



INTERNATIONAL TELECOMMUNICATION UNION

ITU-T

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

Z.100

Corrigendum 1
(08/2004)

SERIES Z: LANGUAGES AND GENERAL SOFTWARE
ASPECTS FOR TELECOMMUNICATION SYSTEMS

Formal description techniques (FDT) – Specification and
Description Language (SDL)

Specification and Description Language (SDL)

Corrigendum 1

ITU-T Recommendation Z.100 (2002) – Corrigendum 1

ITU-T Z-SERIES RECOMMENDATIONS
LANGUAGES AND GENERAL SOFTWARE ASPECTS FOR TELECOMMUNICATION SYSTEMS

FORMAL DESCRIPTION TECHNIQUES (FDT)	
Specification and Description Language (SDL)	Z.100–Z.109
Application of formal description techniques	Z.110–Z.119
Message Sequence Chart (MSC)	Z.120–Z.129
Extended Object Definition Language (eODL)	Z.130–Z.139
Testing and Test Control Notation (TTCN)	Z.140–Z.149
User Requirements Notation (URN)	Z.150–Z.159
PROGRAMMING LANGUAGES	
CHILL: The ITU-T high level language	Z.200–Z.209
MAN-MACHINE LANGUAGE	
General principles	Z.300–Z.309
Basic syntax and dialogue procedures	Z.310–Z.319
Extended MML for visual display terminals	Z.320–Z.329
Specification of the man-machine interface	Z.330–Z.349
Data-oriented human-machine interfaces	Z.350–Z.359
Human-computer interfaces for the management of telecommunications networks	Z.360–Z.369
QUALITY	
Quality of telecommunication software	Z.400–Z.409
Quality aspects of protocol-related Recommendations	Z.450–Z.459
METHODS	
Methods for validation and testing	Z.500–Z.519
MIDDLEWARE	
Distributed processing environment	Z.600–Z.609

For further details, please refer to the list of ITU-T Recommendations.

ITU-T Recommendation Z.100

Specification and Description Language (SDL)

Corrigendum 1

Summary

This corrigendum describes a number of minor changes to ITU-T Rec. Z.100 (08/2002) and to Amendment 1 to Z.100 (10/2003) to correct and clarify some minor issues in the main body of Z.100 and in Annex B.

The changes are divided into three parts:

- Changes to the main body (and Annex D) of Z.100 published in Z.100 (08/2002);
- Replacement of Annex B.9;
- Additions to Annex B.

Each of the changes to the main body (and Annex D) of Z.100 is described with a heading stating if it is a clarification, deficiency correction, extension or modification (see Appendix II/Z.100) followed by the number of the heading in Z.100 for the text to be changed. In most cases the sub-division (*Abstract grammar*, *Concrete grammar*, *Semantics* or *Model*) in the Z.100 text is also given. The heading of each change concludes with a few descriptive words.

The changes to Annex B are to add the legacy syntax for data type definitions, which had been overlooked in the first publication of that annex in Amendment 1 to Z.100 (10/2003).

Source

Corrigendum 1 to ITU-T Recommendation Z.100 (2002) was approved on 29 August 2004 by ITU-T Study Group 17 (2001-2004) under the ITU-T Recommendation A.8 procedure.

FOREWORD

The International Telecommunication Union (ITU) is the United Nations specialized agency in the field of telecommunications. The ITU Telecommunication Standardization Sector (ITU-T) is a permanent organ of ITU. ITU-T is responsible for studying technical, operating and tariff questions and issuing Recommendations on them with a view to standardizing telecommunications on a worldwide basis.

The World Telecommunication Standardization Assembly (WTSA), which meets every four years, establishes the topics for study by the ITU-T study groups which, in turn, produce Recommendations on these topics.

The approval of ITU-T Recommendations is covered by the procedure laid down in WTSA Resolution 1.

In some areas of information technology which fall within ITU-T's purview, the necessary standards are prepared on a collaborative basis with ISO and IEC.

NOTE

In this Recommendation, the expression "Administration" is used for conciseness to indicate both a telecommunication administration and a recognized operating agency.

Compliance with this Recommendation is voluntary. However, the Recommendation may contain certain mandatory provisions (to ensure e.g. interoperability or applicability) and compliance with the Recommendation is achieved when all of these mandatory provisions are met. The words "shall" or some other obligatory language such as "must" and the negative equivalents are used to express requirements. The use of such words does not suggest that compliance with the Recommendation is required of any party.

INTELLECTUAL PROPERTY RIGHTS

ITU draws attention to the possibility that the practice or implementation of this Recommendation may involve the use of a claimed Intellectual Property Right. ITU takes no position concerning the evidence, validity or applicability of claimed Intellectual Property Rights, whether asserted by ITU members or others outside of the Recommendation development process.

As of the date of approval of this Recommendation, ITU had not received notice of intellectual property, protected by patents, which may be required to implement this Recommendation. However, implementors are cautioned that this may not represent the latest information and are therefore strongly urged to consult the TSB patent database.

