International Telecommunication Union

ITU-T Implementer's guide

(September 2019)

TELECOMMUNICATION STANDARDIZATION SECTOR OF ITU

SERIES Z: LANGUAGES AND GENERAL SOFTWARE ASPECTS FOR TELECOMMUNICATION SYSTEMS

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

Specification and Description Language implementer's guide – Version 4.0.0

This page is intentionally blank.

Specification and Description Language implementer's guide – Version 4.0.0

Summary

The purpose of this implementer's guide is to compile reported defects with resolutions and other agreed changes for the ITU-T Specification and Description Language related ITU-T Recommendations (Z.100, Z.101, Z.102, Z.103, Z.104, Z.105, Z.106, Z.107, Z.109, Z.111 and Z.119) prior to these changes being published in approved Recommendations.

This implementer's guide includes all changes agreed by Q12/17 to the texts of the relevant Recommendations consented or in-force at the date the guide is approved by SG17 and applies until either the implementer's guide is updated to following version or all the changes are incorporated into the relevant Recommendations and the implementer's guide is updated to version 3.0.3. It is assumed that consented Recommendations will subsequently be approved.

Source

SDL implementer's guide version 4.0.0 was approved on 05 September 2019 by ITU-T Study Group 17 (2017-2020)

CONTENTS

Page

Specification and Description Language implementer's guide – Version 4.0.0

1 Introduction

This Guide is a compilation of reported defects and maintenance issues with their resolutions for the Specification and Description Language ITU-T Recommendations:

- Z.100, Z.101, Z.102, Z.103, Z.104, Z.105, Z.106, Z.107, Z.109, Z.111 and Z.119.

The Recommendations ITU-T Z.111 and Z.119 are included in the above list because they are essential normative references. Agreed changes to these documents that have not yet been issued in approved Recommendations are therefore listed here.

At the time this Guide was approved, there were no plans for further work on the ITU-T Specification and Description Language, and the SDL-2010 version of the language as consented for approval in September 2019 was (subject to further error detection/correction) expected to be a final version. For that reason, it was decided to increase the version number to 4.0.0 from 3.0.2 in the previous version of this Guide, Z.Imp100 (09/18).

This Guide is intended to be an additional authoritative source of information for implementers to be read in conjunction with the Recommendations themselves.

This Guide itself is not an ITU-T Recommendation. However, it records agreed corrections to reported defects.

This Guide is for the SDL-2010 version of the language as consented for approval in September 2019. The earlier Guide version 1.0.2 was for the SDL-2000 version of the language and therefore did not include Recs. ITU-T Z.101, Z.102 and Z.103, and included the previous Z.107 (withdrawn in 2008). The changes to SDL-2000 versions of the relevant superseded Recommendations can be found in Annex C of Z.Imp100 (04/15) Version 2.0.2 and is not repeated in this version, Z.Imp (09/19).

1.1 Scope of the Guide

The Guide records the resolution of defects and maintenance in the following categories as described in Rec. ITU-T Z.100 Appendix II Guidelines for maintenance of the Specification and Description Language:

- errors
- open items
- deficiencies
- clarifications
- modifications
- decommitted features
- extensions

NOTE: This Guide addresses proposed changes (extensions, deletions, or modifications) to the Recommendations that are strictly related to maintenance of the Specification and Description Language as described in the Z.100 series. Proposals for new features should be made in the normal way through contributions to ITU-T Study Group 17, but if agreed may result in maintenance changes (extensions, deletions, or modifications).

1.2 Approval of the Guide

This Guide is approved by ITU-T Study Group 17.

1.3 Distribution of the Guide

This Guide is available on-line at no charge from the ITU-T at (http://www.itu.int/rec/T-REC-Z.Imp100/en).

1.4 Contact

Any comments should be addressed to the ITU/TSB Secretariat for Study Group 17:

Mrs. Xiaoya Yang
ITU/TSB
Place des Nations
CH-1211 Geneva 20
Switzerland

Tel: +41 22 730 6206
Fax: +41 22 730 4882
E-mail: tsbsg17@itu.int

2 Error reporting procedure

2.1 Submission of error reports and change requests

Any implementer of the Specification and Description Language defined in the ITU-T Z.100, Z.101, Z.102, Z.103, Z.104, Z.105, Z.106, Z.107 and Z.109 Recommendations is invited to submit a report using the form found in Appendix II of Z.100 and copied below in Annex A. The report should be submitted to the ITU-T Study Group 17 Secretariat (see clause 1.4). Each form should cover a single error ("error correction") or proposed change. Where the form reports an error, it is important that the form is completed accurately, especially the sections that relate to the base material against which the error report is being raised.

2.2 Resolution of errors

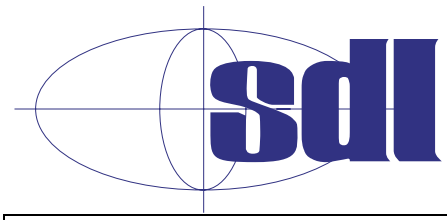
ITU-T Study Group 17 will address the submitted error. Following agreement on a resolution to the error, the proposed resolution will be approved using the appropriate procedures in ITU-T.

Please note that individual responses are not given specifically to those submitting reports, and that the procedure is not intended as a consulting service.

2.3 Documenting the Resolution of Defects and maintenance changes

The ITU-T Recommendations that have errors or agreed changes, are recorded in Annex B with defects and their resolutions including the changes, or with the reasons for maintenance and the changes.

Annex A
Change Request Form



Change Request Form

Please supply the following details.		
Type of change:	❑ error correction	❑ clarification (or question)
	❑ simplification	❑ extension
	❑ modification	❑ decommission

Short summary of change request		
Short justification of the change request		
Is this view shared in your organization?	❑ yes	❑ no
Have you consulted other users?	❑ yes	❑ no
How many users do you represent?	® 1-5 ® 11-100	® 6-10 ® over 100
Your name and address		

Please attach further sheets with details if necessary.

SDL (ITU-T Z.100) Rapporteur, c/o ITU-T, Place des Nations, CH-1211 Geneva 20, Switzerland.
Fax: +41 22 730 5853, e-mail: SDL.rapporteur@itu.int.

Annex B

Master List of Changes

This is the master list of changes for the ITU-T Z.100 (SDL) series Recommendations approved by the Working Party 3 of Study Group 17 in 2019 according the rules for maintenance in Rec. ITU-T Z.100 itself.

History: The previous version of this list was published in version 3.0.2 of this document, which replaced versions 3.0.1, 3.0.0, 2.0.2, 2.0.0, 1.0.2 and 1.0.1 of this document and the earlier document COM 17-TD 3250 [2001-2004] one of the documents of July 2004 WP C/17 meeting. COM 17-TD 3250 [2001-2004] records the history up to that point, and there seems to be no benefit repeating that historical information in this document.

In accordance with Appendix II to Recommendation ITU-T Z.100, the information in this document is distributed to users by various means including sdlnews@sdl-forum.org.

B.1 Objectives and scope

The purpose of this document is to record agreed changes to SDL Recommendations (ITU-T Z.100 to Z.109) and issues that require study and are therefore "open", or have been studied and a decision has been made that the issue is "closed": that is no further study should be undertaken.

The agreed changes come in two categories:

a) Correction of *errors* and *clarifications* (see definitions below and in Rec. ITU-T Z.100, Appendix II);

b) *Extensions* and *modifications* (see definitions below and in Rec. ITU-T Z.100, Appendix II).

The rules for maintenance in an Appendix to Rec. ITU-T Z.100, state that *errors* and *clarifications* published in the Master list of changes "come into effect immediately". Such changes should be published in a Corrigendum, Addendum or revision of the Recommendation as soon as is practical.

Modification and *extensions* imply some change to SDL. The rule in this case is "Unless there are special circumstances requiring such changes to be implemented as soon as possible, such changes will not be recommended until Rec. ITU-T Z.100 is revised."

B.2 Terminology

An *error* is an inconsistency in one or more Recommendations ITU-T Z.100 to Z.109.

A *textual correction* is a change in the text or diagrams of Recommendations that corrects clerical or typographical errors.

An *open item* is an issue identified but not resolved.

A *deficiency* is an issue identified where the semantics of SDL are not clearly defined in the Recommendations.

A *clarification* is a change to the text (or diagrams) in a Recommendation that does not (intentionally) change the meaning of SDL, but is intended to make the Recommendations less ambiguous or easier to understand.

A *modification* changes the semantics of SDL.

An *extension* is a new feature that does not change the semantics of SDL defined in the approved Recommendations for SDL.

B.3 Maintenance of Z.100 to Z.109

A Rec. ITU-T Z.100 Appendix documents the procedure to be followed for the maintenance of Recommendations ITU-T Z.100, Z.101, Z.102, Z.103, Z.104, Z.105, Z.106, Z.107 and Z.109. This procedure requires error corrections, proposed modifications and extensions to be widely publicized and a Master list of changes to be maintained. Clarifications or corrections for errors and deficiencies in the list of changes come into effect "immediately" (that is as soon as the Working Party or Study Group approves the list). Other changes take effect only when the relevant Recommendation is updated.

B.4 Z.100 changes

None.

B.5 Z.101 changes

None.

B.6 Z.102 changes

None.

B.7 Z.103 changes

None.

B.8 Z.104 changes

None.

B.9 Z.105 changes

None.

B.10 Z.106 changes

None.

B.11 Z.107 changes

None.

B.12 Z.109 changes

None.

B.13 Z.111 changes

None.

B.14 Z.119 changes

None.

B.15 List of Open Items

The following is a list of issues classified as open items according to the rules for maintenance for SDL. It was agreed in September 2015 that each item on the list at that time should be progressed and removed from this list once the item has been included in the revised SDL-2010.

B.16 List of Closed items (see B.1 for meaning of a "closed" item)

To facilitate the tracking of items each item uses the identifier of the form (month/year).<number> given when the item was first put onto the open item list. For example "(04/97).3". If the items were never on the open item list, the numbers are consecutive to the open items for the meeting at which the closed item is identified.

B.16.1 allow algorithmic operators with external data

This was listed as an item in COM-10-1 in 1997 but the requirement is not clear.

B.16.2 more flexible USE syntax

Originally listed as item (04/97).25 of Q.6/10 (SDL) Meeting, 29 April – 05 May 1997.

The proposal was to **`use`** `p1, p2, p3;` instead of **`use`** `p1;` **`use`** `p2;` **`use`** `p3;` to use packages `p1`, `p2` and `p3`. A <package use clause> has an optional <definition selection list> after the package name and <definition selection list> is a comma separated list of names, so that allowing list of comma separated package names would lead to syntactic ambiguity. Therefore the proposed change has no benefit.

B.16.3 operators returning sets of values (multivalued operators)

This issue was originally listed as item (10/96).13 of COM-10-1 1997.

This is adequately be handled by **`struct`**.

B.16.4 signal priority

Originally listed as item (04/97).18 in the Q.6/10 (SDL) Meeting, 29 April – 05 May 1997 report.

The concept is to give signals a (possibly dynamic) priority. The selection signals for consumption is already relatively complex and a fundamental part of SDL-2010.

In SDL-2010 it is possible to state the availability time for signals determined by the sending agent. Signals are not available to be consumed from the input port of the recipient until the specified time has been reached. Two or more signals that become available at the same time are ordered according to a priority specified by the senders.

In SDL-2010 which available signal instance is selected for consumption depends on the current state, the priority of the input for the signal in that state and the order of the signals in the input port.

B.16.5 **`virtual`** as default

Originally listed as item (04/97).31 in the Q.6/10 (SDL) Meeting, 29 April – 05 May 1997 report.

In SDL-2010 (and SDL-2000 and SDL-92) a type in which <virtuality> is omitted is treated as **`finalized`**. This had been extensively discussed (and **`virtual`** as default rejected) when SDL-92 was formulated, and has implications on the use of constraints.

B.16.6 remote process creation

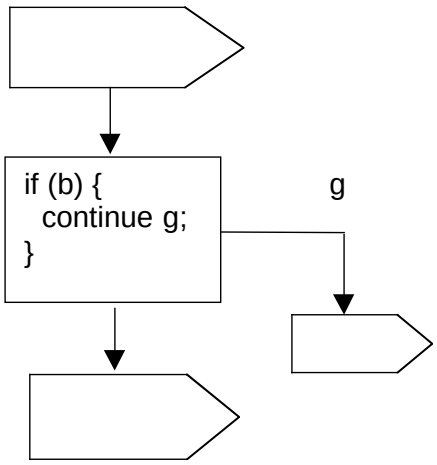
Originally listed as item (04/97).31 in the Q.6/10 meeting 21-25 September 1998 report.

