naur Form)

F | forme BNF (Backus-Naur Form)

S | FBN (forma Backus-Naur)

BNF (Backus-Naur Form) is a formal notation used for expressing the *concrete textual syntax* of a language. An extended form of *BNF* is used for expressing the *concrete graphical grammar* . [1.5.2, 1.5.3]

Boolean

F | bool´een

S | booleano

Boolean is a *sort* defined in a predefined *partial type definition* and has the *values* True and False. For the *sort Boolean* the predefined *operators* are NOT, AND, OR, XOR and implication. [5.6.1]

channel

F | canal

S | canal

A *channel* is the connection conveying *signals* between two *blocks* . *Channels* also convey *signals* between a *block* and the *environment* . *Channels* may be unidirectional or bidirectional. [2.5.1]

channel definition

F | d´efinition

S | *definici´on de canal*

A *channel definition* is the definition of a *channel* in *SDL/PR* . [2.5.1]

channel definition area

F | *zone de d´efinition de canal*

S | *´area de definici´on de canal*

The *channel definition area* is the definition of a *channel* in *SDL/GR* . [2.5.1]

channel substructure

F | *sous-structure de canal*

S | *subestructura de canal*

A *channel substructure* is a *partitioning* of a *channel* into a set of *channels* and *blocks* at a *lower level of abstraction* [3.2.3]

channel substructure definition

F **|** *d´efinition de sous-structure de canal*

S **|** *defínici´on de subestructura de canal*

A *channel substructure definition* is the definition of the *channel substructure* in *SDL/PR* . [3.2.3]

channel substructure diagram

F **|** *diagramme de sous-structure de canal*

S **|** *diagrama de subestructura de canal*

A *channel substructure diagram* is the definition of the channel substructure in *SDL/GR* . [3.2.3]

character

F **|** *caract`ere (character)*

S **|** *car´acter; character*

Character is a *sort* defined in a predefined *partial type definition* for which the *values* are the elements of the CCITT No. 5 alphabet, (e.g., 1, A, B, C, etc.). For the *sort character* the *ordering operators* are predefined. [5.6.2]

chartstring

F **|** *cha* **|** *ne de caract`eres (character string)*

S **|** *cadena-de-caracteres; chartstring*

Chartstring is a *sort* defined in a predefined *partial type definition* for which the *values* are *strings* of *characters* and the *operators* are those of the *string* predefined *generator* instantiated for *characters* . [5.6.4]

comment

F **|** *commentaire*

S **|** *comentario*

A *comment* is information which is in addition to or clarifies the *SDL specification* . In *SDL/GR comments* may be attached by a dashed line to any *symbol* . In *SDL/PR comments* are introduced by the keyword *COMMENT* . *Comments* have no *SDL* defined meaning. See also *Note* . [2.2.6]

common textual grammar

F **|** *grammaire textuelle commune*

S **|** *gram´atica textual com´un*

The *common textual grammar* is the subset of the *concrete textual grammar* which applies to both *SDL/GR* and *SDL/PR* . [1.2]

communication path

F **|** *trajet de communication*

S **|** *trayecto de comunicaci´on*

A *communication path* is a transportation means that carriers *signal instances* from one *process instance* or from the *environment* to another *process instance* or to the *environment* . A *communication path* comprises either *channel* path(s) or *signal route* path(s) or a combination of both. [2.7.4]

complete valid input signal set

F **|** *ensemble complet de signaux d'entr´ee valides*

S **|** *conjunto completo de señales de entrada v´alidas*

The *complete valid input signal set* of a *process* is the union of the *valid input signal* set, the local *signals, timer signals* and the *implicit signals* of the *process* . [2.4.4]

concrete grammar

F | grammaire concr`ete

S | gram´atica concreta

A *concrete grammar* is the *concrete syntax* along with the *well-formedness rules* for that *concrete syntax* . *SDL/GR* and *SDL/PR* are the *concrete grammars* of *SDL* . The *concrete grammars* are mapped to the *abstract grammar* to determine their *semantics* . [1.2]

concrete graphical grammar

F | grammaire graphic concr´ete

S | gram´atica gr´afica concreta

The *concrete graphical grammar* is the *concrete grammar* for the graphical part of SDL/GR.

concrete graphical syntax

F | syntaxe graphique concr`ete

S | sintaxis gr´afica concreta

The *concrete graphical syntax* is the *concrete syntax* for the graphical part of *SDL/GR* . The *concrete graphical syntax* is expressed in Z.100 using an extended form of *BNF* . [1.2, 1.5.3]

concrete syntax

F | syntaxe concr`ete

S | sintaxis concreta

The *concrete syntax* for the various representations of *SDL* is the actual *symbols* used to represent *SDL* and the interrelationship between *symbols* required by the syntactic rules of *SDL* . The two *concrete syntaxes* used in Z.100 are the *concrete graphical syntax* and the *concrete textual syntax* . [1.2]

concrete textual syntax

F | syntaxe textuelle concr`ete

S | sintaxis textual concreta

The *concrete textual syntax* is the *concrete syntax* for *SDL/PR* and the textual parts of *SDL/GR* . The *concrete textual syntax* is expressed in Z.100 using *BNF* . [1.2, 1.5.2]

conditional expression

F | expression conditionnelle

S | *expresi´on condicional*

A *conditional expression* is an *expression* containing a *Boolean expression* which controls whether the consequence *expression* or the alternative *expression* is interpreted. [5.5.2.3]

connect

F | *connect*

S | *conectar*

Connect indicates the connection of a *channel* to one or more *signal routes* . [2.5.3]