© ITU 2004

All rights reserved. No part of this publication may be reproduced, by any means whatsoever, without the prior written permission of ITU.

CONTENTS

		Page
1	Changes to the main body and to Annex D of ITU-T Rec. Z.100.....	1
1.1	Extension – 6.6 – Grammar – Permit empty <page number area>	1
1.2	Deficiency correction – 7.2 – Concrete grammar – "attached" replaces "connected".....	1
1.3	Deficiency correction – 8.1.1.1 – Concrete grammar – Incorrect syntax.....	1
1.4	Clarification – 8.1.1.1 – Model – Virtuality of implied composite states	1
1.5	Deficiency correction – 8.1.1.5 – Concrete grammar – Incorrect syntax.....	1
1.6	Deficiency correction – 8.1.1.5 – Concrete grammar – Incorrect syntax.....	2
1.7	Deficiency correction – 8.1.5 – Concrete grammar – Unnecessary restriction.....	2
1.8	Deficiency correction – 8.3.1 – Concrete grammar – "attached" replaces "connected"	2
1.9	Deficiency correction – 8.5 – Concrete grammar – "attached" replaces "connected".....	2
1.10	Deficiency correction – 9 – Concrete grammar – Incorrect syntax.....	2
1.11	Deficiency correction – 9 – Concrete grammar – "attached" replaces "connected".....	3
1.12	Clarification/Deficiency correction – 9 – Semantics – Default state machine.....	3
1.13	Deficiency correction – 9.2 – Concrete grammar – Incorrect syntax.....	3
1.14	Deficiency correction – 9.3 – Concrete grammar – Incorrect syntax.....	3
1.15	Clarification – 9.5 – Concrete grammar – Incorrect syntax	3
1.16	Deficiency correction – 10.1, 10.2 – Concrete grammar – "attached" replaces "connected"	4
1.17	Clarification – 11.2 – Concrete grammar – Incorrect syntax	4
1.18	Clarification – 11.2 – Concrete grammar – Use of <composite state name>	4
1.19	Clarification – 11.2 – Model – Use of <composite state name>	4
1.20	Deficiency correction – 11.3 – Concrete grammar – Incorrect syntax.....	5
1.21	Deficiency correction – 11.6 – Concrete grammar – Incorrect syntax.....	5
1.22	Deficiency correction – 11.9 – Concrete grammar – Incorrect syntax.....	5
1.23	Clarification – 11.11 – Use of < <u>composite state</u> name>	5
1.24	Clarification – 11.11 – Abstract grammar – Procedure with states for entry/exit procedures	6
1.25	Deficiency correction – 11.11.1 – Concrete grammar – Incorrect syntax.....	6
1.26	Clarification – 11.11.1 – Concrete grammar – Use of < <u>composite state</u> name>	6
1.27	Clarification – 11.11.1 – Concrete grammar – Incorrect syntax	6
1.28	Deficiency correction – 11.11.2 – Concrete grammar – Incorrect syntax.....	6
1.29	Clarification – 11.11.2 – Concrete grammar – Use of < <u>composite state</u> name>	6

		Page
1.30	Deficiency correction – 11.11.2 – Concrete grammar – Allow gates on inherited state partition	6
1.31	Deficiency correction – 11.11.2 – Concrete grammar – Incorrect syntax.....	7
1.32	Deficiency correction – 11.11.3 – Concrete grammar – Incorrect syntax.....	7
1.33	Clarification – 11.11.4 – Concrete grammar – Incorrect syntax	7
1.34	Clarification – 11.12.2.2 – Concrete grammar – Incorrect syntax	7
1.35	Clarification – 11.13.3 – Concrete grammar – Ambiguity.....	8
1.36	Modification – 11.13.4 – Semantics – Signal routing	8
1.37	Modification – 12.1.2 – Model of implicit interfaces	8
1.38	Deficiency correction – 12.1.8 – Concrete grammar – Incorrect syntax.....	9
1.39	Clarification – 12.3.4.1 – Semantics – Unit of time	9
1.40	Clarification – 13.1 – Concrete grammar – Improved syntax	10
1.41	Clarification – D.3.11.2, D.3.12.2 – Unit of time.....	10
2	Change to B.9 (Annex B – Backwards compatibility)	10
3	Additions to Annex B – Backwards compatibility	11
B.11	Data definition	11
B.12	Data type definition	11
B.13	Syntypes	13

ITU-T Recommendation Z.100

Specification and Description Language (SDL)

Corrigendum 1

1 Changes to the main body and to Annex D of ITU-T Rec. Z.100

Each of the headings below in this clause consists of:

- a heading number, such as 1.1;
- a categorization of the change (Clarification, Deficiency correction, Extension or Modification);
- the number of the clause to be changed;
- optionally, the text sub-division (*Abstract grammar*, *Concrete grammar*, *Semantics* or *Model*);
- a brief description of the change.