The need is satisfied by remote procedure in the state machine of an agent, and it was decided an additional construct made the language too complex.

B.16.7 exit connection points for tasks

Listed as an issue (06/00).1 in the Report of Q.6/10 meeting Oslo 8-9 June 2000, 6.6.

The issue is whether to allow a statement in a textual algorithm in a task symbol to transfer control to an additional exit point from the task symbol. The following diagram gives an outline example where g is an exit connection point.



This issue is considered closed. There should only be one exit point from a task symbol.

B.16.8 Issues that are closed because no proposals were received over several years

These items had been listed as additional open items in Z.Imp100 for a considerable time, but no proposals were received. It was decided in September 2015 to close these items.

- Data type library extensions (predefined object types, Standard Template Library analogue)
- Memory management issues
- Instance sets vs. container types and navigation into composite agents
- Broadcast mechanisms
- Interrupts

Annex C

Master List of Changes to SDL-2010 (2016-04)

The changes listed below if applied the SDL-2010 Recommendations published in 2016-04, should produce the text of the corresponding SDL-2010 Recommendations consented for approval in 2019.

C.1 Z.100 changes

C.1.1 *Textual correction* – Heading 7.2, Language

The language of Heading 7.2 is corrected to "English UK" instead of "French".

C.1.2 *Modification* – Annex A, Abstract syntax index

The rule State-identifier is added to the table with "defined" in the Z.101 column.

The rule Channel-endpoint is added to the table with "defined" in the Z.101 column.

C.1.3 *Modification* – Annex B, BNF syntax index

The rule <actual context parameters> is deleted from the table.

The rule <field of kind> is added to the table with "defined" in the Z.101 column.

The rule <nextstate body name> is added to the table with "defined" in the Z.101 column.

C.1.4 *Textual correction* –D.2.3 String

In paragraph 1, correct the round brackets after "String" to "< … >" for actual context parameters, Change

"`String(Integer)` is a string of integers. In particular, the `Charstring` is defined as `String(Character)`."

to

"`String < Integer >` is a string of integers. In particular, the `Charstring` sort is defined as `String < Character >`."

C.1.5 *Modification* –D.2.7 Real

The text is revised, so that `Real` literals are always distinct from `Integer` literals. Replace the paragraph:

The notation for a `Real` value is a sequence of one or more of the numbers 0 to 9 (the same as an `Integer`) or a sequence of one or more of the numbers 0 to 9 followed by a decimal point (represented by a full stop) followed by one or more of the numbers 0 to 9. Examples are: 42; 999; 10.3; 0.79 and .001.

by the paragraph:

The notation for a `Real` value is a sequence of one or more of the numbers 0 to 9 followed by a decimal point (represented by a full stop) followed by one or more of the numbers 0 to 9. Examples are: 42.0; 10.3; 0.79 and .001..

C.1.6 *Clarification* – Appendix III.4, Differences between SDL 2000 and SDL 2010

In the sixth paragraph, change "further study is in progress to provide these benefits in another way" to "further study was progressed to provide these benefits as defined in [ITU-T Z.107]", and delete the text in in the paragraph after this sentence.

After the paragraph ending "timer or is specifically named." insert a new paragraph

> The SDL-2010 construct <agent instance pid value> enables initialization of values to denote the `Pid` values of the agent instances that exist when the system is initialized. Without such language defined initialization either the system has to be designed so that the `Pid` values of the agent instances are dynamically communicated between agents before normal handling of external signals commences, or some tool initialization of the values has to be used.

and a new paragraph

> In SDL-2010, all timers of an agent are reset by the construct **`reset`** `*`, and all timers of an agent with the given timer name `timername` are reset by the construct **`reset`** `timername *`.

At the start the paragraph starting "In SDL-2010, when a signal is placed insert the sentence:

> The signals that are available for input are placed in the input port of the agent instance to receive the signal.

After the modified paragraph above insert a new paragraph

> In SDL-2010 the values of unconsumed signals available in the input port can be examined through the **`signallist`** variable. The **`signallist`** variable has a string sort that is denoted **`as signallist`**, so that **`signallist`**`[n]` (where `n` is an integer) provides a choice value for an available signal. The string is ordered so that **`signallist`**`[1]` is the first available signal and the remaining string elements are in the availability order of the corresponding signals. The name of signal n is given by **`signallist`**`[n]!Present`, and is used to access the values conveyed by the signal in the choice value **`signallist`**`[n]`.

In the paragraph starting "In SDL-2010 a **`signallist`** definition", correct the text "meaning defining" to "meaning as defining".

After the paragraph

> In SDL-2010, it is possible to specify the delay between output of signal and the signal being available for consumption in the destination input port.

insert the new paragraph

> In SDL-2010, there are alternatives to specifying the name of signal to use in an output: an expression can be given where the sort of the expression is a choice sort that corresponds to a choice of signals that can be output; or if encoding rules are specified for a communication path and the output is directed via that path, an expression can be given that corresponds to the data type (`Charstring`, `Octetstring` or `Bitstring`) for that encoding. When a signal is input as an alternative to assigning each of the signal parameters to variables, the signal can be assigned to a variable with choice data type that corresponds to a choice of signals that can be input. The **`signal`** keyword denotes a variable that can hold any of the signals that can be received. The **signal** variable can be used in an input, the choice value of the **`signal`** variable can be accessed in expressions, and the **`signal`** variable can be used in an output to send a signal instance.

and after this the new paragraph

SDL-2010 allows an optional natural expression after the <agent identifier> of a <destination> to select a specific agent instance when there is more than one instance of the identified agent set, otherwise any existing instance of the set of agent instances is selected.

C.2 Z.101 changes

C.2.1 *Textual correction* – Language

Change all parts of the main body (from page 1) text marked as "French" or "Swiss French" or "English (US)" or "Spanish" or "German (Switzerland)" to "English (UK)".

C.2.2 *Textual correction* – Non-breaking hyphens in abstract syntax names

Abstract syntax rule names are always in italic, start with an uppercase letter followed by lowercase letters and hyphens and ending in a lowercase letter. To make it easy to find the abstract syntax rule names in the MSWord macros used to maintain SDL-2010, these should not be non-breaking hyphens. Change any non-breaking hyphen in abstract syntax names to normal (line breaking) hyphens. If this causes the line to break on a hyphen in an abstract syntax name, a line break (^l) is inserted immediately before the abstract syntax name.

C.2.3 *Clarification* – 6.6 Names and identifiers, name resolution and visibility, *Abstract grammar*

The *Identifier* for system or for any package not contained in another package should have an empty *Path-item* list, therefore in the rule *Qualifier* change "*Path-item* +" to "*Path-item**" so that the list can be empty. At the end of the paragraph after the rule *Name* insert the sentence "For the system and any package not contained in another package the full path is an empty *Path-item* list."

C.2.4 *Clarification* – 6.6 Names and identifiers, name resolution and visibility, *Concrete grammar, NOTE 2*

Delete the sentence: "Therefore, it is not possible to refer to the identifiers introduced in a definition attached to these scope units by qualifiers."

C.2.5 *Clarification* – 6.10 Frame symbol and page numbers, *Concrete grammar*

In the first text paragraph correct <block type diagram> to <block type page> and correct <process type diagram> to <process type page>.

C.2.6 *Clarification* – 7.1 Framework

The paragraph starting "The *Package-definition-**set*** …" is moved from *Concrete grammar* to become a new *Model* sub-clause.

C.2.7 *Clarification* – 7.2 Package, *Model*

Add a *Model* containing the paragraph:

> If a package is mentioned in several <package use clause> items of a definition (in the same text area or different text areas of the definition), these are replaced by one <package use clause> in one text area that selects the union of the definitions selected in the <package use clause> items.

C.2.8 *Clarification* – 7.3 Referenced definition, *Concrete grammar*

In the first text paragraph after the syntax for <diagram> after "For each <referenced definition>" insert "except any outermost <package diagram>" and in the next paragraph delete "except any outermost <package diagram>".

In the second paragraph after the syntax for <diagram>, delete
"except any outermost <package diagram>"
change
"If two <referenced definition>s" to "If two <referenced definition> items",
change
"The <qualifier> shall" to "The <qualifier> in a <referenced definition> shall",
and change
"context, otherwise the <package diagram>" to "context, except if the <package diagram>".

Add a new paragraph at the end of the clause:

It is not allowed to specify a <qualifier> after the initial keyword(s) for definitions which are not <referenced definition> items.

C.2.9 *Clarification* – 8.1.1.1 Agent types, *Concrete grammar*

In the syntax rule for <agent type diagram> delete the line
[***is associated with*** <package use area>]

Delete the paragraph starting "The <package use area>" between the syntax rule for <agent type diagram> and the syntax rule for <type preamble>. The paragraph is not needed every time <package use area> occurs in a syntax rule, because it is described in the first text paragraph of *Concrete grammar* in **6**.10 Frame symbol and page numbers:

> - the <package use clause> is associated with the frame symbol for the page of the diagram (not the diagram – which in any case is not a symbol).

C.2.10 *Modification* – 8.1.1.1 Agent types, *Model*

Add a *Model* clause:

Model

An <agent formal parameters> list item with a <parameters of sort> that defines multiple parameter names is replaced by a sequence of <agent formal parameters> list items with the same <aggregation kind> each <parameters of sort> defining one name.

C.2.11 *Correction* – 8.1.1.2 System type, *Concrete grammar*

At the end of the syntax for <system type page> add the line
[***is associated with*** <package use area>]

At the end of the clause add the paragraph:

Each <gate on diagram> of the <block type page> represents a *Gate-definition-**set*** item of the *Agent-type-definition*.

C.2.12 *Correction* – 8.1.1.3 Block type, *Concrete grammar*

At the end of the syntax for <block type page> add the line
[***is associated with*** <package use area>]

At the end of the clause add the paragraph:

Each <gate on diagram> of the <system type page> represents a *Gate-definition-**set*** item of the *Agent-type-definition*.

C.2.13 *Correction* – 8.1.1.4 Process type, *Concrete grammar*

At the end of the syntax for <process type page> add the line

[***is associated with*** <package use area>]

At the end of the clause add the paragraph:

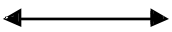
Each <gate on diagram> of the <process type page> represents a *Gate-definition-**set*** item of the *Agent-type-definition*.

C.2.14 *Correction* – 8.1.1.5 Composite state type, *Abstract grammar*

NOTE 1 is incorrect. Instead there is a condition that the gate definition set is non-empty when the composite state type is used for a state machine. Renumber NOTE 2/NOTE 3 as NOTE 1/NOTE 2, replace the old NOTE 1 by:

The *Gate-definition-**set*** of a *Composite-state-type-definition* shall not be empty if there is a *State-machine* based on the *Composite-state-type-definition*.

C.2.15 *Textual correction* – 8.1.4 Gate, *Concrete grammar*, <gate symbol 2>

Change the symbol  to the symbol  . The horizontal line was missing between the arrows.

C.2.16 *Modification* – 9.4 Procedure, *Model*

Add a *Model* clause:

Model

A <formal variable parameters> with a <parameters of sort> that defines multiple parameter names is replaced by a sequence of <formal variable parameters> with the same <parameter kind> and <aggregation kind>, and each <parameters of sort> defining one name.

C.2.17 *Correction* – 10.1 Channel, *Abstract grammar*

Add *Channel-endpoint* to the syntax for *Channel-path* as follows:

Channel-path :: *Channel-endpoint*
Originating-gate
Channel-endpoint
Destination-gate
*Signal-identifier-**set***

At the end of the syntax rules add *Channel-endpoint* and *State-identifier* as follows:

Channel-endpoint = *Agent-identifier*
State-identifier
ENV

State-identifier = *Identifier*

In the fourth paragraph after the syntax, make a clarification by changing
"the same scope unit in the abstract syntax"
to
"the same scope unit (which includes directly enclosed scopes) in the abstract syntax".

C.2.18 *Clarification* – 10.1 Channel, *Concrete grammar*

After the paragraph starting " For a <channel symbol 1>", insert the paragraph:

If the arrowhead of a <channel symbol 1> points away from an attached <agent area>, the first *Channel-endpoint* is the *Agent-identifier* for that agent. If the arrowhead of a <channel symbol 1> points away from an attached <state machine area>, the first *Channel-endpoint* is the *State-identifier* for the state machine. Otherwise, if the <channel symbol 1> points away from an attached <gate on diagram> and the first *Channel-endpoint* is **ENV**.