connector

F | *connecteur*

S | *conector*

A *connector* is an *SDL/GR symbol* which is either an *in-connector* or an *out-connector* . A *flow line* is implied from *out-connectors* to the associated *in-connector* in the same *process* or *procedure* identified by having the same *name* . [2.6.6]

consistent partitioning subset

F **|** *sous-ensemble de subdivision coh´erent*

S **|** *subconjunto de partici´on consistente*

A *consistent partitioning subset* is a set of the *blocks* and *subblocks* in a *system specification* which provides a complete view of the *system* with related parts at a corresponding *level of abstraction* . Thus, when a *block* or *subblock* is contained in a *consistent partitioning subset* , its ancestors and siblings are too. [3.2.1]

consistent refinement subset

F **|** *sous-ensemble de raffinement coh´erent*

S **|** *subconjunto de refinamiento consistente*

The *consistent refinement subset* is a *consistent partitioning subset* which contains all *blocks* and *subblocks* which use the *signals* used by any of the *blocks* or *subblocks*

continuous signal

F **|** *signal continu*

S **|** *señal continua*

A *continuous signal* is a means to define that when in a *state* the associated *Boolean* condition becomes True, the *transition* following the *continuous signal* is interpreted. [4.11]

control flow diagram

F **|** *diagramme de liaison de contr* **|** *le*

S **|** *diagrama de flujo de control*

A *control flow diagram* is either a *process diagram* , a *procedure diagram* , or a *service diagram* .

create

F **|** *cr´eer*

S **|** *crear*

Create is a synonym for *create request* .

create request

F **|** *demande de cr´eation*

S **|** *petici´on de crear*

A *create request* is the *action* causing the creation and starting of a new *process instance* using a specified *process type* as a template. The *actual parameters* in the *create request* replace the *formal parameters* in the *process* [2.7.2]

create line area

F **|** *zone de ligne de création*

S **|** *área de línea de crear*

The *create line area* in a *block diagram* connects the *process area* of the *creating (PARENT) process* with the *process area* of the *created (OFFSPRING) process* [2.4.3]

data type

F **|** *type de données*

S **|** *tipo de datos*

A *data type* is the definition of sets of *values (sorts)* , a set of *operators* which are applied to these *values* and a set of algebraic rules (*equations*) defining the *behaviour* when the *operators* are applied to the *values* . [2.3.1]

data type definition

F **|** *d´efinition de type de donn´ees*

S **|** *definici´on de tipo de datos*

A *data type definition* defines the validity of *expressions* and relationship between *expressions* at any given point in an *SDL specification* . [5.2.1]

decision

F **|** *d´ecicion*

S **|** *decisi´on*

A *decision* is an *action* within a *transition* which asks a question to which the answer can be obtained at that instant and accordingly chooses one of the several outgoing *transitions* from the *decision* to continue interpretation. [2.7.5]

decision area

F **|** *zone de d´ecision*

S **|** *´area de decisi´on*

A *decision area* is the *SDL/GR* representation of a *decision*

default

F **|** *d´efault*

S **|** *por defecto*

The *default assignment* is a denotation of a *value* that is initially associated to each *variable* of the *sort* of the *default* clause. The *default* clause may appear in *data type definitions* [5.5.3.3]

description

F **|** *description*

S **|** *descripci´on*

A *description* of a *system* is the description of its actual *behaviour* . [1.1]

diagram

F **|** *diagramme*

S **|** *diagrama*

A *diagram* is the *SDL/GR* representation for a part of a *specification* . [2.4.2]

duration

F **|** *dur´ee (duration)*

S **|** *duraci´on; duration*

Duration is a *sort* defined in a predefined *partial type definition* for which the *values* are denoted as *reals* and represent the interval between two time instants. [5.6.11]

enabling condition

F **|** *condition de validation*

S **|** *condici´on habilitante (o habilitadora)*

An *enabling condition* is a means for conditionally accepting a *signal* for *input* . [4.12]

enabling condition area

F | *zone de condition de validation*

S | *área de condición habilitante (o habilitadora)*

The *enabling condition area* is the *SDL/GR* representation of an *enabling condition* . [4.12]

entity class

F | *classe d'entité*

S | *clase de entidad*

An *entity class* is a categorization of *SDL types* based on similarity of use. [2.2.2]

environment

F | *environnement*

S | *entorno*

The term *environment* is a synonym for the *environment of a system* . Also when context allows, it may be a synonym for the *environment* of a *block, process, procedure* or a *service* . [1.3.2]

environment of a system

F | *environnement d'un système*

S | *entorno de un sistema*

The *environment of a system* is the external world of the *system* being specified. The environment interacts with the *system* by sending/receiving *signal instances* to/from the *system* . [1.3.2]

equation

F | *équation*

S | *ecuación*

An *equation* is a relation between *terms* of the same *sort* which holds for all possible *values* substituted for each *value identifier* in the *equation* . An *equation* may be an *axiom*

error

F | *erreur*

S | *error*

An *error* occurs during the interpretation of a *valid specification* of a *system* when one of the dynamic conditions *SDL* is violated. Once an *error* has occurred, the subsequent *behaviour* of the *system* is not defined by *SDL* . [1.3.3]

export

F **|** *export*

S **|** *exportaci´on*

The term *export* is a synonym for *export operation* .

exported variable

F **|** *variable export´ee*

S **|** *variable exportada*

An *exported variable* is a *variable* which can be used in an *export operation* . [4.13]

exporter

F **|** *exportateur*

S **|** *exportador*

An *exporter* of a *variable* in the *process instance* which owns the *variable* and *exports* its *values* . [4.13]

export operation

F | *op´eration d'exportation*

S | *operaci´on de exportaci´on*

An *export operation* is the operation by which the *exporter* discloses the *value* of a *variable* . See *import operation*

expression

F | *expression*

S | *expresi´on*

An *expression* is either a *literal* , an *operator* application, a *synonym* , a *variable access* , a *conditional expression* , or an *imperartive operator* applied to one or more *expressions* . When an *expression* is interpreted a *value* is obtained (or the *system* is in *error*). [2.3.4, 5.4.2.1]

external synonym

F | *synonyme externe*

S | *sin´onimo externo*

An *external synonym* of a predefined *sort* whose *value* is not specified in the *system specification* . [4.3.1]

extract!

