INTERNATIONAL TELECOMMUNICATION UNION

TELECOMMUNICATION STANDARDIZATION SECTOR

STUDY PERIOD 2005-2008

COM 17 – C 39 – E

June 2006

English only

Original: English

Question(s):	11, 13/17

STUDY GROUP 17 – CONTRIBUTION 39

Source:	ITU-T Study Group 17
Title:	Specification and Description Language Implementers' Guide - version 1.0.1 – April 2006

Agreed to at the 28 April 2006 plenary session of ITU-T Study Group 17

Contact: Rick Reed
Rapporteur Q.11/17

Tel: +44 15394 88462
Fax: +44 15394 88218
Email: rickreed@tseng.co.uk

SPECIFICATION AND DESCRIPTION LANGUAGE IMPLEMENTERS' GUIDE

Version 1.0.1

April 2006

SPECIFICATION AND DESCRIPTION LANGUAGE IMPLEMENTERS' GUIDE

1. Introduction

This Guide is a compilation of reported defects and their resolutions to the Specification and Description Language (SDL) ITU-T Recommendations:

- Z.100, Z.104, Z.105, Z.106, Z.107 and Z.109.

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.

1.1 Scope of the Guide

The Guide records the resolution of defects and maintenance in the following categories as described in Z.100 Appendix II Guidelines for maintenance of SDL:

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

NOTE: This Guide only addresses proposed additions, deletions, or modifications to the Recommendations that are strictly related to maintenance of SDL as described in Z.100. Proposals for new features should be made in the normal way through contributions to ITU-T Study Group 17.

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 (http://www.itu.int/ITU-T/).

1.4 Contact

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

Mr. B. Georges SEBEK
ITU/TSB
Place des Nations
CH-1211 Geneva 20
Switzerland

Tel: +41 22 730 5994
Fax: +41 22 730 5853
Email: sebek@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 Z.100, 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 no individual responses can be given to those submitting reports, and that the procedure is not intended as a consulting service.

2.3 Documenting the Resolution of Defects

The following ITU-T Recommendations have errors or agreed changes. The defects and their resolutions are recorded in Annex B.

- ITU-T Recommendation Z.100
- ITU-T Recommendation Z.106
- ITU-T Recommendation Z.109

Annex A

Change Request Form

Please supply the following details.		
Type of change:	error correction simplification modification	clarification (or question) extension 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 (Z.100) Rapporteur, c/o ITU/TSB, Place des Nations, CH-1211 Geneva 20, Switzerland. Fax: +41 22 730 5853, e-mail: tsbmail@itu.int

Annex B

This is the master list of changes for the Z.100 (SDL) series Recommendations approved by Study Group 17 in April 2006.

History: The previous version of this list was published in version 1.0 of this document, which replaced 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 Z.100 (08/2002), 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 (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 Z.100, Appendix II);

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

The rules for maintenance in Z.100, Appendix II 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 Z.100 is revised."

B.2 Terminology

An *error* is an inconsistency in one or more Recommendations 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

Z.100 (11/99), Appendix II documents the procedure to be followed for the maintenance of Recommendations Z.100, 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

Much of the previous SDL study work (under Q.6/10, then Q.13/17, then Q.N/17 [2001-2004] before Q.11/17 [2005-2008]) took place by electronic correspondence hosted at meeting@sdl-forum.org. As well as emails between participants there have been discussion documents posted on ftp servers.

Where a reference is given in parentheses after the heading, this refers to a discussion document that can be found on the ftp servers for SDL discussion at ftp://username:password@ties.itu.int/u/tsg17/sg17/documents
or the SG 17 web page at www.itu.int/ITU-T/studygroups/com17,
or ftp://username:password@ftp.sdl-forum.org/sg17/.

Both of these ftp servers and the ITU-T web access require a username and password for access.

The following changes have been agreed but are not yet included in a document approved for consent.