At the end of the next paragraph add the sentence:

If the arrowhead of a <channel symbol 1> points away from an attached <gate on diagram>, this gate represents the *Originating-gate*.

After the above sentence add a new paragraph:

If the arrowhead of a <channel symbol 1> points to an attached <agent area>, the second *Channel-endpoint* is the *Agent-identifier* for that agent. If the arrowhead of a <channel symbol 1> points to an attached <state machine area>, the second *Channel-endpoint* is the *State-identifier* for the state machine. Otherwise, if the <channel symbol 1> points to an attached <gate on diagram> and the second *Channel-endpoint* is **ENV**.

At the end of the next paragraph (containing "the <gate> in this area") add the sentence:

If the arrowhead of a <channel symbol 1> points to an attached <gate on diagram>, this gate represents the *Destination-gate*.

Change the next paragraph (just before the syntax for <channel symbol 1>) to:

For a <channel symbol 2> there are two *Channel-path* items: one arrowhead corresponds to one *Channel-path* and the other arrowhead to the other *Channel-path*.

In the second paragraph after the syntax for <delaying channel symbol 2>, after "(compared with any other <signal list area>)" insert " and represents the *Signal-identifier*-**set** for the corresponding *Channel-path*", and add the following sentence to the end of the paragraph:

The *Channel-endpoint* items, *Originating-gate* and *Destination-gate* for this *Channel-path* are determined in the same way as for the arrowhead on a <channel symbol 1>.

In the next paragraph (the third paragraph after the syntax for <delaying channel symbol 2>), after the text "channel has no delay" add " and represents **NODELAY** in the *Channel-definition*".

C.2.19 *Textual correction* – 10.1 Channel, *Semantics*, fifth paragraph

The hyphens in "First-In-First-Out" should be consistent (all normal or all non breaking). Change to normal (line break) hyphens (a line break should be inserted before if needed).

NOTE "First-In-First-Out" is not an abstract syntax name.

C.2.20 *Clarification* – 10.3 Signal, *Concrete grammar*

At the end of the *Concrete grammar* insert the paragraph:

If several <signal definition> items are specified in one <signal definition list>, this is equivalent to individual <signal definition list>s for each of them.

C.2.21 *Modification* – 11.3 Input, *Concrete grammar*

At the end of the *Concrete grammar* insert the paragraph:

A <variable> of a <stimulus> shall not be a global variable of a system (type) or block (type) except if the <stimulus> is within the state machine actions of system (type) or block (type).

C.2.22 *Clarification* – 11.12.2.1 Nextstate, *Concrete grammar*

Replace the rule <nextstate body> (to be compatible with Z.100 Annex F2) by

```
<nextstate body> ::=
                        <nextstate body name>
                    |   <dash nextstate>
                    |   <history dash nextstate>

<nextstate body name> ::=
                        <basic state name>
                    |   <composite state name> <nextstate parameters>
```

C.2.23 *Modification* – 11.13.3 Procedure call, *Semantics*

Delete the first paragraph of *Semantics* starting "If the *Procedure-definition* denoted ", which instead should be a *Model* referring to the <procedure definition>.

C.2.24 *Clarification* – 11.13.3 Procedure call, *Semantics*

In the third paragraph delete "or *Inout-parameter*", because the actual parameter has to have the same *Sort-reference-identifier* as the *Procedure-formal-parameter* (see *Abstract grammar)* and therefore the value cannot be out of range for the formal parameter.

C.2.25 *Modification* – 11.13.3 Procedure call, *Model*

Add a new *Model* clause containing the following text paragraph and Note:

If the procedure identified by the <procedure type expression> of the <procedure call body> is not defined within the agent type enclosing the call, within the enclosing agent type there is an implicitly defined local procedure with the same name as identified by the <procedure type expression> and the call uses this local procedure. In the local procedure, identifiers of items (such as variables) external to the procedure definition are bound in the context of the original procedure definition rather than the context of the procedure call if that is different.

NOTE – An implicitly defined local procedure is an inherited subtype of the procedure identified by the <procedure type expression> of the <procedure call body> (see clause 8.4 Specialization of [ITU-T Z.102], and clause 9.4 Procedure of [ITU-T Z.102]).

C.2.26 *Clarification* – 11.13.4 Output, *Abstract grammar*

The syntax for *Signal-destination* is changed to:

Signal-destination :: { *Expression* | *Agent-identifier* | **THIS** } [*Destination-number*]

A sentence is added to the end of the paragraph starting "The sort of *Expression* of " about *Destination number*:

The *Destination-number* is always omitted for a *Signal-destination* that is an *Expression.*

At the end of the seventh text paragraph, remove italic from the full stop after "is an *Expression*".

C.2.27 *Textual correction* – 11.13.5 Decision, *Abstract grammar*

Change the "::" to "=" in syntax for *Decision-node* (as in Z.102):

Decision-node = *Decision-body*

In the text paragraph after the syntax, change "shall be compatible the" to "shall be sort compatible with the".

C.2.28 *Textual correction* – 12.1.2 Interface definition, *Concrete grammar*

In the first paragraph change "*Agent-definition*" to "*Agent-type-definition*" twice (because an *Agent-type-definition* does not contain a *Data-type-definition-**set***).

In the second paragraph correct "each agent type definitions" to "each agent type definition".

In the list for items in the *Signal-identifier-**set*** for <interface use list> use, in item (c) change "a corresponding *Signal-identifier* in the *Signal-identifier-**set*** for" to "corresponding *Signal-identifier* items in the *Signal-identifier-**set***: one for".

After the syntax for <interface use list> insert the following two text paragraphs:

Each <signal list item> of the <signal list> in an <interface use list> of an <interface definition> shall be a <signal identifier> or an <interface identifier>. An <interface<identifier> that is part of the <signal list> shall also respect the restriction.

The <interface definition> shall not contain the <interface identifier> defined by the <interface definition> either directly or indirectly (via another <interface identifier>).

C.2.29 *Textual correction* – 12.1.2 Interface definition, *Semantics*

Change the start of the paragraph starting "The *Signal-identifier-**set*** of a *Interface-definition* is the set of signals" to "The *Signal-definition-**set*** of an *Interface-definition* is the set of signals". In the next paragraph change " appears in the concrete syntax" to " appears in the syntax".

C.2.30 *Textual correction* – 12.1.3 Operation signature, *Abstract grammar*

Change the rule

Static-operation-signature = *Operation-signature*

to

Static-operation-signature :: *Operation-signature*

In the first paragraph after the abstract grammar change "Procedure-*identifier*" to all italics: "*Procedure-identifier*".

C.2.31 *Clarification* – 12.1.3 Operation signature, *Concrete grammar*

Move the last paragraph (starting "An <operation signature> of") before the *Semantics* heading to immediately after the syntax rule <result>.

C.2.32 *Textual correction* – 12.1.6.1 Literals constructor, *Concrete grammar*

So that literal values are numbered from zero, the last two lines before *Semantics* are replaced by:

```
literals B, A = 2, C, D;
```

has B < C , C < A , A < D , num(C) = 1, num(D) = 3

C.2.33 *Textual correction* – 12.1.6.1 Literals constructor, *Semantics*, sixth paragraph

The hyphens in "less-or-equal-than" should be consistent (all normal or all non breaking). Change to normal (line break) hyphens (a line break is inserted before).

C.2.34 *Textual correction* – 12.1.6.2 Structure data types, *Concrete grammar*

To match F2, in the syntax for <fields of sort> change <aggregation kind> <field name> to <field of kind>, and add new rule:

<field of kind> ::=
<aggregation kind> <field name>

The enumerated list for the implicit operations of a structure, described *Operation-signature* parameters with an <aggregation kind>, but aggregation is not part of an operation signature. This has been corrected to describe the *Parameter-aggregation* of the parameter of the same name in the procedure identified by the *Operation-signature*. The changes are:

In item (c) replace:

> `S` is an **in/out** parameter with an empty <aggregation kind>,
> and the result has the same <aggregation kind> as the field `fn`.

by

> and in the procedure identified by the *Operation-signature*
> `S` is an **in/out** parameter with **PART** *Parameter-aggregation* for `S`
> and the *Result-aggregation* is derived from the <aggregation kind> of field `fn`.

In item (d) replace:

> `S` is an **in/out** parameter with an empty <aggregation kind>,
> `fs` is an **in** parameter with the same <aggregation kind> as the field `fn`,
> and the procedure identified by the *Operation-signature* has a *Result-aggregation* that is **PART**.

by

> and in the procedure identified by the *Operation-signature*
> `S` is an **in/out** parameter with **PART** *Parameter-aggregation*,
> `fs` is an **in** parameter with *Parameter-aggregation* derived from the <aggregation kind> of field `fn`,
> and the *Result-aggregation* is **PART**.

In item (e) replace:

> is an **in/out** parameter with an empty <aggregation kind>,
> and the procedure identified by the *Operation-signature* has a *Result-aggregation* that is **PART**.

by

> and in the procedure identified by the *Operation-signature*
> `S` is an **in/out** parameter with **PART** *Parameter-aggregation*,
> and the *Result-aggregation* is **PART**.

In item (f) replace:

> `S` is an **in/out** parameter with an empty <aggregation kind>,
> and the procedure identified by the *Operation-signature* has a *Result-aggregation* that is **PART**.

by

> and in the procedure identified by the *Operation-signature*
> `S` is an **in/out** parameter with **PART** *Parameter-aggregation*,
> and the *Result-aggregation* is **PART**.

C.2.35 *Textual correction* – 12.1.6.3 Choice data types, *Concrete grammar*

The enumerated list for the implicit operations of a choice, described *Operation-signature* parameters with an <aggregation kind>, but aggregation is not part of an operation signature. This has been corrected to describe the *Parameter-aggregation* of the parameter of the same name in the procedure identified by the *Operation-signature*. The changes are:

In item (b) replace:

> `fs` is an **in** parameter with the same <aggregation kind> as the field `fn`,
> and the procedure identified by the *Operation-signature* has a *Result-aggregation* that is **PART**.

by

> and in the procedure identified by the *Operation-signature*
> `fs` is an **in** parameter with *Parameter-aggregation* derived from the <aggregation kind> of field `fn`,
> and *Result-aggregation* is **PART**.

In item (c) replace:

> `c` is an **in/out** parameter with an empty <aggregation kind>,
> and the result has the same <aggregation kind> as the field `fn`.

by

> and in the procedure identified by the *Operation-signature*
> `c` is an **in/out** parameter with **PART** *Parameter-aggregation*,
> and *Result-aggregation* is derived from the <aggregation kind> of field `fn`.

In item (d) replace:

> `c` is an **in/out** parameter with an empty <aggregation kind>,
> `fs` is an **in** parameter with an empty <aggregation kind>,
> and the procedure identified by the *Operation-signature* has a *Result-aggregation* that is **PART**.

by

> and in the procedure identified by the *Operation-signature*
> `c` is an **in/out** parameter with **PART** *Parameter-aggregation*,
> `fs` is an **in** parameter with *Parameter-aggregation* derived from the <aggregation kind> of field `fn` ,
> and *Result-aggregation* is **PART**.

In item (e) replace:

> `c` is an **in/out** parameter with an empty <aggregation kind>,
> and the procedure identified by the *Operation-signature* has a *Result-aggregation* that is **PART**.

by

> and in the procedure identified by the *Operation-signature*
> `c` is an **in/out** parameter with **PART** *Parameter-aggregation*,
> and *Result-aggregation* is **PART**.

In item (f) replace:

> `c` is an **in/out** parameter with an empty <aggregation kind>,and the procedure identified by the *Operation-signature* has a *Result-aggregation* that is **PART**.

by

and in the procedure identified by the *Operation-signature*
`C` is an **in/out** parameter with **PART** *Parameter-aggregation*,
and *Result-aggregation* is **PART**.

In item (g) replace:

where `C` is an **in/out** parameter with an empty <aggregation kind>,
and the procedure identified by the *Operation-signature* has a *Result-aggregation* that is **PART**.

by

and in the procedure identified by the *Operation-signature*
where `C` is an **in/out** parameter with **PART** *Parameter-aggregation*,
and *Result-aggregation* is **PART**.

C.2.36 *Modification* – 12.1.6.3 Choice data types, *Concrete grammar*

In the last text paragraph of the *Concrete grammar*, change "in the context that the *Data-type-definition* for `C` occurs" to "in the *Data-type-definition* for `C`, therefore both `AnonPresent` and its contained *Literal-signature-**set*** are visible where `C` is visible" to make this consistent with visibility rules in 6.6.