1.1 Extension – 6.6 – Grammar – Permit empty <page number area>

In the grammar, *insert* "[" just before "<implicit text symbol>" and *delete* the "[" before " <page number>" to make the rule:

[<implicit text symbol> *contains* <page number> [(<number of pages>)]]

1.2 Deficiency correction – 7.2 – Concrete grammar – "attached" replaces "connected"

In the rule <package dependency area>, *change* "*connected*" to "*attached*" because the package diagram or reference area is produced by the rule for the enclosing diagram.

1.3 Deficiency correction – 8.1.1.1 – Concrete grammar – Incorrect syntax

In the rule <agent type diagram>, *insert* "{ " before "<system type diagram>", *insert* "}" after <process type diagram> and *insert* "]" after <package use area> to make the rule:

<agent type diagram> ::=
{ <system type diagram> | <block type diagram> | <process type diagram> }
[*is associated with* <package use area>]

1.4 Clarification – 8.1.1.1 – Model – Virtuality of implied composite states

To clarify that an implied state machine of an agent type can be redefined if the agent type can be redefined, the following changes are made to the text.

In 8.1.1.1 – Model

At the end of the paragraph ending "represented by the <agent body area>.", *insert* the sentence "The virtuality of the composite state definition is the same as the virtuality of the agent type."

In the paragraph following the one above, after the bullet items, *replace* "virtual implied composite state type" *by* "implied composite state type with the same virtuality as the agent type."

1.5 Deficiency correction – 8.1.1.5 – Concrete grammar – Incorrect syntax

In the rule <composite state type diagram>, *insert* the missing "}" after the item <composite state structure area>.

1.6 Deficiency correction – 8.1.1.5 – Concrete grammar – Incorrect syntax

In the rule <composite state type diagram>, *replace* "**associated with**" before { <state connection point>* } *set by* "**connected to**" because the items are joined (like gates) and the text also describes them as being "connected". At the same time, *rename* all occurrences of <state connection point> in Z.100 *to* <state connection point area> to be consistent with the graphical meta-grammar.

1.7 Deficiency correction – 8.1.5 – Concrete grammar – Unnecessary restriction

Delete the last paragraph of the clause just before Semantics containing the text:

"The <interface identifier> of a <interface gate definition> must not identify the interface implicitly defined by the entity to which the gate is connected (see 12.1.2)."

1.8 Deficiency correction – 8.3.1 – Concrete grammar – "attached" replaces "connected"

In the rule <specialization area>, *change* "connected" *to* "attached" because the type reference area is produced by the rule for the enclosing diagram.

1.9 Deficiency correction – 8.5 – Concrete grammar – "attached" replaces "connected"

The <association area> needed to be "attached" to the <linked type reference area>s because the reference area is produced by the rule for the enclosing diagram.

Replace the rule <association area> *by*:

```
<association area> ::=
    <association symbol>
    [ is associated with <association name> ]
    is attached to { <linked type reference area> <linked type reference area> } set
    is associated with { <association end area> <association end area> } set
```

and *add* the following paragraph after this rule:

"The <linked type reference area>s are attached at either end of the <association symbol> and the <association end area> nearest to the corresponding end of the <association symbol> is relevant for that <linked type reference area>."

Replace the rule <association end area> *by*:

```
<association end area> ::=
    { [<role name>] [<multiplicity>] [<ordering area>] [<symbolic visibility>] } set
```

and in the paragraphs after the syntax, *make* the following changes:

Change "If an <association end area> identifies" *to* "If an <association end area> corresponds to".

Change "is connected to a <linked type reference area>" *to* "is attached to a <linked type reference area>".

1.10 Deficiency correction – 9 – Concrete grammar – Incorrect syntax

In the rule <agent body area> the optional <on exception association area> should be independent of any <start area>.

Delete "[" before "[<on exception association area>" and *insert* "[" before <start area> to make the syntax line:

```
{ [<on exception association area>] [ <start area> ]
```


1.11 Deficiency correction – 9 – Concrete grammar – "attached" replaces "connected"

In the rule <create line area>, change "**connected**" to "**attached**" because the agent (type) or state partition is produced by the rule for the enclosing diagram.

1.12 Clarification/Deficiency correction – 9 – Semantics – Default state machine

The issue was to more clearly define the state machine for the case when there is no other explicit or implicit state machine. The word "no" was missing before "contained instances" in the second sentence:

The paragraph:

"If an agent has no explicit or implicit state machine, as soon as all the initial contained agents have been created the agent enters a stopping condition. An agent with contained initial instances and no contained state machines therefore ceases to exist as soon as it is created."

is *changed* to:

"If an agent has no other explicit or implicit state machine, there is a state machine that has just a *Stop-node*. As soon as all the initial contained agents have been created, the agent enters a stopping condition. An agent with no contained initial instances and no contained state machines therefore ceases to exist as soon as it is created."

1.13 Deficiency correction – 9.2 – Concrete grammar – Incorrect syntax

In the rule <block diagram>, the <external channel identifiers> are *associated with* the block (not "**connected to**" it).