B.4.1 Textual corrections – spelling, minor grammar, and textual errors throughout the document (COM 17 TD3174 Jeju April 2006)

A thorough grammar and spelling check was made of the plain text version of the document (that is before using the special Z.100 macros to insert bookmarks and cross references). The grammar and spelling was checked against the Microsoft Word 2004 UK English data. Standard writing style grammar rules were used with the following options selected: Capitalization, Commonly confused words, Hyphenated and compound words, Misused words, Negation, Passive sentences, Possessives and plurals, Punctuation, Relative clauses, Subject verb agreement, Verb and noun phrases, and Spaces between sentences (=1). As a result: some spelling errors were corrected, in some lists the first word was not capitalized and the word was therefore marked as *no proofing*, list items ending in a semicolon had the semicolon marked as *no proofing*, and sentences beginning with a <syntax item> were in some cases changed to start with a capitalized word (such as “The”) or had the <syntax item> marked as *no proofing*. Some sentences or paragraphs were unacceptable to the grammar checker, but seemed OK and so were marked as *no proofing*. To avoid a large number of spelling and grammar errors the Word styles used for syntax, code and examples were changed to for *no proofing* (do not check spelling or grammar). The grammar checker (correctly) objects to clauses starting with “which” that are not preceded by a comma: in some cases the “which” was changed to “that” and in other cases a comma was inserted.

Extra spaces were removed at the following places

In section 1.5 in the list item (b) of the paragraph starting “Changes have been made” before “and merging of”.

In section 3.19 before the text “are defined that, when instantiated”.

In section 5.3.2 subsection item (c) *Semantics* before the text at the end of the second paragraph starting “An example of a property”.

In section 5.4.2 second paragraph list item (b) before the “)” character.

In section 8.3.1 *Concrete grammar* second list, item (b) before “represents the *Composite-state-type-identifier*”.

In section 8.4 *Concrete grammar* second paragraph after the syntax for <procedure reference area> in the centre of “<procedure name>.”

In section 8.5 *Concrete grammar* second paragraph after the syntax before the text “visibility shall not be used”.

In section 9 *Concrete grammar* item (d) in the list in the third paragraph after the syntax before the text “are specified, then the two”.

In section 10.1 *Concrete grammar* at the very end of the first paragraph after the syntax for <nondelaying channel symbol 2>.

In section 10.1 *Model* item (b) in the list for transform 3 before the text “and no explicit”.

In section 11.5 in the heading *Concrete grammar* before the word “*grammar*”.

In section 11.11.4 *Semantics* first paragraph before “*State-exit-point-name*”.

In section 11.13.1 *Concrete grammar* second paragraph after syntax before the second sentence starting “A <task area> containing”.

In section 11.13.5 *Semantics* first paragraph before the second sentence starting “The determination of whether the”.

In section 11.13.5 *Model* last paragraph in the text “<graphical answer>s .” before the full stop.

In section 12.1 *Model* fourth paragraph before the sentence starting “This anonymously named”.

In section 12.1.2 *Model* sixth paragraph before the text “contains in its <interface specialization>”.

In section 12.1.3 *Concrete grammar* first paragraph after syntax first sentence after the text “(or <interface specialization>“.

In section 12.3.1 *Semantics* list items (1) and (2) in both cases before the text “if the variable has”.

The word “which” was changed to “that” in the following places

In section 1.4 in the paragraph starting “In a few cases” in the text that was “words which are keywords of SDL-92”.”

In section 2 in the first paragraph in the text “provisions, which, through reference” and remove the comma after provisions.

In section 5.3.1 first paragraph list item (c) in the text “dynamic conditions which have to be fulfilled”.

In section 6.3 *Concrete grammar* in the paragraph starting “When the <name> part of an <identifier> denotes” in the text “an <expression> which is not part of any other <expression>”.

In section 10.5 *Model* Note 1 in the text “remote procedure calls which were left through”.

In section 11.3 *Concrete grammar* first sentence after the syntax for <asterisk input list> in the text “the *Input-node*s which this <input symbol> represents”.

In section 11.11.2 *Model* second paragraph in the text “signals in the complete valid input set of an agent which are”.

In section 11.12.2.2 first sentence in the text “the one which contains”.

In section 11.13.5 *Semantics* last paragraph in the text “to <answer>s which are <character string>s”.

In section 12 second paragraph in the text “operations which can be applied”.

In section 12.1.7 first sentence in the text “all data items which can”.

In section 12.2.2.*Concrete grammar* second paragraph after syntax in the text “defined *Literal-identifier* which satisfies”.

In Annex B first paragraph in the text “using tools which supported older versions”.

In Annex D section D.1 second paragraph first sentence in the text “Boolean sort which defines”.