C.2.37 *Textual correction* – 12.1.6.3 Choice data types, *Semantics*, ninth paragraph (ignoring NOTE paragraphs)

Change the hyphens in "*field-present-name*" to normal hyphens and insert a line break before "*field-present-name*".

C.2.38 *Textual correction* – 12.1.7 Behaviour of operations, *Concrete grammar*

In the fifth text paragraph (ignoring NOTEs and syntax) change "shall not contain" to "shall contain".

C.2.39 *Textual correction* – 12.1.8.2 Constraint, *Concrete grammar*

Change " if `Length` has been defined" to " if `length` has been defined". The name `length` is all lowercase.

C.2.40 *Clarification* – 12.1.8.2 Constraint, *Concrete grammar*

In the list item c) sub item 3):

Change " with a formal parameter `A P` " to " with a formal parameter `A` of sort `P` ", and change sub sub items:

i) `length(A) = RC`, for each <open range> `RC` in `RC` of the form `constant`;

ii) `length(A) RC`, for each <open range> `RC` in `RC` of the form `= constant`, `/= constant`, `< constant`, `<= constant`, `> constant`, and `>= constant`;

to

i) `length(A) = OP`, for each <open range> `OP` in `RC` of the form `constant`;

ii) `length(A) OP`, for each <open range> `OP` in `RC` of the form `= constant`, `/= constant`, `< constant`, `<= constant`, `> constant`, and `>= constant`;

C.2.41 *Clarification* – 12.2.2 Literal, *Concrete grammar*

In the second text paragraph after the syntax rule <literal identifier>, change "defined *Literal-identifier*" to "sort": that is, there must be exactly one binding that satisfies resolution by context.

In the third text paragraph change "contains" to "ends with".

C.2.42 *Clarification* – 12.2.7 Range check expression, *Abstract grammar*

In the first text paragraph change " *Expression* shall be sort compatible" to "*Expression* of a *Range-check-expression* shall be sort compatible".

C.2.43 *Clarification* – 12.2.7 Range check expression, *Concrete grammar*

In the first text paragraph change " <sort identifier> shall " to "<sort identifier> of a <range check constrained sort> shall ".

Change " <identifier> <constraint sort> " to "<range check constrained sort>".

Change "<sort identifier> identifies a syntype" to "<sort identifier> in <range check constrained sort> identifies a syntype".

C.2.44 *Textual correction* – 12.3.1 Variable definition, *Abstract grammar*, NOTE 1

Change "Aggregation-kind" to italic: "*Aggregation-kind*" (twice in NOTE).

The NOTE seems to be followed by a blank paragraph. Delete it.

C.2.45 *Textual correction* – 12.3.2 Variable access, *Abstract grammar*

Change the "=" to "::" in syntax for *Variable-access*.

C.2.46 *Textual correction* – 12.3.3.1 Extended variable, *Model*

Change the format of the line

<variable> **(** <actual parameter list> **)** <is assigned sign>

to "z.100 text" (rather than "z.100 syntax").

Change the format of the text

<variable> <left square bracket> <actual parameter list> <right square bracket> <is assigned sign> <expression>

to "z.100 text" (rather than "z.100 syntax").

The model for <field variable> assignment should assign the result of the *field-modify-name* operator to the <variable> (similar to the model for <indexed variable> assignment). Insert "<variable> <is assigned sign> " before "*field-modify-name* **(** <variable>, <expression> **)**".

C.2.47 *Clarification* – 12.3.3.2 Default initialization, *Concrete grammar*

Delete the paragraph "A <data type definition> or <syntype definition> shall contain at most one <default initialization>."

C.2.48 *Textual correction* – 12.3.4.2 Pid expression, *Concrete grammar*, <pid expression>

Insert a ">" after "<offspring expression" in the syntax rule <pid expression>.

C.2.49 *Textual correction* – 12.3.4.3 Timer active expression and timer remaining duration, *Semantics*

In the 4th paragraph, 2nd sentence after "result value" insert " for an active timer".

In the 4th paragraph, 3rd sentence after "if the time" insert " is active but".

Change the last sentence of the 4th paragraph to " If the timer is inactive, the value is zero (which can be distinguished from an active time returning zero by a subsequent timer active expression)."

C.3 Z.102 changes

C.3.1 *Textual correction* – Language

Change all parts of the main body (from page 1) text marked as "French" or "Swiss French" or "English (US)" or "Spanish" or "German (Switzerland)" to "English (UK)".

C.3.2 *Clarification* – 6.6 Visibility rules, names and identifiers – additional scope units, Note 1

Delete NOTE 1 and change numbering of "NOTE 2" to "NOTE".

C.3.3 *Extension* – 6.6 Visibility rules, names and identifiers – additional scope units

After the paragraph "A formal context parameter is an entity of the same entity kind as the corresponding actual context parameters." insert the following headings, rules and paragraphs for the Abstract and concrete grammar:

Abstract grammar

Path-item	=	*Package-qualifier*
	|	*Agent-type-qualifier*
	|	*Agent-qualifier*
	|	*State-type-qualifier*
	|	*State-qualifier*
	|	*Data-type-qualifier*
	|	*Procedure-qualifier*
	|	*Interface-qualifier*
	|	*Compound-node-qualifier*
Compound-node-qualifier	::	*Interface-name*
Compound-node-name	=	*Name*

The abstract syntax is extended from Basic SDL-2010 to include *Compound-node-qualifier* in *Path-item* to identify the scope of a compound statement.

Concrete grammar

<scope unit kind> ::=

package
| **system type**
| **system**
| **block**
| **block type**
| **process**
| **process type**
| **state**
| **state type**
| **procedure**
| **signal**
| **type**
| **operator**
| **method**
| **interface**
| **composition**

The syntax of <scope unit kind> is extended from Basic SDL-2010 to include **composition** in Path-item to identify the scope of a compound statement. If the <scope unit kind> of a <qualifier> is **composition**, the <qualifier> a represents a *Compound-node-qualifier* and <name> of the <qualifier> is the <connector name> of a <compound statement>. If no <connector name> is given for the <compound statement>, a newly created anonymous name represents the *Connector-name*; therefore no explicit <qualifier> can be given. In this case the specification is only valid if all uses of a variable name defined in the compound statement are uniquely bound without a qualifier.

C.3.4 *Clarification* – 8.1.2 Type expression, *Concrete grammar*

In the rule <type expression> change "<actual context parameters>" to "<actual context parameter list>". In the next paragraph (after the syntax rule) change "<actual context parameters>" to "<actual context parameter list>" (twice).

C.3.5 *Clarification* – 8.1.2 Type expression, *Model*

The list of how the "anonymous type definition is formed" is modified (to match the formal definition) as follows:

In item (2) change "<actual context parameter> in" to "<actual context parameter> (if there is one) in";

In item (3) change "<formal context parameter list> and" to "<formal context parameter list> if there is a corresponding <actual context parameter>, and ";

In item (4) change "by the anonymous unique name." to "by a <type expression> with a <base type> that identifies the type with the anonymous unique name and no <actual context parameter list>.".

In NOTE 1 following item(4) change "<actual context parameters>" to "<actual context parameter list>".

C.3.6 *Clarification* – 8.3 Context parameters, *Concrete grammar*

Delete the syntax rule <actual context parameters>; change the rule <actual context parameter list> to:

<actual context parameter list> ::=
<context parameters start>
[<actual context parameter>] {**,** [<actual context parameter>] }*
<context parameters end>

In the last line before *Model* change "in <actual context parameters>" to "in the <actual context parameter list>".

C.3.7 *Correction* – 8.3.2 Agent context parameter, *Concrete grammar*, <agent signature>

<aggregation kind> was missing in <agent signature>.

Change the syntax for <agent signature> to include <aggregation kind> before <sort> (twice), so that the revised syntax is:

<agent signature> ::=
<sort list>
| [<end>] **fpar** <aggregation kind> <sort> {, <aggregation kind> <sort> }

C.3.8 *Correction* – 8.3.12 Gate context parameter, *Concrete grammar*

In the paragraph after the syntax for delete the last sentence: "A type with a <gate context parameter> from which instances are defined shall have a <signal list> in any <gate constraint> of the <gate context parameter>."

In the next paragraph add a comma (",") after <gate constraint>.

C.3.9 *Clarification* – 8.4.1 Adding properties, *Semantics*

In item (c) of the list change "contains <actual context parameters>" to "contains an <actual context parameter list>"

C.3.10 *Textual correction* – 8.4.3 Virtual transition/save, *Concrete grammar*

In the first text paragraph, correct "clause 11.8 (virtual spontaneous transition)" to "clause 11.9 (virtual spontaneous transition)".

C.3.11 *Textual correction* – 8.8.3 Procedure context parameter, *Concrete grammar*

There was an error that matching the result of the procedure definition and signature did not apply for case (b). Insert "both have a result of the same <sort> or if neither returns a result, and " before the colon at the end of the 3rd text paragraph after the syntax, and delete ", and if both have a result of the same <sort> or if neither returns a result"..

C.3.12 *Clarification* – 9.2 Block, *Model*

Replace the last paragraph by the following paragraph and NOTE:

This transformation takes place after replacing agent definitions with typebased agent definitions and transforming context parameters, and before the transforming of remote procedures.

NOTE – The `set_v` and `get_v` procedures have a parameter that holds the complete value associated with the variable, so that using these procedures for a global block variable with a significant size (such as an array, vector, string, or large structure) to write or read an element of the variable is probably inefficient compared with providing explicit remote procedures that write or read just that element.

C.3.13 *Textual correction* – 9.4 Procedure, *Concrete grammar*

In the first text paragraph change "<formal context parameters> and <virtuality constraint> for procedures with context parameters, and" to "<formal context parameters>, <virtuality constraint>, and".

In the second text paragraph after the syntax for <exported> change "calling procedure" to "calling the procedure".

In the sixth text paragraph after the syntax for <exported> change the text
"in the nearest surrounding scope with the same name and signature as the exported procedure and this <remote procedure definition> is used"
to the text
"in a surrounding scope with the same name and signature as the exported procedure and the nearest such <remote procedure definition> is used".

C.3.14 *Clarification* – 10.4 Signal list area, *Concrete grammar*

After the first text paragraph, insert the following paragraph:

The condition in clause 12.1.2 of Basic SDL-2010 is extended to include remote procedures and remote variables: Each <signal list item> of the <signal list> in an <interface use list> of an <interface definition> shall be a <signal identifier> or an <interface identifier> or a <remote procedure identifier> or <remote variable identifier>.

C.3.15 *Textual correction* – 10.5 Remote procedure, *Concrete grammar*

Replace the paragraph starting "If <destination> in the <communication constraints> " by the paragraph and Note:

If a remote procedure is a value returning procedure, each action shall contain no more than one <remote procedure call body> used as an <expression0> for the same remote.

NOTE 1 – The constraint above on repeating calls of the same value returning remote more than once in a action is a consequence of the model, where the returned value is assigned to an implicit variable in the <remote procedure call body> transform inserted before the action (see below).

C.3.16 *Modification* – 10.5 Remote procedure, *Concrete grammar*

At the end of the *Concrete grammar*, insert the paragraph:

A <variable> of a <timer communication constraint> shall not be a global variable of a system (type) or block (type) except if the <timer communication constraint> is within the state machine actions of system (type) or block (type).

C.3.17 *Correction* – 10.5 Remote procedure, *Model*

In the third paragraph starting "There are two anonymously", replace
"implicit <signal definition>s for each <remote procedure definition> in an agent type or in the system for definition in a package used by the system. The <signal name>s in these <signal definition>s are"
by
"implicit <signal definition list> items for each <remote procedure definition> in a system definition. The <signal name> items in these <signal definition> items are".

In the next paragraph change the list item (a) before the figure to:

a) For each imported procedure, two implicit anonymous `Integer` variables (in this description called `n` and `newn`) are defined in the enclosing scope unit of the <remote

procedure call body>, and `n` is initialized to 0. The same two variables (`n` and `newn`) are used for every <remote procedure call body> in the scope unit for the same remote procedure.

NOTE 1 – The parameter `n` is introduced to recognize and discard reply signals of remote procedure calls that were left through associated timer expiry.

If remote procedure is a value returning procedure, there is an implicit anonymous variable (in this description called `res`) defined in the enclosing scope unit of the <remote procedure call body> with the sort returned by the procedure.