The syntax line:

is connected to { {<gate on diagram> | <external channel identifiers>}* } *set*

is *replaced* by:

is connected to { {<gate on diagram> }* } *set*
is associated with { <external channel identifiers> }* } *set*

1.14 Deficiency correction – 9.3 – Concrete grammar – Incorrect syntax

In the rule <process diagram>, the <external channel identifiers> are *associated with* the process (not "**connected to**" it).

The syntax line:

is connected to { {<gate on diagram> | <external channel identifiers>}* } *set*

is *replaced* by:

is connected to { {<gate on diagram> }* } *set*
is associated with { <external channel identifiers> }* } *set*

1.15 Clarification – 9.5 – Concrete grammar – Incorrect syntax

In the rule <procedure body area>, *apply* {} *set* to the whole right hand side to make it clearer that the order is not important. The resulting syntax lines are:

{[<on exception association area>] [<procedure start area>]
{<state area> | <exception handler area> | <in connector area> }* } *set*

1.16 Deficiency correction – 10.1, 10.2 – Concrete grammar – "attached" replaces "connected"

The rule <channel definition area> is *connected to* agents, states or gates at either end of the channel. However, "*connected to*" means that the syntax item is produced and is therefore incorrect. The correct meta-symbol is "*attached to*". The word "*connected*" is *changed* to "*attached*" in the syntax. The revised syntax is:

```
<channel definition area> ::=
    <channel symbol>
    is associated with
    { [<channel name>] { [<signal list area>] [<signal list area>] } set }
    is attached to {
    { <agent area> | <state partition area> | <gate on diagram> }
    { <agent area> | <state partition area> | <gate on diagram> } } set
```

Consequently, *change* "connected" to "attached" in the rest of the text of 10.1 (14 times) and in previous 10.2 *Semantics* and *Model* (7 times after other deletions as below).

Also in 10.1 *Concrete grammar*, *change* the word "connection" to "attachment" (twice).

While making these changes it was noticed that the text in 10.2 *Semantics* and *Model* mentioned channels internal to the scope unit to which the <external channel identifiers> are attached.

Therefore, *delete* the following text in 10.2 *Semantics*:

"Each channel identified by a <channel identifier> in an <external channel identifiers> must be defined in the same agent in which the connection is defined and it must have the boundary of that agent as one of its endpoints."

The following text in 10.2 *Semantics* seems to be an unnecessary constraint and it is suggested that it be *deleted*:

"Each channel defined in the surrounding agent and which has its environment as one of its endpoints must be mentioned in exactly one <external channel identifiers>."

The heading *Semantics* in 10.2 is incorrect (the text essentially refers to the grammar) and this heading is *deleted*.

The text in 10.2 *Model* "in their respective scope units" is *deleted* because it is irrelevant – there is only one relevant scope unit (the surrounding one).

At the end of 10.2 *Model*, *add* the clarification sentence:

"The identities of the channels and gates are derived from the attachment (see 10.1)."

1.17 Clarification – 11.2 – Concrete grammar – Incorrect syntax

Delete the rule <connect association area>. This is defined in 11.11.4 replacing <connect area>. See also the change to 11.11.4 in 1.33 of this corrigendum.

1.18 Clarification – 11.2 – Concrete grammar – Use of <composite state name>

The use of <composite state name> is improved.

Change "In this case the <state area> must only contain one <composite state name>" to "In this case the <state area> must only contain one <composite state name>".

1.19 Clarification – 11.2 – Model – Use of <composite state name>

The use of <composite state name> is improved.

Change "one for each <state name> and <composite state name> of the body in question" to "one for each <state name> of the body in question".

1.20 Deficiency correction – 11.3 – Concrete grammar – Incorrect syntax

The syntax for <input area>

```
<input symbol> contains { [<virtuality>] <input list> }  
[ is connected to <on exception association area> ]  
[ is associated with <solid association symbol> is connected to <enabling condition area> ]  
is followed by <transition area>
```

means that when an enabling condition is used, there should be a line (a solid association symbol) joined to the enabling condition AND a flow line joined to the transition from the input symbol. This is clearly incorrect. The *corrected* syntax is:

```
<input symbol> contains { [<virtuality>] <input list> }  
[ is connected to <on exception association area> ]  
{  
    | is connected to <enabling condition association area>  
    | is followed by <transition area> }
```

and the following paragraph is *added* after this syntax:

"The <enabling condition association area> defines the <transition area> in the case of an enabling condition."

There is a corresponding change to 11.6 (see 1.21 below).

1.21 Deficiency correction – 11.6 – Concrete grammar – Incorrect syntax

To correct the syntax for <input area> (see 1.20 above), a new syntax rule is introduced for <enabling condition association area> and the rule <enabling condition area> is extended to include a <transition area>. The *corrected* syntax is:

```
<enabling condition association area> ::=  
    <solid association symbol> is connected to <enabling condition area>  
<enabling condition area> ::=  
    <enabling condition symbol> contains <provided expression>  
    is followed by <transition area>
```

and the following paragraph is *added* after this syntax:

"The <transition area> corresponds to the *Transition* of the *Input-node* or *Spontaneous-transition* for the *Provided-expression*. The syntax appears here to get the correct graphical production of a flow line from the <enabling condition symbol> to the transition."