In Annex D section D.2.2 first paragraph first sentence in the text “equations which only”.

In Annex D section D.2.4 *Semantics* in the text “of truth which holds”.

In Annex D section D.2.4 *Semantics* second paragraph in the text “string which contains”.

In Appendix II section II.1.1 list item (e) in the text “diagrams which could”.

A comma was inserted before “which” in the following places

Section 6.3 *Concrete grammar* in the list item (c) in the text “Consider only those elements in the product which do”

In section 8.1.1.1 *Model* last sentence in the text “Each implied type has a constraint which”.

In section 9 *Semantics* first paragraph in the text “An *Agent-definition* has a name which”.

In section 12.3 first paragraph in the text “the imperative expressions which”.

B.4.2 Textual corrections – MS Word format errors throughout the document (COM 17 TD3174 Jeju April 2006)

The outline level of various paragraphs was wrong causing problems for the Word document map, the automatic table of contents and also when generating a PDF file with bookmarks. The levels of the paragraphs have been made consistent. As Z.100 uses paragraph numbering up to 4 levels, the special headings for *Abstract grammar*, *Concrete grammar*, *Semantics*, and *Model* have been set to level 5. The first line of syntax entries has been set to level 6. These levels are part of the styles *z.100 heading, z.100 abs syntax 1st line, z100 syntax 1st line,* and *z100 syntax 1st line base*.

B.4.3 Textual corrections – MS Word file housekeeping (COM 17 TD3174 Jeju April 2006)

Unused styles were deleted from the document.

B.4.4 Textual corrections and minor clarifications throughout the document (COM 17 TD3174 Jeju April 2006)

The following clarifications and textual corrections were made (in some cases to avoid grammar errors)

Summary - Status/Stability - after “Compliance to this Recommendation (added 2003)” delete a full stop.

In section 1.2 in the paragraph starting “SDL is a rich language” delete the word “then” before “many aspects”.

In section 1.5 change “but further significant gains could be made by incorporating better support for this use in SDL.” to “but incorporating better support for this use in SDL could make further significant gains”

In section 1.5 in the paragraph starting “A small number of constructs” insert “and” before “macro diagrams” in the first sentence.

In section 6.1 at the end near section 6.2 change the text
“A <lexical unit> is terminated by the first character, which cannot be part of <lexical unit> according to the syntax specified above”
to
“The first character that cannot be part of the <lexical unit> according to the syntax specified above terminates the <lexical unit>”.

In section 6.9 in the first sentence change “any <diagram>” to “all kinds of <diagram>”.

In section 8.1.1.1 *Model* correct the spelling of “vitruality” and “vrituality” to “virtuality”.

In section 8.1.1.1 *Model* correct “a implied composite state type” to “an implied composite state type”.

In section 8.1.2 *Concrete grammar* first sentence after syntax change
“<actual context parameters> can be specified if and only if <base type> denotes a parameterized type.”
to
“It is valid to have <actual context parameters> if and only if <base type> denotes a parameterized type.”

In section 8.1.3.2 *Concrete grammar* the symbol for <dashed block symbol> was redrawn.

Section 8.1.5 *Concrete grammar* in the seventh paragraph after the syntax change
“the definition of block type”
to
“the definition of a block type”.

In section 8.4 *Concrete grammar* the symbols for <block type symbol> and <procedure symbol> were redrawn.

In section 9.5 *Model* fifth paragraph second sentence change “followed by a unlabelled” to “followed by an unlabelled”.

In section 10.1 *Model* third paragraph change “three transforms, that must be applied” to “three transforms, which must be applied”.

In section 10.5 *Concrete grammar* change the Word style of the syntax for <communication constraints> from “normal” to “z100 syntax last line”.

In section 11.11.2 *Semantics* third paragraph change “non deterministic” to “non-deterministic”.

In section 11.11.3 first sentence change “and represent connection points” to “and the state connection points represent connection points”.

In section 11.11.4 *Concrete grammar* second paragraph change “an *State-exit-point-name* corresponds” to “a *State-exit-point-name* corresponds”.

In the section headings for 11.12.2.1, 11.12.2.2, 11.12.2.3, 11.12.2.4 and 11.12.2.5 change the two tab characters after the heading by a single tab character.