F | *extract!*

S | *extraer!; extract!*

Extract is an *operator* which is implied in an *expression* when a *variable* is immediately followed by bracketed *expression(s)* [5.4.2.4, 5.6.8]

flow line

F | *ligne de liaison*

S | *l´inea de flujo*

A *flow line* is a *symbol* used to connect *areas* in a *control flow diagram* . [2.2.4, 2.6.7.2.2]

formal parameter

F | *param`etre formel*

S | *par´ametro formal*

A *formal parameter* is a *variable name* to which *actual values* are assigned or which are replaced by *actual variables* . [2.4.4, 2.4.5, 4.2, 4.10]

formal parameter list

F **|** *liste de param`etres formels*

S **|** *lista de par´ametros formales*

A *formal parameter list* is list of a *formal parameters* .

functional behaviour

F | comportement fonctionnel

S | comportamiento funcional

Functional behaviour is a synonym for *behaviour* .

general option area

F: zone d'option g´en´erale

S: ´area de opci´on general

The *general option area* is the *SDL/GR* representation of an *option* . [4.3.3]

general parameters

F | param`etres g´en´eraux

S | par´ametros generales

The *general parameters* in both a *specification* and a *description* of a *system* relate to such matters as temperature limits, construction, exchange capacity, grade of service, etc., and are not defined in *SDL* . [1.1]

generator

F | g´en´erateur

S | generador

A *generator* is an incomplete *newtype* description. Before it assumes the status of a *newtype* , a *generator* must be instantiated by providing the missing information. [5.4.1.1.2]

graph

F | graphe

S | gr´afico

A *graph* in the *abstract syntax* is a part of an *SDL specification* such as *procedure graph* or a *process graph* .

ground expression

F | expression close

S | expresi´on fundamental

A *ground expression* is an *expression* containing only *operators* , *synonyms* and *literals* . [5.4.2.2]

hierarchical structure

F **|** *structure hi´erarchique*

S **|** *estructure jer`arquica*

A *hierarchical structure* is a structure of a *system specification* where *partitioning* and *refinement* allow different views of the *system* at different *levels of abstraction. Hierarchical structures* allow the management of complex *system specifications* . See also *block tree diagram* . [3.1]

identifier

F **|** *identificateur*

S **|** *identificador*

An *identifier* is the unique identification of an object, formed from a *qualifier part* and a *name* . [2.2.2]

imperative operator

F | *op´erateur imp´eratif*

S | *operador imperativo*

An *imperative operator* is a now *expression, view expression, timer active expression, import expression* or one of the *PId expressions: SELF, PARENT, OFFSPRING* or *SENDER* . [5.5.4]

implicit transition

F | *transition implicite*

S | *transici´on impl´icita*

An *implicit transition* is in the *concrete syntax* initiated by a *signal* in the *complete valid input signal set* and not specified in an *input* or *save* for the *state* . An *implicit transition* contains no *action* and leads directly back to the same *state* [4.6]

import

F | *import*

S | *importaci´on*

The term *import* is a synonym for *import operation* . [4.13]

imported variable

F | *variable import´ee*

S | *variable importada*

An *imported variable* is a *variable* used in an *import operation* . [4.13]

importer

F | *importeur*

S | *importador*

An *importer* of an *imported variable* is the *process instance* which *imports* the *value* . [4.13]

import operation

F | *op´eration d'importation*

S | *operaci´on de importaci´on*

An *import operation* is the operation that yields *value* of an *exported variable* . [4.13]

IN variable

F **|** *variable ‘‘IN’’*

S **|** *variable IN*

An *IN variable* is a *formal parameter* attribute denoting the case when a *value* is passed to a *procedure* via an *actual parameter* . [2.4.5]

IN/OUT variable

F **|** *variable ‘‘IN/OUT’’*

S **|** *variable IN/OUT*

An *IN/OUT variable* is a *formal parameter* attribute denoting the case when a *formal parameter name* is used as a synonym for the *variable* (i.e. the *actual parameter* must be a *variable*

in-connector

F | *connecteur d’entr´ee*

S | *conector de entrada*

An *in-connector* is a *connector* .

infix operator

F | *op´erateur infixe*

S | *operador infijo*

An *infix operator* is one of the predefined dyadic *operators* of *SDL* (=>, OR, XOR, AND, IN, /=, =, >, <, < =, >=, +, —, //, *, /, MOD, REM) which are placed between its two arguments. [5.4.1.1]

informal text

F | *texte informel*

S | *texto informal*

Informal text is text included in an *SDL specification* for which *semantics* are not defined by *SDL* , but through some other model. *Informal text* is enclosed in apostrophes. [2.2.3]

initial algebra

F | *alg`ebre initiale*

S | *´algebra inicial*

An *initial algebra* is the formalism for defining *abstract data types* . [5.3]

inlet

F | *acc`es entrant*

S | *acceso de entrada*

An *inlet* represents a line, such as a *channel* or a *flow line* , entering an *SDL/GR macro call* . [4.2.3]