1.22 Deficiency correction – 11.9 – Concrete grammar – Incorrect syntax

The syntax for <spontaneous transition area> means that when an enabling condition is used, there should be a line (a solid association symbol) joined to the enabling condition AND a flow line joined to the transition from the input symbol. This is clearly incorrect. The *corrected* syntax is:

```
<input symbol> contains { [<virtuality>] <spontaneous designator> }  
[ is connected to <on exception association area> ]  
{  
    | is connected to <enabling condition association area>  
    | is followed by <transition area> }
```

and the following paragraph is *added* after this syntax:

"The <enabling condition association area> defines the <transition area> in the case of an enabling condition."

There is a corresponding change to 11.6 (see 1.21 above).

1.23 Clarification – 11.11 – Use of <composite state name>

The use of <composite state name> is improved.

In the 2nd paragraph, *change* "<composite state name>" *to* "<composite state name>".

1.24 Clarification – 11.11 – Abstract grammar – Procedure with states for entry/exit procedures

The text after the Abstract syntax starting "*Entry-procedure-definition* represents" to before "*Semantics*" is replaced by:

"A procedure with states is a procedure that contains a state (explicit or implicit) or calls a procedure with states.

Entry-procedure-definition of a *Composite-state-graph* or *State-aggregation-node* is a procedure without parameters explicitly defined in the *Composite-state-graph* or *State-aggregation-node* respectively with the name entry. An entry procedure shall not be a procedure with states.

Exit-procedure-definition of a *Composite-state-graph* or *State-aggregation-node* is a procedure without parameters explicitly defined in the *Composite-state-graph* or *State-aggregation-node* respectively with the name exit. An exit procedure shall not be a procedure with states."

1.25 Deficiency correction – 11.11.1 – Concrete grammar – Incorrect syntax

In the rule <composite state graph area>, replace "*associated with*" before { <state connection point>* } set by "*connected to*" because the items are joined (like gates) and the text also describes them as being "connected". At the same time, rename all occurrences of <state connection point> to <state connection point area> to be consistent with the graphical grammar.

1.26 Clarification – 11.11.1 – Concrete grammar – Use of <composite state name>

The use of <composite state name> is improved.

In the rule <composite state heading>, change "<composite state name>" to "<state name>".

1.27 Clarification – 11.11.1 – Concrete grammar – Incorrect syntax

In the rule <composite state body area>, apply {} set to the whole right hand side to make it clearer that the order is not important. At the same time, the layout is made consistent with other similar rules. The resulting syntax lines are:

$$\{ [\text{on exception association area}] \text{<start area>}^* \\ \{ \text{<state area>} \mid \text{<exception handler area>} \mid \text{<in connector area>} \}^* \} \text{ set}$$

1.28 Deficiency correction – 11.11.2 – Concrete grammar – Incorrect syntax

In the rule <state aggregation area>, replace "*associated with*" before { <state connection point>* } set by "*connected to*" because the items are joined (like gates) and the text also describes them as being "connected". At the same time, rename all occurrences of <state connection point> to <state connection point area> to be consistent with the graphical grammar.

1.29 Clarification – 11.11.2 – Concrete grammar – Use of <composite state name>

The use of <composite state name> is improved.

In the rule <state aggregation heading>, change "<composite state name>" to "<state name>".

1.30 Deficiency correction – 11.11.2 – Concrete grammar – Allow gates on inherited state partition

It should be allowed to have gates within <inherited state partition definition>. The revised rule is:

$$\text{<inherited state partition definition> ::=} \\ \text{<dashed state symbol> contains } \{ \text{<composite state identifier>} \{ \text{<gate>}^* \} \text{ set } \}$$

1.31 Deficiency correction – 11.11.2 – Concrete grammar – Incorrect syntax

In addition, [] instead of {} was used by mistake in the rule <state partition connection area>. It should not be allowed for the solid association symbol to be unconnected.

Furthermore, <outer graphical point> was incorrectly shown as associated to a <frame symbol>, whereas the <frame symbol> is a syntax production of the enclosing state aggregation (type) (a <state aggregation area> or <composite state type diagram>). The <outer graphical point> has to be an <outer graphical point> defined by the <state connection point area> of the diagram.

```
<state partition connection area> ::=
    <solid association symbol>
    is attached to <frame symbol>
    is attached to <state partition area>
    is connected to { <outer graphical point> <inner graphical point> }
```

Then, *add* the following paragraph:

"The <solid association symbol> is attached at one end to the <frame symbol> of the enclosing diagram and the <outer graphical point> is placed nearby outside this <frame symbol> of the enclosing diagram. The <solid association symbol> is attached at the other end to a <state partition area> and the <inner graphical point> is placed nearby. The <outer graphical point> shall refer only to names defined as state entry or exit points of the enclosing diagram. The <inner graphical point> shall refer only to names defined as state entry or exit points of the <state partition area>."