In section 11.13.2 *Semantics* first paragraph first sentence insert a comma after the text “inside the agent that performs the create”.

In section 11.13.2 *Semantics* first paragraph last sentence make the word “offspring” bold.

In section 11.14.2 *Semantics* list item (f) sublist items (ii) and (iii) capitalize the initial word “if”.

In section 11.16.2 *Semantics* sixth paragraph change the text “trigger (e.g. input or handle)” to “trigger (an input or a handle)”

In section 11.16.2 *Semantics* subparagraph with the heading “a) *Whole agent/procedure/operation graph*” change “instance is in the stopping” to “instance enters a stopping”.

In section 11.16.2 *Semantics* subparagraph with the heading “b) *Start transition*” change “interprets a nextstate node or is in the stopping” to “interprets a *Nextstate-node* or enters a stopping”.

In section 11.16.2 *Semantics* subparagraph with the heading “e) *State*” change “a nextstate node or enters a stopping” to “a *Nextstate-node* or enters a stopping”.

In section 11.16.2 *Semantics* subparagraph with the heading “f) *Exception handler*” change “procedure enters a nextstate node or enters a stopping” to “procedure interprets a *Nextstate-node* or enters a stopping”.

In section 11.16.2 *Semantics* subparagraph with the heading “g) *Input*” change “procedure enters a nextstate node or enters a stopping” to “procedure interprets a *Nextstate-node*, or enters a stopping”.

In section 11.16.2 *Semantics* subparagraph with the heading “h) *Handle*” change
“The exception handler for the current Handle-node is activated whenever interpretation the Transition of Handle-node is started in the agent, operation or procedure. The exception handler is deactivated when a Nextstate-node is entered in the agent, operation or procedure”
to
“The exception handler for the current *Handle-node* is activated whenever interpretation the Transition of *Handle-node* is started in the agent, operation or procedure. The exception handler is deactivated when the agent, operation or procedure interprets a *Nextstate-node*”.

In section 11.16.2 *Semantics* NOTE change
“the input transition enters a nextstate node. Implicit states or stimuli are covered by exception handlers of the syntactical context”
to
“the input transition interprets a *Nextstate-node*. Exception handlers of the syntactical context cover implicit states or stimuli”.

In section 12.1.1 second paragraph change “these values are what is referenced” to “these values are referenced”.

In section 12.1.4 *Abstract grammar* first paragraph after the syntax change “The *Identifier* in a operator” to “The *Identifier* in an operator”.

In section 12.1.7.2 *Model* list item (a) delete a comma to change “an operator, Make, to create structures” to “an operator, Make to create structures”.

In section 12.1.8 *Semantics* last paragraph delete a full stop to change “description. (see clause 13)” to “description (see clause 13)”.

In section 12.1.9.6 *Concrete grammar* last paragraph delete a full stop to change “specification. (see 13)” to “specification (see 13)”.

In section 12.2.1 *Concrete grammar* add the missing “::=” to the first line of the <expression0> syntax.

In section 12.3.4.4 *Semantics* first paragraph first sentence change “Boolean sort which has the result true, if the timer” to “Boolean sort, which has the result true if the timer” (inserts a comma after “sort” and deletes a comma after “true”).

In Appendix III list item 1 last paragraph change “semantical” to the correct English word “semantic”.

B.4.5 Textual corrections – to avoid MS Word grammar errors (COM 17 TD3174 Jeju April 2006)

To avoid grammar errors (because the sentence starts with a lower case letter) the following changes were made

In section 6.2.2 insert “Each of the “ before the sentence starting “<macro actual parameter>s of a macro call must be”.

In section 6.2.2 insert “A “ before the sentence starting “<macro name> is visible in”.

In section 6.3 *Concrete grammar* insert “Any” at the start of the paragraph starting “<state name>s, <connector name>s, and <gate name>s”.

In section 6.6 in the third paragraph after the syntax insert “The” at the start of the second sentence starting “<page number area> is placed”.

In section 6.6 in the fourth paragraph after the syntax insert “The” at the start of the first sentence starting “<extra heading> must be shown”.

In section 6.6 in the fourth paragraph after the syntax insert “The” at the start of the second sentence starting “<heading> and <drawing kind>”.