input

F | *entr´ee*

S | *entrada*

An *input* is the consumption of a *signal* from the *input port* which starts a *transition* . During the consumption of a *signal* , the *values* associated with the *signal* become available to the *process instance* . [2.6.4, 4.10.2]

input area

F **|** *zone d'entr´ee*

S **|** *´area de entrada*

An *input area* is the *SDL/GR* representation of an *input* . [2.6.4]

input port

F **|** *port d'entr´ee*

S **|** *puerto de entrada*

An *input port* of a *process* is a queue which receives and retains *signals* in the order of arrival until the *signals* are consumed by an *input* . The *input port* may contain any number of *retained signals* . [2.4.4]

instance

F **|** *instance*

S **|** *instancia*

An *instance* of a *type* is an object which has the properties of the *type* (given in the definition). [1.3.1]

instantiation

F **|** *instantiation*

S **|** *instanciaci´on*

Instantiation is the creation of an *instance* of a *type* . [1.3.1]

integer

F **|** *entier (integer)*

S **|** *entero; integer*

Integer is a *sort* defined in a predefined *partial type definition* for which the *values* are these of mathematical integers (. | | , —2, —1, 0, +1, +2, . | |). For the *sort integer* the predefined *operators* are +, —, *, / and the *ordering operators*

interaction diagram

F **|** *diagramme d'interaction*

S **|** *diagrama de interacci´on*

An *interaction diagram* is a *block diagram, system diagram, channel substructure diagram,* or *block substructure diagram* .

keyword

F **|** *mot cĺe*

S **|** *palabra clave*

A *keyword* is a reserved *lexical unit* in the *concrete textual syntax* . [2.2.1]

label

F **|** *´etiquette*

S **|** *etiqueta*

A *label* is a *name* followed by a colon and is used in the *concrete textual syntax* for connection purposes. [2.6.6]

level

F | *niveau*

S | *nivel*

The term *level* is a synonym for *level of abstraction* .

level of abstraction

F | *niveau d’abstraction*

S | *nivel de abstracci´on*

A *level of abstraction* is one of the levels of a *block tree diagram* . A description of a *system* is one *block* at the highest *level of abstraction* and is shown as a single *block* at the top of a *block tree diagram* . [3.2.1]

lexical rules

F | r`egles lexicales

S | reglas l´exicas

Lexical rules are rules which define how *lexical units* are built from characters. [2.2.1, 4.2.1]

lexical unit

F | unit´es lexicales

S | unidad l´exica

Lexical units are the terminal *symbols* of the *concrete textual syntax* . [2.2.1]

literal

F | litt´eral

S | literal

A *literal* denotes a *value* . [2.3.3, 5.1.2, 5.4.1.14]

macro

F | macro

S | marco

A *marcro* is a named collection of syntactic or textual items, which replaces the *macro call* before the meaning of the *SDL* representation is considered (i.e., a *macro* has meaning only when replaced in a particular context). [4.2]

macro call

F | appel de macro

S | llamada a (de) macro

A *macro call* is an indication of a place where the *macro definition* with the same *name* should be expanded. [4.2.3]

macro definition

F | d´efinition de macro

S | definici´on de macro

A *macro definition* is the definition of a *macro* in *SDL/PR*

macro diagram

F | *diagramme de macro*

S | *diagrama de macro*

A *macro diagram* is the definition of a *macro* in *SDL/GR* . [4.2.2]

make!

F | *make!*

S | *hacer!; make!*

Make! is an operation only used in *data type* definitions fo form a *value* of a complex type (e.g., *structured sort*). [5.4.1.10, 5.6.8]

merge area

F | *zone de fusion*

S | *área de fusión*

A *merge area* is where one *flow line* connects to another. [2.6.7.2.2]

Meta IV

F | *Meta IV*

S | *Meta IV*

Meta IV is a formal notation for expressing the *abstract syntax* of a language. [1.5.1]

model

F | *modèle*

S | *modelo*

A *model* gives the mapping for *shorthand* notations expressed in terms of previously defined *concrete syntax* . [1.4.1, 1.4.2]

modify!

F | *modify!*

S | *modificar!; modify!*

Modify is an *operator* which is implied in *expressions* when a *variable* is immediately followed by bracketed expressions and then :=. Within axioms *modify!* is used explicitly (see *extract!*) [5.4.1.10, 5.6.8]

name

F | *nom*

S | *nombre*

A *name* is a *lexical unit* used to name *SDL* objects. [2.2.1, 2.2.2]

natural

F | *naturel*

S | *natural*

Natural is a *syntype* defined in a predefined *partial type definition* for which the *values* are the non-negative integers (i.e., 0, 1, 2, | | |). The *operators* are the *operators* of the *sort integer* . [5.6.6]

newtype

F **|** *nouveau type (newtype)*

S **|** *niotipo*

A *newtype* introduces a *sort* , a set of *operators* , and a set of *equations* . Note that the term *newtype* might be confusing because actually a new *sort* is introduced, but *newtype* is maintained for historical reasons. [5.2.1]

node

F **|** *noeud*

S **|** *nodo*

In the *abstract syntax* , a *node* is a designation of one of the basic concepts of *SDL* **|**

note

F: note

S: nota

A *note* is text enclosed by /* **|** nd **|** / which has no *SDL* defined semantics. See *comment* . [2.2.1]

null

F **|** *null*

S **|** *null; nulo*

Null is the *literal* of *sort PId* . [5.6.10]