Delete the associated graphical parts in the graphical point rules (these are invalid syntax according to the meta-grammar) so that they become:

```
<outer graphical point> ::=
    { <state entry points> | <state exit points> }
<inner graphical point> ::=
    { <state entry points> | <state exit points> }
```

1.32 Deficiency correction – 11.11.3 – Concrete grammar – Incorrect syntax

Delete the last line "*is connected to* <frame symbol>" of the rule <state connection point> because the connection point is connected to the frame and syntactically produces the connection point (rather than the other way around). At the same time, *rename* all occurrences of <state connection point> *to* <state connection point area> to be consistent with the graphical grammar.

1.33 Clarification – 11.11.4 – Concrete grammar – Incorrect syntax

Replace the rule <connect area> *by*:

```
<connect association area> ::=
    <solid association symbol> is associated with { [<virtuality>] [<connect list>] }
    [ is connected to <on exception association area> ]
    is followed by <exit transition area>
```

and *change* all occurrences of "<connect area>" *by* "<connect association area>". See also the change to 11.2 in 1.17 of this corrigendum.

1.34 Clarification – 11.12.2.2 – Concrete grammar – Incorrect syntax

Change "*connected*" *to* "*attached*" in the rule <merge area> to ensure there is another production that produces the flow line.

1.35 Clarification – 11.13.3 – Concrete grammar – Ambiguity

In <procedure call body>, the alternatives <procedure identifier> and <procedure type expression>, were ambiguous because a <procedure identifier> is a <procedure type expression>. The revised syntax is:

```
<procedure call body> ::=  
    [ this ] <procedure type expression> [<actual parameters>]
```

After this syntax the following paragraph is *added* to explain the meaning of <procedure identifier> in the rest of the text of the clause:

"In the following text, <procedure identifier> means the <procedure identifier> for the <base type> of the <procedure type expression>, and if the <procedure type expression> is a <procedure identifier>, it is simply this <procedure identifier>."

1.36 Modification – 11.13.4 – Semantics – Signal routing

To support the ability to direct a signal to an agent instance by reference to its Pid and allow the agent to internally route the signal to the correct contained agent, the interface of an agent should include communication to contained agents. This also requires the change in 1.37 below.

The following paragraph is inserted before the paragraph starting "Note that specifying the same *Channel-identifier*":

"When a signal instance is delivered to an instance of an agent instance set and there is an internal communication path that conveys the signal to the state machine of the agent instance, the signal instance is delivered to that state machine. Otherwise, a communication path within the agent instance able to convey the signal instance is arbitrarily chosen and the signal instance is delivered to an instance set of a contained agent."

1.37 Modification – 12.1.2 – Model of implicit interfaces

To support the ability to direct a signal to an agent instance by reference to its Pid and allow the agent to internally route the signal to the correct contained agent, the interface of an agent should include communication to contained agents. The text defining the implicit interface was hard to understand and at the same time an attempt was made to improve it.

The paragraph:

"The interface defined by an agent or agent type contains in its <interface specialization> all interfaces given in the incoming signal list associated with explicit or implicit gates of the agent or agent type such that the gates are connected via implicit or explicit channels to the gates of the state machine of the agent or agent type. The interface also contains in its <interface use list> all signals, remote variables and remote procedures given in the incoming signal list associated with explicit or implicit gates of the agent or agent type such that the gates are connected via implicit or explicit channels to the gates of the state machine of the agent or agent type. In addition, the interface for an agent type that inherits another agent type also contains in its <interface specialization> the implicit interface defined by the inherited agent type."

is *replaced* by the paragraph:

"Internally connected gates of an agent (or agent type) are explicit or implicit gates of the agent (or agent type respectively) that are connected via implicit or explicit channels to the gates of either the state machine of the agent (or agent type respectively) or a contained agent. The interface defined by an agent or agent type contains in its <interface specialization> all interfaces given in the incoming signal list associated with internally connected gates. The interface contains in its <interface use list> all signals, remote variables and remote procedures given in the incoming signal list associated with internally connected gates. In addition, the interface for an agent type that

inherits another agent type also contains in its <interface specialization> the implicit interface defined by the inherited agent type."

If the text in this paragraph "either the state machine of the agent (or agent type respectively) or a contained agent" is replaced by "the state machine of the agent (or agent type respectively)", it should have the same meaning as before.