The <remote procedure call body> is transformed as below, so that the following is inserted before the action that contained the <remote procedure call body>, where in the output the **to** clause is omitted if the destination is not present, and the **via** clause is omitted if it is not present in the original expression:

After the figure, change "including an additional parameter" to "including the implicit variable `res` as an additional parameter", and after this insert two unnumbered enumeration items:

The transform is labelled with the label on the action containing the remote procedure call or a new label if this action is not labelled, and the preceding path is changed to join this label.

If a value returning remote procedure call is transformed, the true path above is terminated with a join to the action that contained the remote procedure call with a new label, and the remote procedure call is replaced by an access of the implicit variable `res` used to receive the returned value. Otherwise the remote procedure call action is removed, and the true path above is joined to the action following the remote procedure call action.

Before the list item (b) replace the text starting "In all states" and the associated figure and the text "is inserted" by the two unnumbered enumeration items:

In all other states, `pREPLY` is discarded. This is not explicitly modelled: instead the handling of `pREPLY` is left unspecified in the transformed concrete syntax except for the `pWAIT` states with the consequence that there is an implicit transition (see clause 11.8 of [ITU-T Z.103]) for other states that discards the signal.

If the <remote procedure call body> was directly enclosed by a <remote procedure call>, it is an <action> that is the <remote procedure call> and the transform replaces the <remote procedure call body>. Otherwise the <remote procedure call body> is a <value returning procedure call> as an <expression0>, and transform is inserted before the action that contained the <value returning procedure call>, and the <value returning procedure call> is replaced in this action by an access of the implicit variable used to receive the returned value.

Renumber NOTE 1 and NOTE 2 in the clause as NOTE 2 and NOTE 3 (because NOTE 1 introduced in the *Concrete grammar*).

C.3.18 *Correction* – 10.6 Remote variable, *Concrete grammar*

Replace the text between the syntax for <import expression> and the syntax for <export body>, by the paragraph:

If <destination> in the <communication constraints> of an <import expression> is an <agent identifier> or **this**, the <remote variable identifier> shall represent a remote variable contained in the interface of the agent type.

Before the heading *Model*, add the paragraph:

Each <action> shall contain no more then one <import expression> for the same remote variable.

C.3.19 *Correction* – 10.6 Remote variable, *Model*

After the paragraph starting "The import access is modelled", move the paragraph starting "If a default initialization" to be the second unnumbered enumeration item of the list item b) Exporter.

Change the next paragraph starting "There are two implicit" to:

There are two implicit signal definitions for each variable of a <remote variable definition> in a system definition. A <remote variable definition> that defines multiple variables is expanded to a <remote variable definition> list with one variable per <remote variable definition>. The <signal name>s in the implicit signal definitions are denoted in this model by `xQUERY` and `xREPLY` respectively, where `x` denotes an implicit <name> associated with the <remote variable definition>. The signals are defined in the same scope unit as the <remote variable definition>. The signal `xQUERY` has an argument of the predefined sort Integer and `xREPLY` has arguments of the sort of the variable and Integer.

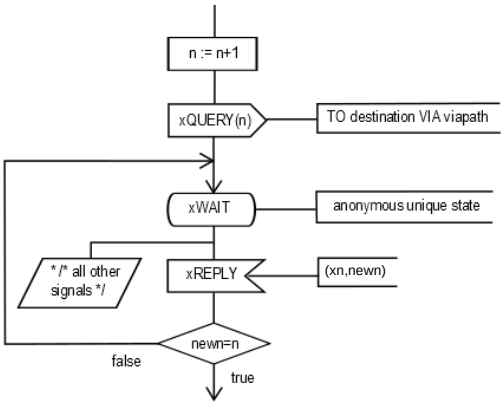
C.3.20 *Correction* – 10.6 Remote variable, *Model a) Importer*

Change the first unnumbered enumeration item to:

> For each imported variable, two implicit anonymous `Integer` variables (in this description called `n` and `newn`) are defined in the enclosing scope unit of the <import expression>, and `n` is initialized to 0. The same two variables (`n` and `newn`) are used for every <import expression> in the scope unit for the same remote variable. In addition, an implicit anonymous variable (in this description called `xn`) of the sort of the remote variable is defined for each <import expression>.

In the second unnumbered enumeration item, after "**`import`** `(x` **`to`** `destination` **`via`** `viapath)`" before the unnumbered enumeration item starting "Additionally, the following" change to:

> is transformed so that the following is inserted before the action that contained the <import expression>, where in the output the **`to`** clause is omitted if the destination is not present, and the **`via`** clause is omitted if it is not present in the original expression:



– The insertion is labelled with the label on the action containing the import expression or a new label if this action is not labelled, and the preceding path is changed to join this label.

– The true path above is terminated with a join to the action that contained the import expression with a new label.

– The import expression is changed to an access of the variable `xn`.

In all other states, `xREPLY` is discarded. This is not explicitly modelled: instead the handling of `xREPLY` is left unspecified in the transformed concrete syntax except for the `xWAIT` states with the consequence that there is an implicit transition (see clause 11.8 of [ITU-T Z.103]) for other states that discards the signal.

NOTE 1 – Until 2017 `xREPLY` was saved in other states: the change to discarded is consistent with remote procedures.

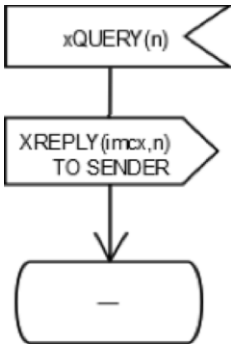
C.3.21 *Correction* – 10.6 Remote variable, *Model b) Importer*

Insert a new first unnumbered enumeration item:

For each exported variable, an implicit anonymous variable (in this description denoted by `imcx`) is defined to hold the exported value of the exported variable, and an implicit anonymous variable of type Integer (in this description denoted by `n`) is defined.

The second unnumbered enumeration item is the text starting "The import access is modelled" moved from the general *Model* description. In this text, change "implicit copy is" to "implicit copy `imcx` is".

In the text before the figure, delete " excluding implicit states derived from import," and replace the figure by:



After the figure, delete

For each such state, an implicit anonymous variable of sort `Pid` (in this description called `ivar)` and an implicit anonymous variable of type `Integer` (in this description called `n`) are defined."

Number the NOTE as NOTE 2 (a new NOTE 1 is added in *Model a) Importer*).

C.3.22 *Textual correction* – 11.2 State, *Abstract grammar*

Delete the paragraph starting "Each *Connect-node* in".

C.3.23 *Clarification* – 11.2 State, *Concrete grammar*

Before the syntax rule for <spontaneous association area> add the following Note:

NOTE 1 – Although the concrete grammar allows <continuous signal association area> and <spontaneous association area> for a <state area> with a <state list item> that is a <typebased composite state> or <composite state list item>, for a composite state in the abstract grammar *Spontaneous-transition* and *Continuous-signal* are not allowed, therefore they are only valid for basic state items.

C.3.24 *Modification* – 11.2 State, *Concrete grammar*

After the syntax rule for <state timer> add the following text paragraph and Note:

When two <state> items contain the same <state name>, a <state timer> shall not be specified for both <state> items or both <state> items shall specify the same <state timer>.

NOTE 2 – A <state timer> with **state timer** is a *Model* (see below).

and at the start of the next paragraph delete the sentence:

"A <state timer> with **state timer** <Time expression> represents a unique implicit *Timer-definition* with an anonymous name identified by the *Timer-identifier* of the *State-timer*."

C.3.25 *Textual correction* – 11.2 State*, Semantics*

In the last paragraph correct the reference to "11.8" to "11.9" (twice).

C.3.26 *Modification* – 11.2 State*, Model*

Add a *Model* clause with the following paragraph:

A <state timer> with **state timer** <Time expression> is equivalent to <state timer> with a <set clause> with a parameterless timer definition with an anonymous name set to the <Time expression>.

C.3.27 *Textual correction* – 11.8 Empty clause

Insert the level 2 heading "**11.8 Empty clause**" and the paragraph:

This clause is intentionally left blank.

C.3.28 *Textual correction* – 11.9 Spontaneous transition

This is renumbered. It was clause 11.8.

C.3.29 *Textual correction* – 11.10 Label

This is renumbered. It was clause 11.9. This replaces clause **11.10 Empty clause** which had the paragraph:

This clause is intentionally left blank.

C.3.30 *Clarification* – 11.11.2 State aggregation*, Semantics*

Change the first sentence of the fifth paragraph that starts "If there are signals " to:

If there are signals in the complete valid input set of the *Composite-state-type-definition* where a *State-aggregation-node* occurs that are not consumed by any *State-partition* of a *State-aggregation-node*, there is an implied additional *State-partition.*

C.3.31 *Textual correction* – 11.11.4 Connect*, Abstract grammar*

At the end of the *Abstract grammar*, insert the paragraph (previously in **11.2 State**, *Concrete grammar*):

Each *Connect-node* in the *Connect-node-**set*** of a composite state application shall either be the only *Connect-node* without a *State-exit-point-name* or have a *State-exit-point-name* that is different from every other *Connect-node* in the *Connect-node-**set***.

C.3.32 *Textual correction* – 11.12.2.4 Return*, Concrete grammar*

Remove the MSWord comment on the last line of the syntax for <return area>.

C.3.33 *Textual correction* – 11.14 Statement lists, *Concrete grammar*

Remove the MSWord comment on <return body> in the syntax for <return statement>.

C.3.34 *Clarification* – 11.14.1 Compound and loop statements, *Abstract grammar*

The representation of <loop step> items as *Step-graph-node* items implies that every *Step-graph-node* is a *Graph-node* that is a *Task-node*. This is clarified by adding after the syntax for *Step-graph-node* the following sentence:

A *Graph-node* of a *Step-graph-node* shall be a *Task-node*.

C.3.35 *Textual corrections* – 11.14.1 Compound and loop statements, *Concrete grammar*

The syntax of <loop step> allowed an alternative for a procedure call provided the procedure is not a value returning procedure. However, the <procedure call body> is stated to represent an *Expression* for an *Assignment*, which would require a value returning procedure, and a <value returning procedure call> is an <expression>. The correction is to remove the option for <procedure call body> from the <loop step> syntax and references to the <procedure call body> in the text. That is:

Revised syntax for <loop step>

<loop step> ::=
[**,** [<expression>]]

In the paragraph following the rule <loop step>, change "therefore if a <loop step> " to "therefore if a <loop step> of a <loop clause>" and change "<loop variable indication> shall" to "<loop variable indication> of the <loop clause> shall". In the same paragraph, replace "<expression> or <procedure call body>" by "<expression>" (twice).

Delete the paragraph starting "The keyword **call**" and the subsequent paragraph starting "The <procedure identifier>"

C.4 Z.103 changes

C.4.1 *Textual correction* – Language

Change all parts of the main body (from page 1) text marked as "French" or "Swiss French" or "English (US)" or "German (Switzerland)" to "English (UK)".

C.4.2 *Textual correction* – 8.1.1.1 Agent types, *Model*

In the second paragraph, third enumerated item replace "<actual context parameters>" by "<actual context parameter list>".

C.4.3 *Textual correction* – 8.1.4 Gates defined by interface gates, *Model*

In the second paragraph delete "<gate constraint> or ".

C.4.4 *Textual correction* – 9 Agents, *Concrete grammar*

In the syntax rule for <agent diagram> delete the line
[***is associated with*** <package use area>]

C.4.5 *Clarification* – 10.1 Channel, *Concrete grammar*

It is clarified that the derivation of signals for an omitted <signal list area> is only possible from an agent/state-machine that is typebased. After the sentence ending "in the direction for the <signal list

area>", insert the two sentences: "The signal set is defined by channel connection to an <agent area> only for a <typebased agent definition>. The signal set is defined by the channel connection to a <state machine area> only for a <state symbol> containing a <typebased composite state>."

C.4.6 *Modification* – 10.1 Channel, *Concrete grammar*

The derivation of signals for an omitted <signal list area> is no longer considered possible for the case that an internal channel does not have a defined signal list, but is connected to an agent with a defined signal list. Therefore change "all internal channels or agents" to "all internal channels".

C.4.7 *Clarification* – 10.1 Channel, *Model*

It is clarified that the model for a omitted <signal list area> is to insert a derived <signal list area> by replacing the paragraph:

If an associated <signal list area> is omitted from a <channel definition area>, the corresponding *In-signal-identifier-**set*** or *Out-signal-identifier-**set*** for the *Destination-gate* or *Originating-gate* of the channel is derived from the channel connection, if necessary.

with the paragraph:

If an associated <signal list area> is omitted from a <channel definition area>, the <signal list area> is replaced by a <signal list area> derived from the channel connection (see *Concrete grammar* above), that corresponds to the *In-signal-identifier-**set*** or *Out-signal-identifier-**set*** for the *Destination-gate* or *Originating-gate*.