OFFSPRING

F **|** *DESCENDANT (OFFSPRING)*

S **|** *OFFSPRING; VASTAGO*

OFFSPRING is an *expression* of *sort PId* . When *OFFSPRING* is evaluated in a *process* it gives the *PId-values* of the *process* most recently *created* by this *process* . If the *process* has not *created* any *processes* , the result of the evaluation of *OFFSPRING* is *null* [2.4.4, 5.5.4.3]

operator

F **|** *op´erateur*

S **|** *operador*

An *operator* is a denotation for an operation. *Operators* are defined in a *partial type definition* . For example +, —, *, /, are *names* for *operators* defined for *sort integer* . [5.1.2, 5.1.3]

operator signature

F **|** *signature d'op´erateur*

S **|** *signatura de operador*

An *operator signature* defines the *sort(s)* of the *values* to which the *operator* can be applied and the *sort* of the resulting *value* . [5.2.2]

option

F **|** *option*

S **|** *opci´on*

An *option* is a *concrete syntax* construct in a generic *SDL system specification* allowing different *system* structures to be chosen before the *system* is interpreted. [4.3.3, 4.3.4]

ordering operators

F **|** *op´erateurs de relation d'ordre*

S **|** *operadores de ordenaci´on*

The *ordering operators* are <, <=, > or >=. [5.4.1.8]

out connector

F **|** *connecteur de sortie*

S **|** *conector de salida*

An *out-connector* is a connector.

outlet

F | acc`es sortant

S | acceso de salida

An *outlet* represents a line, such as a *channel* or *flow line* , existing a *macro diagram* . [4.2.2]

output

F: sortie

S: salida

An *output* is an *action* within a *transition* which generates a *signal instance* .

output area

F | zone de sortie

S | ´area de salida

The *output area* in a *control flow diagram* represents the *SDL/GR* concept of an *output* . [2.7.4]

page

F | page

S | p´agina

A *page* is one of the components of a physical partitioning of a *diagram*

PARENT

F | PARENT

S | PARENT; PROGENITOR

PARENT is a *PId expression* . When a *process* evaluates this *expression* , the result is the *PId-value* of the parent *process process* was created at *system* initialization time, the result is *null* . [2.4.4, 5.5.4.3]

partial type definition

F | d´efintiion partielle de type

S | definici´on parcial de tipo

The *partial type definition* for a *sort* defines some of the properties related to the *sort* . A *partial type definition* is part of a *data type* definition. [5.2.1]

partitioning

F | *subdivision*

S | *partici´on*

Partitioning is the subdivision of a unit into smaller components which when taken as a whole have the same *behaviour* as the original unit. *Partitioning* does not affect the static interface of a unit. [3.1, 3.2]

PId

F | *PId*

S | *PId*

PId is a *sort* defined in a predefined *partial type definition* for which there is one *literal* , *null* . *PId* is an abbreviation for process instance identifier, and the *values* of the *sorts* are used to identify *process instances* . [5.5.4.3, 5.6.10]

powerset

F | *mode ensembliste*

S | *conjunista*

Powerset is the *predefined generator* used to introduce mathematical sets. The *operators* for *powerset* are IN, Incl, Del, union, insersection and the *ordering operators* . [5.6.9]

predefined data

F | *donn´ees pr´ed´efinies*

S | *datos predefinidos*

For simplicity of description the term *predefined data* is applied to both predefined *names* for *sorts* introduced by *partial type definitions* and predefined *names* for *data type generators character, chartstring, duration, integer, natural PId, real* and *time* are *sort names* which are predefined. *Array, powerset* , and *string* are *data type generator names* which are predefined. *Predefined data* are defined implicitly at *system level* in all *SDL systems* . [5.6]

procedure

F | *proc´edure*

S | *procedimiento*

A *procedure* is an encapsulation of the *behaviour* of a *process* . A *procedure* is defined in one place but may be referred to several times within the same *process* . See *formal parameter* and *actual parameter* . [2.4.5]

procedure call

F | *appel de proc´edure*

S | *llamada a (de) procedimiento*

A *procedure call* is the invocation of a named *procedure* for interpretation of the *procedure* and passing *actual parameters* to the *procedure* . [2.7.3]

procedure call area

F | *zone d'appel de proc´edure*

S | *´area de llamada a (de) procedimiento*

The *procedure call area* is the *SDL/GR* representation of a *procedure call* . [2.7.3]

procedure definition

F | *d´efinition de proc´edure*

S | *definici´on de procedimiento*

A *procedure definition* is the *SDL/PR* definition of a *procedure* . [2.4.5]

procedure diagram

F | *diagramme de proc´edure*

S | *diagrama de procedimiento*

A *procedure diagram* is the *SDL/GR* representation of a *procedure* . [2.4.5]

procedure graph

F **|** *graphe de proc´edure*

S **|** *gr´afico de procedimiento*

A *procedure graph* is a nonterminal in the *abstract syntax* representing a *procedure* . [2.4.5]

procedure return

F **|** *retour de proc´edure*

S **|** *retorno de procedimiento*

Procedure return is a synonym for *return* .