The following paragraph, which defines the implicit interface of an agent state machine, then needs to be changed to exclude the items only for contained agents. An attempt has also been made to improve this text – in particular the description of items not relevant to communication via the agents gates. The original text:

"The interface defined by a state machine of an agent or agent type contains in its <interface specialization> the interface defined by the agent or agent type itself. In addition, the interface contains in its <interface specialization> all interfaces given in the incoming signal list associated with explicit or implicit gates of the state machine such that gates are not connected by implicit or explicit channels to explicit or implicit gates of the agent or agent type. The interface also contains in its <interface use list> all signals, remote variables and remote procedures given in the incoming signal list associated with explicit or implicit gates of the state machine such that gates are not connected by implicit or explicit channels to explicit or implicit gates of the agent or agent type. If the containing entity is an agent type that inherits another agent type, then the interface will also contain in its <interface specialization> the implicit interface of the state machine of the inherited agent type."

is replaced by:

"The interface defined by a state machine of an agent or agent type contains in its <interface specialization> the interface defined by the agent or agent type itself except any part of that interface concerned only with contained agents. However, the interface also contains in its <interface specialization> all interfaces given in the incoming signal list associated with any explicit or implicit gates of the state machine. The interface also contains in its <interface use list> all signals, remote variables and remote procedures given in the incoming signal list associated with explicit or implicit gates of the state machine. If the containing entity is an agent type that inherits another agent type, then the interface will also contain in its <interface specialization> the implicit interface of the state machine of the inherited agent type."

This can be altered to give the same meaning as the original text by removing "except any part of that interface concerned only with contained agents".

1.38 Deficiency correction – 12.1.8 – Concrete grammar – Incorrect syntax

In the rule <operation body area>, the <procedure start area> should be optional. "[" and "]" are added before and after <procedure start area>. This allows a refined or abstract definition to omit the <procedure start area>. {} *set* is applied to the whole right hand side to make it clearer that the order is not important. The revised syntax lines are:

```
{ [ <on exception association area> ] [ <procedure start area> ]  
{ <in connector area> | <exception handler area> }* } set
```

1.39 Clarification – 12.3.4.1 – Semantics – Unit of time

The following sentence:

"Unless otherwise specified, the time unit in SDL specifications is 1 second."

is added to the following places:

- in 12.3.4.1 *Semantics* after the text "unit of time are system dependent.";
- in D.3.11.2 Usage for the Duration sort at the end of the paragraph just before D.3.12;

- in D.3.12.2 Usage for the Time sort at the end of the paragraph just before D.3.13.

1.40 Clarification – 13.1 – Concrete grammar – Improved syntax

The rule <option symbol> is *redefined* as:

```
<option symbol> ::=
    {
        <dashed line symbol> is attached to <dashed line symbol>
        <dashed line symbol> is attached to <dashed line symbol>
        <dashed line symbol> is attached to <dashed line symbol>
        { <dashed line symbol> is attached to <dashed line symbol> }+ } set
```

and the word "rectilinear" is *inserted* before the word "polygon" in the sentence following the rule.

1.41 Clarification – D.3.11.2, D.3.12.2 – Unit of time

See 1.39 of this corrigendum.

2 Change to B.9 (Annex B – Backwards compatibility)

The previous edition did not include some of the grammar needed to support operators defined using SDL-92 syntax. The following replaces B.9 in Addendum 1 to Z.100 (10/2003) and includes the previous text.

B.9 Behaviour of operations

Concrete grammar

To be compatible with SDL-92 models, extra syntax is added for operator definitions: <legacy operator definition>, <legacy operator reference> and <legacy external operator definition>. These are used instead of <operation definitions> within a <legacy data type definition> (see B.12).

```
<legacy operator definition> ::=
    {<package use clause>}*
    <operation heading> <end>
    { <entity in operation> }*
    <start>
    endoperator
    [{<operation identifier> | <operation name> }] <end>
<legacy operator reference> ::=
    <operation heading> referenced <end>
<legacy external operator definition> ::=
    operator <operation name> [ <legacy procedure signature> ] external <end>
```

An <operation heading> in a <legacy operator definition> or <legacy operator reference> shall use the keyword **operator**.

A <legacy operator definition> corresponds to an <operation definition> in SDL-2000.

A <legacy operator reference> corresponds to an <operation reference> in SDL-2000.

A <legacy external operator definition> corresponds to an <external operation definition> in SDL-2000.

<start> is defined in ITU-T Rec. Z.106. The body of the transition for <start> shall contain only items that are allowed in an operation definition.

The syntax for <formal operation parameters> is extended to allow the formal parameters to be specified with **fpar**.

```
<formal operation parameters> ::=
    ( <operation parameters> {, <operation parameters> }* )
    | [ <end> ] fpar <operation parameters> {, <operation parameters> }*
```


NOTE – The optional <end> before the keyword **fpar** is added to validate models to be defined using tools that required a semicolon at this point, even though it was not valid in SDL/GR in SDL-92.

The syntax for <operation result> is extended to allow specification with **returns**.

```
<operation result> ::=  
    <result sign> [ <variable name> ] <sort>  
    | returns [ <variable name> ] <sort>
```

3 Additions to Annex B – Backwards compatibility

The grammar defined in Annex B in Addendum 1 to Z.100 (10/2003) did not include some of the grammar needed to support data and data types using SDL-92 syntax. The following is *added* to Annex B to support this SDL-92 syntax.