In section 6.6 in the fourth paragraph after the syntax insert “The” at the start of the third sentence starting “<extra heading> is not defined further”.

In section 6.7 *Concrete grammar* in the first paragraph after the syntax for <comment> insert “An” at the start of the sentence starting “<end> in <package text area>”.

In section 6.9 insert “The” at the start of the first sentence starting “<text symbol> is used”.

In section 7.2 *Concrete grammar* last paragraph before *Semantics* insert “The keyword” at the start of the second sentence starting “**remote** is used”.

B.4.6 Clarification - 5.4.2 Metalanguage for the Concrete Grammar (COM 17 TD3174 Jeju April 2006)|

In the example <block reference> the syntax given was inconsistent with the actual syntax for <block reference>. The text:

block <block name> referenced <end>

is changed to

block <block name> [<number of instances>] **referenced** <end>

B.4.7 Clarification – 5.4.2 Metalanguage for the Concrete Grammar (COM 17 TD3108 Rev 1 Open 8 Geneva Oct 2005)

The change is to clarify the meaning of semantic subcategories. The paragraph starting “Sometimes the symbol includes an underlined part” is replaced by the two paragraphs:

Sometimes the symbol includes an underlined part. This underlined part defines semantic subcategories of the syntactic construct. For example, <block identifier> is syntactically identical to <identifier>, but semantically it requires the <identifier> shall identify a block. The meaning of <xxx yyy zzz> is that the item shall have the syntax <zzz> and shall denote an item that is an "xxx yyy" <zzz>, where "xxx yyy" is some defined category in the semantics of the language (such as a "state type"). In most cases this avoids having to describe constraints (such as the <expression> shall be a Boolean <expression>) in the concrete grammar, or the need to introduce a non-terminal such as <Boolean expression>. Where the further explanation is needed, this is added in the concrete grammar.

Most of the uses of semantic subcategories are for <identifier> and <name>. Subcategories of <name> are a special case, because they are used in two ways. In cases where <xxx name> occurs in a defining context, the subcategory means that the <name> defines the *Name* part of the *Identifier* of the entity belonging to that xxx class. Usually this is not stated explicitly because there is a correspondence between rules in the *Abstract grammar* and *Concrete grammar*.

Where <xxx name> occurs in a non-defining context, the <name> shall correspond to the *Name* part of an *Identifier* of an entity belonging to the xxx class. In this case <xxx name> is equivalent to <xxx identifier> but with the additional syntactic constraint that there shall be no <qualifier>.

B.4.8 Error correction - 8.1.1.1 Agent types, Abstract grammar (COM 17 TD3015 Moscow April 2005)

The static condition requiring a match between the parameters of an agent and its state machine were missing. Insert before the heading *Concrete grammar* the following paragraph:
The *Composite-state-type-definition* identified by the *Composite-state-type-identifier* of a *State-machine-definition* of an *Agent-type-definition* shall have a *Composite-state-formal-parameter* list that is the same as the *Agent-formal-parameter* list of the *Agent-type-definition*.

B.4.9 Clarification – 8.1.2 Type expression, Model (COM 17 TD3108 Rev 1 Open 2 Geneva Oct 2005)

The change clarifies how context parameters are inserted by describing the model. After "denoted by the identifier of <base type>." insert:
"The anonymous type definition is formed by:

1. Copying the definition of the <base type> in the context where the construct using the <type expression> occurs and changing the name to an anonymous unique name;
2. Replacing each occurrence of each <formal context parameter > name by the corresponding <actual context parameter> in the copy;
3. Removing the <formal context parameter> from the <formal context parameter list> and removing the <formal context parameters> if the <formal context parameter list> is empty;
4. Replacing the <type expression> by the anonymous unique name.

NOTE 1: Two textually identical <type expression>s with the <actual context parameters> denote different types, because they have different anonymous names. To use the same type binding in different places, a type definition using the <type expression> should used to name the <type expression>."

Because a Note has been added the subsequent Note is changed to "Note 2: ".

B.4.10 Clarification – 8.3.2 Virtual type, Concrete grammar; (COM 17 TD3108 Rev 1 Question 17 Geneva Oct 2005)

Add a new sentence to the end of the second paragraph after the syntax, to explain that there shall be a **virtual** type for every redefinition:
"Every redefined type shall be directly or indirectly (via another redefined type) a redefinition of a virtual type that is not redefined (that is with <virtuality> **virtual**)."