C.4.8 *Textual correction* – 11.2 State, *Concrete grammar*

In the second paragraph after the syntax for <composite state list item>, move the *Model* heading to be after the sentence:

The <composite state name> of a <composite state list item> references a <composite state diagram>.

As a consequence the *Model* clause starts with a paragraph containing the sentence starting "A <composite state list item> is a shorthand".

Delete the subsequent paragraph starting "A <composite state list item>" (because it duplicates the first paragraph after the syntax for <composite state list item>).

Delete the next paragraph starting "A <state area> that is a <terminator area>" (because it had the same meaning as what was old first paragraph of the *Model* clause).

C.4.9 *Textual correction* – 11.2 State, *Model*

In the fourth paragraph (after moving *Model* heading and making changes above, previously third paragraph), replace the first sentence by:

When two <state area> items that each contain one <state name>, each contain the same <state name>, these <state area>s are combined into one <state area> having that <state name> with the <state timer> specified for one (or both if it is the same) <state> items.

In the fifth *Model* paragraph, change "under clause 11.11" to "under clause 11.11.1".

C.4.10 *Clarification* – 11.3 Input, *Concrete grammar*

Change the last paragraph before *Model*, starting " Basic SDL-2010 <via path> is extended to allow <channel identifier> as a shorthand for the gate." to:

Basic SDL-2010 <via path> is extended to allow <channel identifier> as a shorthand for the gate. The <channel identifier> for a <via path> of a <stimulus> or <save area> in a composite state type (that is, after application of the models for agent definition, agent structure with an interaction and channel to channel connection) shall identify a channel such that the enclosing state machine of the via path is reachable from the channel with the signal given in the stimulus or save through exactly one gate of the state machine. A <channel identifier> for a <via path> of <communication constraints> output in a composite state type (that is, after application of the models for agent definition, agent structure with an interaction, channel to channel connection, remote procedure and import expressions) shall identify a channel such that the enclosing state machine of the via path is reachable from the channel with the signal given in the stimulus or save through exactly one gate of the state machine. If a <via path> has a <channel identifier>, this shall not be a bidirectional channel with both ends connected to the same state machine.

C.4.11 *Clarification* – 11.3 Input, *Model*

In the last paragraph of *Model*, change "the identified channel nearest to the agent containing the <communication constraints>" to "the channel that connects (directly or indirectly) to the enclosing state machine".

C.4.12 *Textual correction* – 11.11.1 Composite state graph, *Model*

In the third *Model* paragraph, change "In the <composite state body area>, any part of a <qualifier>" to "In the <composite state structure area>, any part of a <qualifier>".

C.4.13 *Textual correction* – 11.11.2 State aggregation, *Model*

Remove the comment "Unlabelled entry/exit -> Z.102".

C.4.14 *Textual correction* – 11.13.1 Task, *Model*

The first *Model* paragraph starting "If a <task body>" should be deleted because a <task body> cannot be empty (according to the specified syntax).

C.4.15 *Clarification* – 11.15 Timer, *Model*

If the <reset body> is an <asterisk>, this is replaced by a bracketed <reset clause> list, with one <reset clause> for each timer definition of the agent (including those for state timers). Each replacement <reset clause> is of the form <timer identifier> <asterisk>.

A <set> is allowed to contain several <set clause> items. This is derived syntax for specifying a sequence of <set> items, one for each <set clause> such that the original order in which they were specified in the <set> is retained.

A <set statement> is allowed to contain several <set clause> items. This is derived syntax for specifying a sequence of <set statement> items, one for each <set clause> such that the original order in which they were specified in the <set> is retained.

A <reset> is allowed to contain several <reset clause> items. This is derived syntax for specifying a sequence of <reset> items, one for each <reset clause> such that the original order in which they were specified in the <reset> is retained.

A <reset statement> is allowed to contain several <reset clause> items. This is derived syntax for specifying a sequence of <reset statement> items, one for each <reset clause> such that the original order in which they were specified in the <reset> is retained.

The shorthand items for <set>, <set statement>, <reset> and <reset statement> are expanded before shorthand items in the contained expressions are expanded.

C.5 Z.104 changes

C.5.1 *Textual correction* – Language

Change all parts of the main body (from page 1) text marked as "French" or "French (Swiss)" or "English (US)" or "Spanish" or "German (Switzerland)" to "English (UK)".

C.5.2 *Textual correction* – Non-breaking hyphens in abstract syntax names

Abstract syntax rule names are always in italic, start with an uppercase letter followed by lowercase letters and hyphens and ending in a lowercase letter. To make it easy to find the abstract syntax rule names in the MSWord macros used to maintain SDL-2010, these should not be non-breaking hyphens. Change any non-breaking hyphen in abstract syntax names to normal (line breaking) hyphens. If this causes the line to break on a hyphen in an abstract syntax name, a line break (^l) is inserted immediately before the abstract syntax name.

C.5.3 *Clarification* – 8.1.4 Gates with encoding rules

Add "**and anonymous choice data type for a gate**" to the heading.

C.5.4 *Clarification* – 8.1.4 Gates with encoding rules and anonymous choice data type for a gate, *Abstract grammar*

At the end of the *Abstract grammar* insert the paragraph:

The *Encoding-rules* associated with a *Gate-definition* of a type based on a supertype shall specify the same set of *Encoding-rules* as the *Encoding-rules* of the corresponding gate definition in the supertype if that gate has *Encoding-rules*.

C.5.5 *Clarification* – 8.1.4 Gates with encoding rules and anonymous choice data type for a gate, *Concrete grammar*

In the second text paragraph after the syntax for <gate definition>, delete the first sentence: "A specification of … has *Encoding-rules*." In the second text paragraph after the syntax for <gate definition>, (in what was the second sentence) change "<inherited gate symbol 2>, and" to "<inherited gate symbol 2> of a type based on a supertype, and".

C.5.6 *Textual correction* – 8.1.4 Gates with encoding rules and anonymous choice data type for a gate, *Model*

Correct the spelling of "orthe" to "or the".

C.5.7 *Modification* – 10.7 Communication path encoding rules, encode and decode, *Abstract grammar*

Changes are made so that encode and decode procedures can be resolved in the abstract grammar.

Encoding-rules has *Encode-procedure-identifier* and *Decode-procedure-identifier* added to make the rule:

Encoding-rules :: *Rules-identifier*
Encode-procedure-identifier
Decode-procedure-identifier

In *Encoding-path*, *Rules-identifier* is replaced by *Encoding-rules* and the presentation is improved (by removing {} and changing the layout), to make the rule:

Encoding-path :: *Gate-identifier*
| *Data-type-identifier Encoding-rules*

In *Rules-identifier* "::" is replaced by "=".

New rules *Encode-procedure-identifier* and *Decode-procedure-identifier* as follows:

Encode-procedure-identifier = *Procedure-identifier*

Decode-procedure-identifier = *Procedure-identifier*

At the end of the third paragraph after the abstract syntax, the sentence "These built-in encode and decode procedures are implicit parts of **`package`** `Predefined`." is added.

In the fifth and sixth paragraphs after the abstract syntax "an additional `Encoding` literal" is replaced by "an `Encoding` literal".

In the paragraph starting "For a *Gate-identifier* of" delete " or reachable via a channel with the *Signal-identifier* of the *Encoding-expression* from the agent".

In the last paragraph before *Concrete grammar* delete " in a context where the context of the *Decoding-expression* is reachable via the *Encoding-path*".

C.5.8 *Modification* – 10.7 Communication path encoding rules, encode and decode, *Concrete grammar* and *Model*

Reasons for changes except the paragraphs added before *Semantics*:

- An <expression> as an <encoding expression> is not transformed here;
- the compatibility check between an <expression> of an <encoding expression> and the choice for the encoding path is done on the abstract grammar;
- omitting a <encoding path> is allowed in an <encoding expression> with an <expression> only if there is only one path with encoding;
- "using" the unique path is moved to the *Model* for both <encoding expression> and <decoding expression>.

Concrete grammar:

Delete the text paragraph starting "If an <encoding expression> …".

In the next (was third) paragraph "output of the signal " with "output (of the signal if a <signal identifier> is given)", and in the same paragraph delete "and, in this case, the encoding for that path is used".

In the next (now third) text paragraph delete "and, in this case, the encoding for that path is used".

Replace the last paragraph before *Semantics* with the following two paragraphs:

> The <interface identifier> of an <encoding path> represents the *Data-type-identifier* of an *Encoding-path*. The <rules identifier> of an <encoding path> represents the *Rules-identifier* of the implicit *Encoding-rules* for the *Encoding-path*.
>
> The encode and decode procedures associated with a <rules identifier> shall be visible where the <rules identifier> is used.

Model:

Delete the first text paragraph starting "If an <encoding expression>".

Add two new paragraphs:

> If <encoding path> is omitted from <encoding expression>, the unique path required by the condition in the *Concrete grammar* is inserted.

If <encoding path> is omitted from <decoding expression>, the unique path required by the condition in the *Concrete grammar* is inserted.

C.5.9 *Modification* – 11.3 Input, *Concrete grammar*

Before the syntax rule <encoded input>, insert the paragraph:

A <variable> of an <in choice> shall not be a global variable of a system (type) or block (type) except if the <in choice> is within the state machine actions of system (type) or block (type).

and, at the end of the *Concrete grammar*, insert the paragraph:

A <variable> of an <encoded input> shall not be a global variable of a system (type) or block (type) except if the <encoded input> is within the state machine actions of system (type) or block (type).

C.5.10 *Textual correction* – 11.3 Input, *Concrete grammar*

Delete NOTE 1 (it is duplicated as NOTE 2 in 11.3 *Model*).

C.5.11 *Textual correction* – 11.3 Input, *Model*

The model is refined and changed to use the **signal** variable rather than implicit local variables. The last text paragraph in the *Model* is deleted and incorporated into the revised third text paragraph (before the NOTE) which is:

> An <input part> for an <encoded input> is transformed to list of <input part> items, one for each signal this is receivable with the <encoded input>. For each of the signals that are receivable from the gate or channel specified in an <encoded input>, there is an <input part> with an <input list> item for the signal, an <in choice> item for the implicit **signal** variable and a <via path> for the gate or channel. For each of the signals of an interface identified by an <interface identifier> of an <encoding path> of an <encoded input>, there is an <input part> with an <input list> item for the signal, an <in choice> item for a <signal expression> and an undefined <via path>. Each revised <input part> contains the same <enabling condition> as the original <input part> for the <encoded input>. The <transition> of each revised <input part> contains a task, where the <variable> given in the <encoded input> is assigned the value of an <encoding expression> for the value in the implicit **signal** variable and the <encoding path> of the <encoding expression>. This task is followed by the original transition for the <encoded input>.

Rename NOTE 2 as NOTE (it duplicated NOTE 1 in 11.3 *Concrete syntax* that has been deleted).

The last text paragraph after the NOTE is deleted (see above).

C.5.12 *Modification* – 11.4 Priority input

Replace the clause body by:

Priority input is extended to allow a choice value to be stored for the signal that was input.

Concrete grammar

<priority input list> ::=
<priority stimulus> {**,** <priority stimulus>}*
| <asterisk input list> [<in choice>] [<priority clause>]

The <priority input list> syntax of [ITU-T Z.102] is extended to allow the storing the signal value for an <asterisk input list> according to an <in choice>.

<priority stimulus> ::=
<stimulus> [<in choice>] [<priority clause>]

The <priority stimulus> syntax of [ITU-T Z.102] is extended to allow the storing the signal value according to an <in choice>.

C.5.13 *Clarification* – 11.7 Save

After "[ITU-T Z.101]" insert ", [ITU-T Z.102] and [ITU-T Z.103]".

C.5.14 *Clarification* – 11.8 Implicit transition

Change "[ITU-T Z.101]" to "[ITU-T Z.103]".

C.5.15 *Clarification* – 11.9 Spontaneous transition

Change "[ITU-T Z.101]" to "[ITU-T Z.102]".

C.5.16 *Clarification* – 11.10 Label

After "[ITU-T Z.101]" insert ", [ITU-T Z.102] and [ITU-T Z.103]".

C.5.17 *Clarification* – 11.11 State machine and composite state

After "[ITU-T Z.101]" insert ", [ITU-T Z.102] and [ITU-T Z.103]".