B.11 Data definition

Concrete grammar

To be compatible with SDL-92 models, the syntax is extended to allow <legacy data type definition> (see B.12) and <legacy syntype definition> (see B.13) in a <data definition>.

```
<data definition> ::=  
    <data type definition>  
    | <legacy data type definition>  
    | <interface definition>  
    | <syntype definition>  
    | <legacy syntype definition>  
    | <synonym definition>
```

To be consistent with SDL-92, a <sort> in any of the components of a <data definition> should always be a sort or syntype identifier.

B.12 Data type definition

Concrete grammar

To be compatible with SDL-92 models, the syntax is extended to allow <legacy syntype definition> (see B.13) in <entity in data type>.

```
<entity in data type> ::=  
    <data type definition>  
    | <legacy data type definition>  
    | <syntype definition>  
    | <legacy syntype definition>  
    | <synonym definition>  
    | <exception definition>  
<legacy data type definition> ::=  
    newtype <sort name>  
        [ <formal context parameters> ]  
        [ <data type specialization> ]  
        | <legacy generators>  
        | <structure definition> ]  
        [ <literal list> ]  
        [ <legacy operator signatures> ]  
        { <legacy operator definition>  
            | <legacy operator reference>  
            | <legacy external operator definition> } *  
        [ <default initialization> [ <end> ] ]  
        [ constants <range condition> ]  
    endnewtype [ <sort name> ]
```

To be consistent with SDL-92, the <data type specialization> in a <legacy data type definition> should contain a <legacy data inheritance> (see B.8).

To be consistent with SDL-92, the <structure definition> in a <legacy data type definition> should not contain <visibility>, **optional** or <field default initialization>.

To be consistent with SDL-92, the <literal list> in a <legacy data type definition> should not contain <visibility> or <named number>.

If a <legacy data type definition> definition contains a <range condition>, it represents the definition of syntype and an anonymous parent data type.

The definition of a legacy operator shall be defined either by the <legacy operator definition> or the operator referenced by a <legacy operator reference> or <legacy external operator definition> (see B.9).

B.12.1 Generators

Concrete grammar

Although SDL-2000 does not include generators for data types, parameterized data types in **package** Predefined of SDL-2000 replace the generators such as Array that were included in the **package** Predefined in SDL-92. The <legacy data type definition> includes the <legacy generators> syntax so that these parameterized data types can be used.

```
<legacy generators> ::=
    <sort identifier> (<legacy generator actual> { , <legacy generator actual> }* )
<legacy generator actual> ::=
    <sort>
    | <literal signature>
    | <operator name>
    | <constant expression>
```

The <sort identifier> should identify one of the parameterized data types in **package** Predefined. The <legacy generator actual> should be an appropriate actual parameter for the parameterized data types.

B.12.2 Operator signatures

Concrete grammar

To be compatible with SDL-92 models, <legacy operator signatures> is allowed as an alternative.

```
<legacy operator signatures> ::=
    operators
    <legacy operator signature> { <end> <legacy operator signature> }* [ <end> ]
<legacy operator signature> ::=
    <operator name> : <arguments> -> <sort>
```

The <legacy operator signature> represents an *Operation-Signature*.

The <sort> of a <legacy operator signature> represents the *Result* of the *Operation-Signature*.

B.13 Syntypes

Concrete grammar

To be compatible with SDL-92 models, <legacy syntype definition> is allowed as an alternative.

```
<legacy syntype definition> ::=  
  syntype  
    <syntype name> = <parent sort identifier>  
    [ <default initialization> [ <end> ] ]  
    [ constants <range condition> ]  
  endsyntype [ <syntype name> ]
```

See also <legacy data type definition> for **syntype** combined with a **newtype** in B.12.

SERIES OF ITU-T RECOMMENDATIONS

Series A	Organization of the work of ITU-T
Series B	Means of expression: definitions, symbols, classification
Series C	General telecommunication statistics
Series D	General tariff principles
Series E	Overall network operation, telephone service, service operation and human factors
Series F	Non-telephone telecommunication services
Series G	Transmission systems and media, digital systems and networks
Series H	Audiovisual and multimedia systems
Series I	Integrated services digital network
Series J	Cable networks and transmission of television, sound programme and other multimedia signals
Series K	Protection against interference
Series L	Construction, installation and protection of cables and other elements of outside plant
Series M	TMN and network maintenance: international transmission systems, telephone circuits, telegraphy, facsimile and leased circuits
Series N	Maintenance: international sound programme and television transmission circuits
Series O	Specifications of measuring equipment
Series P	Telephone transmission quality, telephone installations, local line networks
Series Q	Switching and signalling
Series R	Telegraph transmission
Series S	Telegraph services terminal equipment
Series T	Terminals for telematic services
Series U	Telegraph switching
Series V	Data communication over the telephone network
Series X	Data networks and open system communications
Series Y	Global information infrastructure, Internet protocol aspects and Next Generation Networks
Series Z	Languages and general software aspects for telecommunication systems