B.4.11 Clarification – 8.4 Type references (COM 17 TD3108 Rev 1 Open 11 Geneva Oct 2005)

In the first paragraph delete the sentence starting "A type reference specifies" and from the sentence starting "It is required" to the end of the paragraph. Then insert at the end of the paragraph "The type is fully described in the referenced definition or diagram, and this type has the attributes and properties partially specified as part of the type reference."

Replace the second sentence of the second paragraph by "If there are several references to the same type in a scope unit, this is the same as having one reference. If the referenced type is defined in the same scope unit as the reference, the type reference specifies that the type is defined in the scope unit of the containing definition or diagram."

After the heading *Concrete grammar* insert the following six paragraphs:
Each of the following is a type reference: <agent type reference>, <agent type reference area>, <composite state type reference>, < composite state type reference area>, <procedure reference>, <procedure reference area>, <signal reference>, <signal reference area>, <data type reference>, <data type reference area>, <interface reference>, < interface reference area> or <type reference area> defined below.

If in a type reference the <qualifier> of the <identifier> (or before the <name>) of the referenced type is omitted or identifies the scope unit directly enclosing the type reference, the type reference and <referenced definition> are in the same scope unit.
If in a type reference there is <qualifier> for the <identifier> or before the <name> of the referenced type and the <qualifier> does not identify the scope unit directly enclosing the type reference, the type reference and <referenced definition> are in different scope units. In this case the type reference can be removed from model without changing the semantics of the model and the reference is a form of annotation providing consistent information about the referenced definition.
If the referenced type is in the same scope unit as the reference, and the <referenced definition> is not syntactically contained within the scope unit, there shall be at least one type reference for the <referenced definition> as a proxy for the <referenced definition>. This enables the <referenced definition> to be located, so that the concrete diagrams and definitions can be mapped to a complete system model corresponding to the abstract grammar.
The partial specification as part of a type reference shall be consistent with the specification of the type definition or diagram (the <referenced definition>), that is it shall only contain information that is also specified in the <referenced definition>. In a partial specification of a variable, for example, if the variable name is given but not the sort of the variable, there shall be a definition for a variable with that name in the <referenced definition>. In any variable definition a sort is always defined.
If there is a non-empty <type preamble> in the type reference this shall be the same as the <type preamble> of the <referenced definition>.

B.4.12 Error correction - 8.4 Type references, Concrete grammar (COM 17 TD3174 Jeju April 2006)

The static conditions on <type preamble> were missing for <block type reference>, <process type reference>, <composite state type reference>, <signal reference> and <data type reference>, and similarly the static condition for <virtuality> of <interface reference>, and for **exported** for a <procedure reference>.

Insert the paragraph

"<type preamble> of a <block type reference> shall correspond to <type preamble> of the referenced <block type definition>. If the reference is virtual, the referenced block type shall be virtual. If the reference is abstract, the referenced block type shall be abstract."

before the paragraph

"A <type reference heading> that is part of a <block type reference> must have a <block type name>."

Insert the paragraph

"<type preamble> of a <process type reference> shall correspond to <type preamble> of the referenced <process type definition>. If the reference is virtual, the referenced process type shall be virtual. If the reference is abstract, the referenced process type shall be abstract."

before the paragraph

"A <type reference heading> that is part of a <process type reference> must have a <process type name>."

Insert the paragraph

"<type preamble> of a < composite state type reference> shall correspond to <type preamble> of the referenced < composite state type definition>. If the reference is virtual, the referenced composite state type shall be virtual. If the reference is abstract, the referenced composite state type shall be abstract."

before the paragraph

"A <type reference heading> that is part of a <composite state type reference> must have a <composite state type name>."

Insert the paragraph

"<type preamble> of a <procedure reference> shall correspond to <type preamble> of the referenced <procedure definition>. If the reference is virtual, the referenced procedure shall be virtual. If the reference is abstract, the referenced procedure shall be abstract. If exported is given in a <type reference heading>, the referenced type shall be an exported procedure and if a <remote procedure identifier> is also given, the procedure shall identify the same remote procedure definition.”

before the paragraph

“A <type reference heading> that is part of a <procedure reference> must have a <procedure name>.”