C.5.18 *Clarification* – 11.12 Transition

After "[ITU-T Z.101]" insert ", [ITU-T Z.102] and [ITU-T Z.103]".

C.5.19 *Clarification* – 11.13.1 Task

After "[ITU-T Z.101]" insert ", [ITU-T Z.102] and [ITU-T Z.103]".

C.5.20 *Clarification* – 11.13.2 Create

After "[ITU-T Z.101]" insert " and [ITU-T Z.103]".

C.5.21 *Clarification* – 11.13.3 Procedure call

After "[ITU-T Z.101]" insert " and [ITU-T Z.102]".

C.5.22 *Textual correction* – 11.13.4 Output, *Abstract grammar*

The items *Expression* and *Encoded-expression* need to be distinguishable in the abstract grammar. Therefore change:

Encoded-expression = *Expression*

to

Encoded-expression :: *Expression*

C.5.23 *Clarification* – 11.13.5 Decision

After "[ITU-T Z.101]" insert ", [ITU-T Z.102] and [ITU-T Z.103]".

C.5.24 *Clarification* – 11.14 Statement list

After "[ITU-T Z.101]" insert ", [ITU-T Z.102] and [ITU-T Z.103]".

C.5.25 *Clarification* – 11.15 Timer

After "[ITU-T Z.101]" insert ", [ITU-T Z.102] and [ITU-T Z.103]".

C.5.26 *Extension* – 12.1 Data definitions, *Abstract grammar*

Add an *Abstract grammar* clause as follows:

Abstract grammar

Value-data-type-definition	::	*Sort* [*Data-type-identifier*] *Literal-signature-**set*** *Static-operation-signature-**set*** *Procedure-definition-**set*** *Data-type-definition-**set*** *Syntype-definition-**set*** *Variable-definition-**set*** [*Default-initialization*] [*Abstract*]

Value-data-type-definition is extended include a *Variable-definition* set for synonym definition <entity in data type> items of the <data type definition> represented by the *Value-data-type-definition*. See clause 12.1.8.3 Synonym definition.

C.5.27 *Clarification* – 12.1 Data definitions, *Model*

Change

"or syntype definition in the context of which" in the 1st paragraph

to

"(or syntype definition) in the context where".

C.5.28 *Textual correction* – 12.1 Data definitions, *Model*

In the first paragraph, change "in which" to "where".

Change "is is" in the 2nd paragraph to "is".

C.5.29 *Textual correction* – 12.1.2 Interface definition, *Concrete grammar*

In the paragraph after the syntax for <as interface>, after "<interface definition>" insert "(or implicit interface definition)", and on the third line after "identifies the *Interface-definition*" insert a comma.

C.5.30 *Clarification* – 12.1.2 Interface definition, *Concrete grammar*

Before the rule <interface heading>, insert the paragraph (which is a sentence deleted from the paragraph after the rule <as interface>):

An <interface definition> (or implicit interface definition) defines a *Data-type-definition* of a choice data type with a unique anonymous *Sort* name in the context that the *Interface-definition* is visible.

C.5.31 *Clarification* – 12.1.2 Interface definition, *Model*

Delete the penultimate paragraph of the *Model* starting "The implicit interface for".

C.5.32 *Textual correction* – 12.1.6.2 Structure data types, *Concrete grammar*

In the syntax for <fields of sort>, <aggregation kind> <field name> is replaced (twice) by <field of kind>. This is to match the syntax in F2.

The representation of <structure definition> in the abstract grammar is clarified and extended for the case if `S` has a <data type specialization>.

The text of enumerated item (a) is clarified:

"data type specialization, if no operator named `Make` is given with"

is replaced by

"<data type specialization>, if no constructor (an operator named `Make` with"

and "the `S` structure sort," is replaced by "the `S` structure sort) is given,".

After the text of enumerated item (a), a new NOTE 1 is added:

NOTE 1 – In the absence of <data type specialization> there is only one constructor `Make` operator to inherit, except in the case that more than one constructor `Make` operator is explicitly given.

The text of enumerated item (b) is clarified:

"no operator named `Make` is given" is replaced by "no constructor (an operator named `Make`",

"the `S` structure sort, represents (in the *Operation-signature* set of the *Data-type-definition* for `S`)" is replaced by "the `S` structure sort) is given,"

and "each inherited *Operation-signature*" is replaced by "each (normally one) inherited *Operation-signature*".

In the paragraph after the alphabetic enumeration list, "generic operations" is replaced by "generic constructor operations".

C.5.33 *Textual correction* – 12.1.6.2 Structure data types, *Semantics*

The text paragraph is replaced by new text and a NOTE:

When no constructor is given for a `Make` of a structure data type `T` that inherits from a structure data type `S`, in a `Make` of `T` the result for each field inherited from `S` is the same as the result for that field from a `Make` of `S` with the same actual values for the inherited parameters. The remaining fields of the result a `Make` of `T`, each field is associated with the result of the corresponding parameter, or if no value is given for the field, the default initialization for that field, or "undefined" if there is no default initialization for the field.

NOTE 2 – When no constructor is given for a `Make` of a structure data type `T` that inherits from a structure data type `S`, and the `Make` of `S` is the generic operator named `Make` as in [ITU-T Z.101] clause 12.1.6.2 *Concrete grammar*, the `Make` of `T` is same as the generic operator named `Make` in [ITU-T Z.101] clause 12.1.6.2 *Concrete grammar*.

C.5.34 *Textual correction* – 12.1.6.2 Structure data types, *Model*

The text "<aggregation kind> <field name> pair" is replaced (three times) by "<field of kind>" to match the revised syntax for <fields of sort>.

The transformation of methods is deleted, because in Z.101 `fnExtract`, `fnModify` and `fnPresent` are specified as operators.

C.5.35 *Clarification* – 12.1.8.3 Synonym definition, *Concrete grammar*

In the paragraph following NOTE 1, change "with special property" to "with the special property".

Delete the paragraph (starting "The <constant expression>") following NOTE 2. This paragraph is re-inserted below.

In the next paragraph replace " and the *Sort-reference-identifier* " by ". If the <sort> in the <synonym definition> is omitted the *Sort-reference-identifier* of the *Variable-definition* ".

In the next paragraph replace ", and the sort of the <constant expression> " by ". If a <sort> is specified, the sort of the <constant expression>".

After this paragraph, insert the following two paragraphs:

The *Variable-definition* has a **PART** *Aggregation-kind*.

The <constant expression> in the concrete syntax denotes a *Constant-expression* in the abstract syntax as defined in [ITU-T Z.101].

C.5.36 *Clarification* – 12.1.8.3 Synonym definition, *Model*

Add a *Model* clause as follows:

Model

A <synonym definition> that defines multiple synonyms is a shorthand for a sequence of <synonym definition> items, each defining one synonym.

C.5.37 *Textual correction* – 12.2.1 Expression and expressions as actual parameters, *Abstract grammar*

In the rule *Agent-instance-pid-value* replace "=" by "::".

In the rule *Agent-instance* replace "=" by "::".

C.5.38 *Textual correction* – 12.2.1 Expression and expressions as actual parameters, *Concrete grammar*

The concrete syntax rules for <agent instance pid value> and <agent instance> are modified, so that the <agent instance> list is in <agent instance pid value> rather than <agent instance>. The revised rules are:

<agent instance pid value> ::=
system [<name>]
{ **value** < agent instance> **endvalue**
| <left curly bracket> <agent instance> <right curly bracket> }*

<agent instance> ::=
[**block** | **process**] <agent name>
[<left square bracket> <Natural expression> <right square bracket>]

At the end of the first paragraph after the rule <agent instance>, add the following description for first item of the list in *Agent-instance-pid-value*:

> The keyword **system** in <agent instance pid value> represents the first item of the *Agent-instance* list of the *Agent-instance-pid-value* with the *Name* for the system and an *Instance-number* for the `Natural` number "1".

C.5.39 *Clarification* – 12.3.4.5 Import expression, *Model*

The last sentence of the *Model* is replaced by:

It is not valid for an <import expression> to occur several times in an expression or one action.

C.5.40 *Clarification* – 12.3.4.5 Generic system definition

The first paragraph and NOTE in this clause are deleted, leaving only the text:

See [ITU-T Z.102] and [ITU-T Z.103].

C.5.41 *Clarification* – 12.3.4.6 Any expression, *Abstract grammar*

In the *Concrete grammar*:

- The paragraph "An <expression> in <actual parameters> corresponding to a formal **in**/**out** or **out** parameter shall not be omitted and shall be a <variable identifier>." is deleted, because this is defined as a condition on the *Abstract grammar* in 11.13.3 in Z.101.
- The paragraph "After the *Model* for **this** has been applied, the <procedure identifier> shall denote a procedure that contains a start transition." is deleted, because 9.4 Abstract grammar of Z.101 states " all potentially instantiated procedures shall have a *Procedure-start-node*."
- The paragraph "If **this** is used, <procedure identifier> shall denote an enclosing procedure." is deleted, because the condition does not apply.
- In the paragraph starting "The <remote procedure call body> represents " change "below" to "in clause 10.5 Remote Procedure of [ITU-T Z.102].

In the *Model*:

Remove the two text paragraphs that constrain the sort to have at least one element, because a sort always has at least one element.

C.5.42 *Textual correction* – 12.3.4.9 Signallist expression, *Semantics*

Replace "`Batural`" by "`Natural`".

C.5.43 *Modification* – 12.3.5 Value returning procedure call, *Concrete grammar*

Replace the text paragraph starting "If the <procedure identifier> is" by the Note:

NOTE – When the <procedure identifier> is not defined within the enclosing agent, the procedure call is transformed into a call of a local, implicitly created, subtype of the procedure as described in clause 11.13.3 *Model* of [ITU-T Z.101].

C.5.44 *Modification* – 14.1 Boolean sort, 14.1.1 Definition

The Boolean sort should be "unordered", because "<", ">","<=",">=", first, last, pred, succ, and num are not implicitly defined. Replace

```
  literals true, false;
```

by

```
  literals unordered true, false;
```

C.5.45 *Textual correction* – 14.7.1 Definition (for Real Sort)

Change the font from red to Auto for: (" **or** ('e' **or** 'E') (" **or** '+' **or** '-') ('0':'9')* ('0':'9')).

C.5.46 *Correction* – 14.7.1 Definition (for Real Sort)

To be consistent with <real name> in Z.101 the definition of literals is changed to:

```
          ('0':'9')* ('0':'9')'.'('0':'9')('0':'9')*
            ('' or (('e' or 'E') ('' or '+' or '-' ) ('0':'9') ('0':'9')* ) );
/*that is, decimal notation with an optional exponent */
```

C.5.47 *Correction* – 14.7.1 Definition (for Real Sort)

To be consistent with <real name> in Z.101 the definition of literals is changed to:

```
          ('0':'9')* ('0':'9')'.'('0':'9')('0':'9')*
            ('' or (('e' or 'E') ('' or '+' or '-' ) ('0':'9') ('0':'9')* ) );
/*that is, decimal notation with an optional exponent */
```

C.5.48 *Correction* – 14.11.1 Definition (for Duration Sort)

The Duration sort should have the same literals as the Real sort, therefore replace:

```
  literals unordered nameclass ('0':'9')+ or (('0':'9')* '.' ('0':'9')+);
```

with

```
  literals unordered nameclass
           ('0':'9')* ('0':'9') '.' ('0':'9') ('0':'9')*
             ('' or (('e' or 'E') ('' or '+' or '-' ) ('0':'9') ('0':'9')* ) );
```

C.5.49 *Correction* – 14.12.1 Definition (for Time Sort)

The Time sort should have the same literals as the Real sort, therefore replace:

```
  literals unordered nameclass ('0':'9')+ or (('0':'9')* '.' ('0':'9')+);
```

with

```
  literals unordered nameclass
           ('0':'9')* ('0':'9') '.' ('0':'9') ('0':'9')*
             ('' or (('e' or 'E') ('' or '+' or '-' ) ('0':'9') ('0':'9')* ) );
```

C.5.50 *Correction* – 14.18 Support for ASN.1 character, symbol string and NULL types

The definition of the character string literals of `PrintableString` is incorrect and includes characters that are not allowed. After the line of `PrintableString` containing "**operators** ocs **in nameclass**", replace:

```
             '''' ( (' ':'&') or '''''' or ('(': '?') )+ '''' -> this PrintableString;
/*   character strings of any length of any characters from a space ' ' to a '?'      */
```

by

```
           '''' (
                ' ' or ''''''
             or  ('0': '9') or ('A': 'Z') or ('a': 'z')
             or  '(' or ')' or '+' or ',' or '-' or '.' or '/' or ':' or '=' or '(?'
           )+ '''' -> this PrintableString;
/*   printable character strings of any length */
```

C.5.51 *Textual correction* – A.2.7 Unordered literals, *Semantics*

Replace "(see clause 12.1.6.17) " by "(see clause 12.1.6.1 ITU-T Z.101)", replace "*Model*" by "*Semantics*", delete "the *Model* in clause 12.1.6.17 is not applied. Consequentially, " and after "this data type" insert " (see clause 12.1.6.1 ITU-T Z.101)".