process

F **|** *processus*

S **|** *proceso*

A *process* is a communicating extended finite state machine. Communication can take place via *signals* or shared *variables* . The *behaviour* of a *process* depends on the order of arrival of *signals* in its *input port* . [2.4.4]

process area

F **|** *zone de processus*

S **|** *´area de proceso*

A *process area* in *SDL/GR* is the representation of a *process* or a reference to a *process* in an *interaction diagram* . [2.4.3]

process definition

F **|** *d´efinition de processus*

S **|** *definici´on de processo*

A *process definition* is the *SDL/PR* representation of a *process* . [2.4.4]

process diagram

F **|** *diagramme de processus*

S **|** *diagrama de proceso*

A *process diagram* is the *SDL/GR* representation of the definition of a *process* . [2.4.4]

process graph

F **|** *graphe de processus*

S **|** *gr´afico de proceso*

A *process graph* is nonterminal in the *abstract syntax* representing a *process* . [2.4.4]

process instance

F | *instance de processus*

S | *instancia de proceso*

A *process instance* is a dynamically created *instance* of a *process* . See *SELF, SENDER, PARENT,* and *OFFSPRING* [2.4.4]

qualifier

F | *partie qualificative (qualificatif)*

S | *calificador*

The *qualifier* is part of an *identifier* which is the extra information to the *name* part of the *identifier* to ensure uniqueness. *Qualifiers* are always present in the *abstract syntax* , but only have to be used as far as needed for uniqueness in the *concrete syntax* when the *qualifier* of an *identifier* cannot be derived from the context of the use of the *name* part. [2.2.2]

real

F | *r´eel*

S | *real*

Real is a *sort* defined in a predefined *partial type definition* for which the *values* are the numbers which can be presented by one *Integer* divided by another. The predefined *operators* for the *sort real* have the same *names* as the *operators* of *sort integer* . [5.6.7]

refinement

F | *reaffinement*

S | *refinamiento*

Refinement is the addition of new details to the funtionality at a certain *level of abstraction* . The *refinement* of a *system* causes an enrichment in its *behaviour* or its capabilities to handle more types of *signals* and information, including those *signals* to and from the *environment* . Compare with *partitioning* . [3.3]

remote definition

F | *d´efinition distante*

S | *definici´on remota*

A *remote definition* is a syntactic means of distributing a *system definition* into several parts and relating the parts to each other. [2.4.1]

reset

F | *reset (r´einitialisation)*

S | *reincializar; reponer*

Reset is an operation defined for *timers* which allows *timers* to be made inactive. See *active timer* . [2.8]

retained signal

F | *signal retenu*

S | *señal retenida*

A *retained signal* is a *signal* in the *input port* of a *process* , i.e., a *signal* which has been received but not consumed by the *process* . [2.4.4]

return

F | *retour*

S | *retorno*

The *return* of a *procedure* is the transfer of control to the calling *procedure* or *process* . [2.6.7.2.4]

reveal attribute

F | *attribut d'exposition*

S | *atributo revelado*

A *variable* owned by a *process* may have a *reveal attribute* , in which case another *process* in the same *block* is permitted to view the *value* associated with the *variable* . See *view definition* . [2.6.1.1]

save

F **|** *mise en r´eserve*

S **|** *conservaci´on*

A *save* is the declaration of those *signals* that should not be consumed in a given *state* . [2.6.5]

save area

F **|** *zone de mise en r´eserve*

S **|** *´area de conservaci´on*

The *save area* is the *SDL/GR* representation of a *save* . [2.6.5]

save signal set

F **|** *ensemble de signaux de mise en r´eserve*

S **|** *conjunto de señales de conservaci´on*

The *save signal set* of a *state* is the set of saved *signals* for that *state* . [2.6.5]

SDL (CCITT Specification and Description Language)

F **|** *LDS (langage de description et de sp´ecification du CCITT)*

S **|** *LED (lenguaje de especificaci´on y descripci´on del CCITT)*

CCITT *SDL* (*Specification and Description Language*) is a formal language providing a set of constructs of the *specification* for the functionality of a system.

SDL/GR

F: LDS/GR

S: LED/GR

SDL/GR is the graphical representation in *SDL* . The *grammar* for *SDL/GR* is defined by the *concrete graphical grammar* and the *common textual grammar* . [1.2]

SDL/PE

F **|** *LDS/PE*

S **|** *LED/EP*

SDL/PE is a set of icons which can be used in conjunction with the *state symbol* of *SDL/GR* . [Annex E]

SDL/PR

F **|** *LDS/PR*

S **|** *LED/PR*

SDL/PR is the textual phrase representation in *SDL* . The *grammar* for *SDL/PR* is defined by the *concrete textual grammar* [1.2]

scope unit

F **|** *unit´e de port´ee*

S **|** *unidad de ´ambito*

A *scope unit* in the *concrete grammar* defines the range of *visibility* of *identifiers* . Examples of *scope units* include the *system, block, process, procedure, partial type definitions* and *service definitions* . [2.2.2]

selection

F **|** *s´election*

S **|** *selecci´on*

Selection means providing those *external synonyms* needed to make a specific *system specification* from a generic *system specification*

SELF

F **|** *SELF*

S **|** *SELF; MISMO*

SELF is a *PId expression* . When a *process* evaluates this *expression* , the result is the *PId-value* of that *process* results in the *value Null* . See also *PARENT, OFFSPRING, PId* . [2.4.4, 5.5.4.3]

semantics

F | s´emantique

S | sem´antica

Semantics gives meaning to an entity: the properties it has, the way its *behaviour* is interpreted, and any dynamic conditions which must be fulfilled for the *behaviour* of the entity to meet *SDL* rules. [1.4.1, 1.4.2]

SENDER

F | SENDER (´emetteur)