Insert the paragraph

“<type preamble> of a <signal reference> shall correspond to <type preamble> of the referenced <signal definition>. If the reference is virtual, the referenced signal shall be virtual. If the reference is abstract, the referenced signal shall be abstract.”

before the paragraph

“A <type reference heading> that is part of a <signal reference> must have a <signal name>.”

Insert the paragraph

“<type preamble> of a <data type reference> shall correspond to <type preamble> of the referenced <data type definition>. If the reference is virtual, the referenced data type shall be virtual. If the reference is abstract, the referenced data type shall be abstract.”

before the paragraph

“A <type reference heading> that is part of a <data type reference> must have an <data type name>.”

Insert the paragraph

“<virtuality> of a <interface reference> must correspond to <virtuality> of the referenced <interface definition>. If the reference is virtual, the referenced interface must be virtual.”

before the paragraph

“A <type reference heading> that is part of a <interface reference> must have an <interface name>.”

B.4.13 Error Correction - 8.4 Type references Concrete grammar (COM 17 TD3174 Jeju April 2006)

In the syntax rule <data type reference> replace “<data type identifier>” by
“<data type type reference heading>”. This seems to be a clerical error.

B.4.14 Error Correction - 8.4 Type references Concrete grammar (COM 17 TD3174 Jeju April 2006)

In the syntax rule <type reference kind symbol> the symbol should be a pair of “stereo type” characters. For some reason these appear as quotes, probably due to some font conversions.

On the second line of the syntax rule <type reference kind symbol> replace the quotes by

« »

and follow this by a new paragraph

“The <type reference kind symbol> has the appearance of a LEFT-POINTING DOUBLE ANGLE QUOTATION and RIGHT-POINTING DOUBLE ANGLE QUOTATION character pair on a single line with enough space between these two characters for the contained keyword.”

B.4.15 Clarification and Error Correction - 8.4 Type references Concrete grammar (COM 17 TD3174 Jeju April 2006)

The syntax rule <type reference properties> does not contain any properties, so the use of this rule is confusing. To clarify the syntax, replace “<type reference properties>” by “**referenced** <end>” in the syntax rules <system type reference>, <block type reference>, <process type reference>,

<composite state type reference>, <procedure reference>, <signal reference>, <data type reference> and <interface reference>, and delete the syntax rule:

“<type reference properties> ::=
referenced <end>”

This rule originally allowed attribute and behaviour properties to be specified, but in some cases it is very difficult (or even impossible) to parse the syntax. It was decided only to allow this item to be specified when a <class symbol> is used.

The text on the order of <syntax parameter property>s is an error, because these were not allowed in (the deleted) <type reference properties>. However, the order needs to be stated when a <class symbol> is used. Therefore, replace:
“The <signal parameter property>s in <type reference properties> must occur in the same order as the corresponding properties in the referenced type definition.”

by
“The <signal parameter property>s in <attribute properties area> shall occur in the same order (top to bottom as placed in the <class symbol>) as the corresponding <sort>s in the referenced <signal definition item>.”

B.4.16 Error correction – 8.4 Type references, Concrete grammar; (COM 17 TD3108 Rev 1 Question 8 Geneva Oct 2005)

The grammar for <behaviour property> made the syntax difficult to parse because the keywords were optional. The modification is to remove the “[“ and “]” around the keywords **operator**, **method**, **procedure**, **signal**, **exception**, **timer**.

B.4.17 Clarification – 8.4 Type references, Model (COM 17 TD3108 Rev 1 Open 11 Geneva Oct 2005)

Move the text of the first paragraph of the *Model* to the end of the fifth paragraph ending in “as defined in 7.3.” then delete the text “as defined in 7.3”.

B.4.18 Clarification – 9 Agents, Model; (COM 17 TD3108 Rev 1 Question 3 Geneva Oct 2005)

Delete the unnecessary paragraph:
“An agent that is a specialization is shorthand for defining an implicit agent type and one typebased agent of this type.”