C.5.52 *Textual correction* – A.2.7 Unordered literals, *Semantics*

Replace "(see clause 12.1.6.17) " by "(see clause 12.1.6.1 ITU-T Z.101)", replace "*Model*" by "*Semantics*", delete "the *Model* in clause 12.1.6.17 is not applied. Consequentially, " and after "this data type" insert " (see clause 12.1.6.1 ITU-T Z.101)".

C.5.53 *Textual correction* – Annex C – MSWord comment on title

Delete the comment on the title "Replaced by Annex C of Z.104 (2011) Amendment 1 (10/12) - NOT change marked.".

C.6 Z.105 changes

C.6.1 *Textual correction* – Language

Change all parts of text marked as "English (US)" to "English (UK)".

C.6.2 *Textual correction* – 9.2 Parameterized type assignment, *Example*

Change "synonym **maxSize** <<package " to "synonym **maxSize** <<package ".

C.6.3 *Textual correction* – 9.3 Referencing ASN.1 parameterized type definitions, *Model*

In the second paragraph, change "<actual context parameters>" to "<actual context parameter list>".

C.6.4 *Textual correction* – 10 Definitions in package Predefined for SDL-2010, first line

Change the "and" after "`Predefined`" from the character style "z100code" to the default paragraph font (that is: apply the Normal style).

C.7 Z.106 changes

C.7.1 *Textual correction* – Language

Change all parts of the main body (from page 1) text marked as "French" or "Swiss French" or "English (US)" or "Spanish" or "German (Switzerland)" to "English (UK)".

C.7.2 *Textual correction* – Introduction, page v

Insert a full stop and a paragraph mark at the end of the text in the second paragraph after "information".

C.7.3 *Clarification* – Syntax rule name <composite state>

The rule <composite state> has been renamed <composite state definition> throughout the document.

C.7.4 *Clarification* – 5.5 Agents

The syntax rule <entity in agent> has its alternatives sorted alphabetically.

C.7.5 *Textual correction* – 5.6.1 Channel

At the end of enumeration item (c) delete ", respectively".

C.7.6 *Clarification* – 5.6.1 Channel

Before 5.6.2 Connection add the following NOTE and paragraph:

NOTE – Clause 10.1 *Concrete grammar* of [ITU-T Z.103] contains a description of whether it is possible in the graphical grammar to derive the set of signals for a <signal list area> that has been omitted in a <channel definition area>. The equivalent set of rules for deriving an omitted <signal list> from a <channel path> of a <channel definition> is given below. Equivalent rules for the textual grammar are not normally given as they can usually be simply implied from the rules on the graphical grammar, but to make the rules for the textual grammar clear in this case an explicit description is given.

Derivation of an omitted channel <signal list> is possible if at least one <channel endpoint> identifies a <textual typebased agent definition> or <typebased composite state> (or **`this`** and the state machine is a <typebased composite state>) or **`env`**, and the gate for the <channel endpoint> has a defined set of signals in the direction for the <signal list>. The set of signals is defined for the gate if it identifies a <textual gate definition> that has a <gate constraint> with a defined <signal list>, or if a signal set is defined for all internal channels connected to this gate. The set is defined for a gate connected to <external channel identifiers> if for each external channel either no <signal list area> is omitted or the set for that external channel is derivable.

C.7.7 *Clarification* – 5.7.11.1 Transition body

After the syntax for <terminator node> and the paragraph starting "If the <terminator> of a <transition>" add the extensive explanatory NOTE:

NOTE – In the graphical grammar, <terminator area> has alternatives for <state area>, <merge area>, <decision area> and <transition option area>. A<state area> is transformed in the graphical grammar as described in clause 11.2 *Model* of [ITU-T Z.103] to a <nextstate area> as the <terminator area> of the <transition area> and a <state area> that is not a <terminator area> of a <transition area> (see *Model* in clause 11.12.1 of [ITU-T Z.103]). The equivalent textual grammar has a corresponding <terminator> that is a <nextstate> terminator and a <state>. A <merge area> is transformed in the graphical grammar as described in 11.12.2.2 *Model* [ITU-T Z.103] to an <out connector area> to a unique <connector name> and attaching an <in connector area>, with the same <connector name> to the <flow line symbol> in the <merge area>. The equivalent textual grammar has <join> items for the <out connector area> items and a <label> for the <connector name>. The <decision area> and <transition option area> are topologically different to <decision> and <transition option> as a textual grammar <action item>, because in the textual grammar if a <terminator> is not specified the next item in the <transition string> follows, whereas in the graphical grammar each <transition area> of a <decision area> or <transition option area> has a <terminator>. Any <decision area> or <transition option area> as a <terminator area> in the graphical grammar, in the textual grammar becomes a <decision> or <transition option> (respectively) as the last <action item> of a <transition string> for the <transition string area> and the <terminator area>. In this case, the <transition string> is not followed by a <terminator>.

C.7.8 *Correction* – 5.7.11.2.2 Join

The sentence referring to clause 8.3.1 of Z.101 is removed – there is no such clause and in any case <agent body> only occurs in Z.106. Delete "The rule for <agent body> in a type definition is stated in clause 8.3.1 of [ITU-T Z.101].".

C.7.9 *Textual correction* – 5.7.11.2.4 Return

Remove the comment on <return body>:

"Z.101 changed so that <expression> in <return body> is optional (see <return area> in Z.101)."

C.8 Z.107 changes

C.8.1 *Textual correction* – Language

Change all parts of the main body (from page 1) text marked as "French" or "Swiss French" to "English (UK)".

C.8.2 *Textual correction* – 12.1 Data definitions, *Abstract grammar*

In the rule *Value-data-type-definition* replace "*Null-literal-signature*" by "[*Null-literal-signature*]" because "Every value sort contains a unique named element, the *Null-literal-signature* …".

C.8.3 *Clarification* – 12.1 Data definitions

Introduce a heading "*Concrete grammar*" after the abstract syntax rule *Value-data-type-definition*.

Before the heading "*Semantics*" insert the following paragraph and the subsequent three NOTEs:

The *Result-aggregation* of the *Result* of the *Literal-signature* of the *Null-literal-signature* of a value sort is **REF**.

NOTE 1 – The *Literal-natural* of the *Literal-signature* for a *Null-literal-signature* is arbitrary and is not taken into account when determining other *Literal-natural* values of *Literal-signature* items of a *Value-data-type-definition*.

NOTE 2 – If the name `null` occurs in a context that could be a **PART** or **REF** the specification would be ambiguous and consequently invalid, therefore the name `null` should not be used for literals.

NOTE 3 – The *Result-aggregation* of the *Result* of any *Literal-signature* of a value sort except the *Null-literal-signature* is **PART**.

C.8.4 *Textual correction* – 12.1.3 Operation signature, *Abstract grammar*

Change "=" to "::" in the rule *Dynamic-operation-signature* making it:

Dynamic-operation-signature :: *Operation-signature*

C.8.5 *Clarification* – 12.1.4 Generic data type operations, *Concrete grammar*

Insert a NOTE before the *Semantics*:

NOTE – If the name `Null` occurs in a context that could be a **PART** or **REF** the specification would be ambiguous and consequently invalid, therefore the name `Null` should not be used for literals.

The reference to 12.2.8 is changed to 12.2.7, because clauses 12.2.3 to 12.2.8 are re-ordered in the same order as Z.101, Z.102, Z.103 and Z.104 and therefore renumbered.

C.8.6 *Textual correction* – 12.2.1 Expressions, *Abstract grammar*

The reference to 12.2.8 is changed to 12.2.7, because clauses 12.2.3 to 12.2.8 are re-ordered in the same order as Z.101, Z.102, Z.103 and Z.104 and therefore renumbered.

C.8.7 *Clarification* – 12.2.1 Expressions, *Semantics*

The reference to 12.2.8 is changed to 12.2.7, because clauses 12.2.3 to 12.2.8 are re-ordered in the same order as Z.101, Z.102, Z.103 and Z.104 and therefore renumbered.

At the end of the *Semantics*, text, add the following:

An expression aggregation kind is **PART** or **REF**.

A *Constant-expression* has a **PART** aggregation kind except if it is a *Literal* that identifies a *Null-literal-signature*.

NOTE – The aggregation kind of *Literal* is the *Result-aggregation* of the *Result Literal-signature* of the identified *Result-aggregation*, which is only **REF** for the *Null-literal-signature* of the value sort.

An *Active-expression* that is a *Conditional-expression* or *Equality-expression* or *Imperative-expression* or *Range-check-expression* or *Encoding-expression* or *Decoding-expression* or *Agent-instance-pid-value* or *Type-check-expression* or *Type-coercion* has a **PART** aggregation kind.

For the aggregation kind of an *Operation-application* see clause 12.2.6, *Variable-access* see clause 12.3.2 and *Value-returning-call-node* see clause 12.3.5.

C.8.8 *Textual correction* – 12.2.3 to 12.2.8

The clause **12.2.3 Synonym** is moved to **12.2.8** (to be in the same order as Z.101, Z.102, Z.103 and Z.104) and the intervening clauses renumbered.

C.8.9 *Clarification* – 12.2.5 (was 12.2.6) Conditional expression, *Semantics*

At the end of the *Semantics*, add the following paragraph:

If the selected *Consequence-expression* or *Alternative-expression* has an aggregation kind **REF**, the value returned by the result of the selected expression is used, and the *Conditional-expression* has a **PART** aggregation kind.

C.8.10 *Clarification* – 12.2.6 (was 12.2.7) Operation application, *Semantics*

At the end of the *Semantics*, add the following paragraph:

An *Operation-application* has an aggregation kind that is the *Result-aggregation* (**PART** or **REF**) of the procedure that implements the operation identified by the *Operation-signature* for the operation.

C.8.11 *Textual correction* – 12.2.7 (was 12.2.8) Range check expression, *Abstract grammar*

Remove the following sentence (for an incorrect condition):

The sort of the *Expression* shall be sort compatible with the *Parent-sort-identifier*.

C.8.12 *Clarification* – 12.3.1 Variable definition, *Abstract grammar*

After the rule for *Aggregation-kind* add the following paragraph followed by the heading *Semantics*:

The *Aggregation-kind* of a *Variable-definition* with a *Sort-reference-identifier* for a pid sort shall be **PART**.

C.8.13 *Clarification* – 12.3.2 Variable Access, *Semantics*

At the end of the *Semantics*, add the following paragraph:

A *Variable-access* has an aggregation kind that is the *Aggregation-kind* of the *Variable-definition*: **PART** or **REF**.

C.8.14 *Clarification* – 12.3.3 Assignment, *Semantics*

In enumerated item (a) enumerated item (1), after the first sentence ending "with the identified variable." and before the sentence starting "otherwise" insert:

If the *Expression* is a *Variable-identifier* with aggregation kind **PART**, the variable is associated with the *Variable-identifier* of the *Expression*. If the *Expression* is a *Variable-identifier* with aggregation kind **REF**, the variable is associated with the (reference) value associated with the *Variable-identifier* of the *Expression*. If the *Expression* is a field access *Operation-application*, the variable is associated with a reference to the data item enclosing the field (for example a *Variable-identifier*) and the name of the field. If the *Expression* is a *Literal* that identifies the *Null-literal-signature* of a *Value-data-type-definition*, the variable is associated with the *Null-literal-signature*.

C.8.15 *Clarification* – 12.3.4 Imperative expression

So that clause 12.3.5 can be added to clarify aggregation for a value returning procedure call, add the clause "**12.3.4 Imperative expression**" with a body "See [ITU-T Z.104].".

C.8.16 *Clarification* – 12.3.5 Value returning procedure call

To clarify aggregation for a value returning procedure call, add the clause "**12.3.5 Value returning procedure call**" with the body:

The description is extended from [ITU-T Z.104] to clarify its aggregation kind.

Semantics

A *Value-returning-call-node* has a aggregation kind that is the *Result-aggregation* of the called procedure: **PART** or **REF**.

_____