S | SENDER; EMISOR

SENDER is a *PId expression* . When evaluated *SENDER* yields the *PId value* of the sending *process* of the *signal* that activated the current *transition* . [2.4.4, 2.6.4, 5.5.4.3]

service

F | service

S | servicio

A *service* is an alternative way of specifying a *process* . Each *service* may define a partial *behaviour* of a *process* . [4.10]

service area

F | zone de service

S | ´area de servicio

A *service area* is either a *service diagram* or a reference to a *service* . [4.10.1]

service definition

F | d´efinition de service

S | definici´on de servicio

A *service definition* is the *SDL/PR* definition of a *service*

service diagram

F | diagramme de service

S | diagrama de servicio

A *service diagram* is the *SDL/GR* definition of a *service* . [4.10]

set

F **|** *set (initialisation)*

S **|** *inicializar; poner*

Set is an operation defined for *timers* which allow *timers* to be made *active* . [2.8]

shorthand notation

F | *notation abr´eg´ee*

S | *notaci´on taquigr´afica (o abreviada)*

A *shorthand notation* is a *concrete syntax* notation providing a more compact representation implicitly referring to *Basic SDL* concepts. [1.4.2]

signal

F | *signal*

S | *señal*

A *signal* is an instance of a signal *type* communication information to a *process instance* . [2.5.4]

signal definition

F | *d´efinition de signal*

S | *definici´on de señal*

A *signal definition* defines a *named signal type* and associates a list of zero or more *sort identifiers* with the *signal name signals* to carry *values* . [2.5.4]

signal list

F | *liste de signaux*

S | *lista de señales*

A *signal list* is a list of *signal identifiers* used in *channel* and *signal route definitions* to indicate all the *signals* which may be conveyed by the *channel* or *signal route* in one direction. [2.5.5]

signal list area

F | *zone de liste de signaux*

S | *´area de lista de señales*

The *signal list area* in an *interaction diagram* represents a *signal list* associated with a *channel* or *signal route* . [2.5.5]

signal route

F | *acheminement de signaux*

S | *ruta de señales*

A *signal route* indicates the flow of *signals* between a *process type* and either another *process type* in the same *block* or the *channels* connected to the *block* . [2.5.2]

simple expression

F **|** *expression simple*

S **|** *expresi´on simple*

A *simple expression* is an *expression* which only contains *operators*, *synonyms* , and *literals* of the predefined *sorts* . [4.3.2]

sort

F **|** *sorte*

S **|** *g´enero*

A *sort* is a set of *values* with common characteristics. *Sorts* are always nonempty and disjoint. [2.3.3, 5.1.3]

specification

F | *sp´ecification*

S | *especificaci´on*

A *specification* is a definition of the requirements of a *system* . A *specification* consists of *general parameters* required of the *system* and the *functional specification* of its required *behaviour Specification* may be also used as a shorthand for ‘‘*specification* and/or *description* ’’, e.g., in *SDL specification* or *system specification* [1.1]

start

F | *d´epart*

S | *arranque*

The *start* in a *process* is interpreted before any *state* or *action* . The *start* initializes the *process* by replacing its *formal parameters* by the *actual parameters* as specified in the *create* [2.6.2]

state

F | *´etat*

S | *estado*

A *state* is a condition in which a *process instance* can consume a *signal* . [2.6.3]

state area

F | *zone d'´etat*

S | *´area de estado*

A *state area* is the *SDL/GR* representation of one or more *states* . [2.6.3]

state picture

F | *repr´esentation graphique d'´etat*

S | *pictograma de estado*

A *state picture* is a *state symbol* incorporating pictorial elements used to extend *SDL/GR* to *SDL/PE* . [Annex E]

stop

F | *arr* | *t*

S | *parada*

A *stop* is an action which terminates a *process instance* . When a *stop* is interpreted, all *variables* owned by the *process instance* are destroyed and all *retained signals* in the *input port* are no longer accessible. [2.6.7.2.3]

string

F | cha | ne (string)

S: cadena; string

String is a predefined *generator* used to introduce lists. The predefined *operators* include Length, First, Last, Substring and concatenation. [5.6.3]

structured sort

F | sorte structur´ee

S | g´enero estructurado

A *structured sort* is a *sort* with implicit *operators* and *equations* and special *concrete syntax* for these implicit *operators* fields. The *values* of the fields can be *accessed* and *modified* independently. [5.4.1.10]

subblock

F | *sous-bloc*

S | *subbloque*

A *subblock* is a *block* contained within another *block* . *Subblocks* are formed when a *block* is *partitioned* . [3.2.1, 3.2.2]

subchannel

F | *sous-canal*

S | *subcanal*

A *subchannel* is a *channel* formed when a *block* is *partitioned* . A *subchannel* connects a *subblock* to a boundary of the *partitioned block* or a *block* to the boundary of a *partitioned channel* . [3.2.2, 3.2.3]

subsignal

F | *sous-signal*

S | *subseñal*

A *subsignal* is a *refinement* of a *signal* and may be further *refined* . [3.3]

symbol

F | *symbole*

S | *s´imbolo*

A *symbol* is a terminal in the *concrete syntaxes* . A *symbol* may be one of a set of shapes in the *concrete graphical syntax* .

synonym

F | *synonyme*

S | *sin´onimo*

A *synonym* is a *name* which represents a *value* . [5.4.1.13]

syntax diagram

F | *diagramme de syntaxe*

S | *diagrama de sintaxis*

Syntax diagrams are illustrations of the definitions of the *concrete textual syntax* . [Annex C2]

syntype

F | *syntype*

S | *sintipo*

A *syntype* determines a set of *values* which corresponds to a subset of the *values* of the parent *type* . The *operators* of the *syntype* are the same as those of the parent *type* . [5.4.1.9]

system

F | *syst`eme*

S | *sistema*

A *system* is a set of *blocks* connected to each other and the *environment* by *channels* .

system definition

F **|** *d´efinition de syst`eme*

S **|** *definici´on de sistema*

A *system definition* is the *SDL/PR* representation of a *system* . [2.4.2]

system diagram

F **|** *diagramme de syst`eme*

S **|** *diagrama de sistema*

A *system diagram* is the *SDL/GR* representation of a *system*

task

F **|** *t* **|** *che*

S **|** *tarea*

A *task* is an action within a *transition* containing either a sequence of *assignment statements* or *informal text* . The interpretation of a *task* depends on and may act on information held by the *system* [2.7.1]

task area

F **|** *zone de t* **|** *che*

S **|** *´area de tarea*

A *task area* is the SDL/GR representation of a *task* . [2.7.1]

term

F **|** terme

S **|** *t´ermino*

A *term* is syntactically equivalent to an *expression* . *Terms* are only used in *axioms* and are distinguished from *expressions* for reasons of clarity. [5.2.3, 5.3.3]

text extension symbol

F **|** *symbole d'extension de texte*

S **|** *s´ibolo de ampliaci´on de texto*

A *text extension symbol* is a container of text which belongs to the *graphical symbol* to which the *text extension symbol* is attached. The text in the *text extension symbol* follows the text in the symbol to which it is attached. [2.2.7]

time

F **|** *temps (time)*

S **|** *tiempo; time*

Time is a *sort* defined in a predefined *partial type definition* for which the *values* are denoted as the *values* of *real* . The predefined *operators* using *time* and *duration* are + and—. [5.5.4.1, 5.6.12]

timer

F **|** *temporisateur*

S **|** *temporizador*

A *timer* is an object, owned by a *process instance* , that can be *active* or *inactive* . An *active timer* returns a *timer signal* to the owning *process instance* at a specified time. See also *set* and *reset* . [2.8, 5.5.4.5]

transition

F | transition

S | transici´on

A *transition* is an active sequence which occurs when a *process instance* changes from one *state* to another. [2.6.7.1]

transition area

F | zone de transition

S | `area de transici´on

A *transition area* is the *SDL/GR* representation of a *transition* . [2.6.7.1]

transition string

F | cha | ne de transition

S | cadena de transici´on

A *transition string* is a sequence of zero or more *actions* . [2.6.7.1]

transition string area

F | zone de cha | ne de transition

F | ´area de cadena de transici´on

A *transition string area* is the *SDL/GR* representation of a *transition string* . [2.6.7.1]

type

F | type

S | tipo

A *type* is a set of properties for entities. Examples of classes of *types* in *SDL* include *blocks*, *channels*, *signal routes*, *signals* , and *systems* . [1.3.1]

type definition

F | d´efinition de type

S | definici´on de tipo

A *type definition* defines the properties of a *type* [1.3.1]

undefined

F | *ind´efini (undefined)*

S | *indefinido*

Undefined is a ‘‘special’’ *value* of every *sort* which indicates that a *variable* of that *sort* has not yet been *assigned* a normal *value* . See *access* . [5.5.2.2]

valid input signal set

F | *ensemble de signaux d'entr´ee valides*

S | *conjunto de señales de entrada v´alidas*

The *valid input signal set* of a *process* is the list of all external *signals* handled by any *input* in the *process* those *signals* in *signal routes* leading to the *process complete valid input signal set* . [2.4.4, 2.5.2]

valid specification

F | *sp´ecification valide*

S | *especificaci´on v´alida*

A *valid specification* is a *specification* which follows the *concrete syntax* and static *well-formedness rules* . 1.3.3]

value

F | *valeur*

S | *valor*

A *value* of a *sort* is one of the values which are associated with a *variable* of that *sort* , and which can be used with an *operator* requiring a *value* of that *sort* . A *value* is the result of the interpretation of an *expression* . [2.3.3, 5.1.3]

variable

F | *variable*

S | *variable*

A *variable* is an entity owned by a *process instance* or *procedure instance* which can be associated with a *value* through an *assignment statement* . When *accessed* , a *variable* yields the last *value* which was assigned to it. [2.3.2]

variable definition

F | *d´efinition de variable*

S | *definici´on de variable*

A *variable definition* is the indication that the *variable names* listed will be *visible* in the *process, procedure* or *service* containing the definition. [2.6.1.1]

view definition

F | *d´efinition de visibilit´e*

S | *definici´on de visi´on*

A *view definition* defines a *variable identifier* in another *process* where it has the *revealed attribute* . This allows the viewing *process* to *access* the *value* of that *variable* . [2.6.1.2]

view expression

F | *expression de vue*

S | *expresi´on de visi´on*

A *view expression* is used within an *expression* to yield the current *value* of a *viewed variable* . [5.5.4.4]

visibility

F **|** *visibilit´e*

S **|** *visibilidad*

The *visibility* of an *identifier* is the *scope units* in which it may be used. No two definitions in the same *scope unit* and belonging to the same *entity class* may have the same *name* . [2.2.2]

well-formedness rules

F **|** *r`egles de bonne formation*

S **|** *reglas de formaci´on correcta*

Well-formedness rules are constraints on a *concrete syntax* enforcing static conditions not directly expressed by the syntax rules. [1.4.1, 1.4.2]

(Folios anglais par Montage)

Pour Montage
Fascicule X.1 — Dossier 360
(Anglais)

(FOLIOS 205 — 235: AVEC TEXTE SAISI MEP) FOLIOS 236 — 265 EXTERIEUR

236 **Fascicle X.1 — Rec. Z.100 — Annex B**

Fascicle X.1 — Rec. Z.100 — Annex B 237

238 **Fascicle X.1 — Rec. Z.100 — Annex B**

Fascicle X.1 — Rec. Z.100 — Annex B 239

240 **Fascicle X.1 — Rec. Z.100 — Annex B**

Fascicle X.1 — Rec. Z.100 — Annex B 241

242 **Fascicle X.1 — Rec. Z.100 — Annex B**

Fascicle X.1 — Rec. Z.100 — Annex B 243

244 **Fascicle X.1 — Rec. Z.100 — Annex B**