B.4.19 Clarification – 9.5 Procedure, Semantics (COM 17 TD3174 Jeju April 2006)

Some comments have indicated that it is not obvious that an implicit local procedure is created inside an agent for procedure defined outside an agent, but the non-local items referenced from within the procedure are determined by the explicit definition outside the agent. To clarify before the paragraph starting “An external procedure is a procedure whose” add the following NOTE paragraph:

NOTE: The *Call-node* is always in the same enclosing agent as the *Procedure-definition*, because a subtype *Procedure-definition* is implicitly created locally if necessary (see 11.13.3). In any such subtype, identifiers of items (such as variables) external to the *Procedure-definition* are bound in the context of the super type *Procedure-definition* rather than the context of the *Call-node* if that is different.

B.4.20 Error correction - 10.2 Connection Concrete grammar (COM 17 TD3174 Jeju April 2006)

Delete the last paragraph “Each channel defined in the surrounding agent and which has its environment as one of its endpoints, must be mentioned in exactly one <external channel identifiers>.” There seems to be no reason for this restriction and it is therefore considered a technical error. Removing the restriction means that an external channel can be connected to any number of internal channels, regardless of whether the other endpoint of the external channel is connected to the environment of the surrounding agent or an agent within the surrounding agent.

B.4.21 Error Correction and Clarification - 10.3 Signal, Abstract grammar,(COM 17 TD3174 Jeju April 2006)

In the Abstract syntax the optional base type for specialisation is missing. Also it is not clearly defined whether the <sort list> given in <signal definition item> that has a <specialisation> should add to or replace the list of sorts of the base type. After the line

Sort-reference-identifier*

Add the line

[Signal-identifier *]*

After the line

Signal-name = Name

Add the paragraph

“The optional *Signal-identifier* of a *Signal-definition* identifies the base type (if any). The *Sort-reference-identifier* list is the list defined for the base type (if any) followed by the additional sorts defined for this signal type.”

B.4.22 Clarification - 10.3 Signal, Concrete grammar,(COM 17 TD3174 Jeju April 2006)

To clarify whether the <sort list> given in <signal definition item> that has a <specialisation> should add to or replace the list of sorts of the base type, add the paragraph
“Each <signal definition item> represents one Signal-definition. Each <sort> in the <sort list> adds a Sort-reference-identifier to the end of the Sort-reference-identifier list.”

before the paragraph
“An abstract signal can only be used in specialization and signal constraints.”

B.4.23 Error Correction - 10.5 Remote procedures, Model (COM 17 TD3174 Jeju April 2006)

Change the sentence “Both pCALL and pREPLY have a first parameter of the predefined Integer sort.” to “Both pCALL and pREPLY have a last parameter of the predefined Integer sort.” (that is “last” becomes “first”).

B.4.24 Clarification – 11.8 Implicit transition (COM 17 TD3015 Moscow April 2005)

The whole text of the section is written because it was considered unclear. The revised text is:
“Any signal not handled by an explicit input or save is consumed by an implicit transition (a transition of an implicit <input area> - see below) without a change of state.

Model
A <state area> has an implicit <input area> for <signal identifier>s contained in the complete valid input signal set of an <agent diagram> that are not (explicitly or via <asterisk input list> or <asterisk save list>) in the set of contained in the <input list>s, <priority input list>s and the <save list> of the <state area>.
This implicit <input area> has a <transition area> that only contains a <nextstate area> leading back to the same <state area>.”

B.4.25 Error Correction – 11.11.2 State aggregation, Concrete grammar (COM 17 TD3174 Jeju April 2006)

There is a missing "}" in the syntax rule <state aggregation area>. Add "}" after <composite state structure area>.

B.4.26 Clarification – 11.11.3 State connection point; (COM 17 TD3108 Rev 1 Question 17 Geneva Oct 2005)

Delete the heading *Semantics* because the text really expresses what the concrete syntax means.

B.4.27 Clarification – 11.11.3 State connection point, Concrete grammar; (COM 17 TD3108 Rev 1 Question 11 Geneva Oct 2005)

A note is added before the *Semantics* heading to explain the rationale behind the brackets in the syntax:

NOTE: The reason for the brackets around the state entry or exit points is to make it easy to see the beginning and end of the list. In the case of a single point the brackets can be omitted because there must be at least one point and this should be easy to find.

B.4.28 Modification - 11.12.2.1 Nextstate - (see COM 10-TD 41 item 8 (Geneva, 21-24 November 2000))

This change is necessary to correct a deficiency in the handling of HISTORY.

The *Abstract grammar* is changed from

Nextstate-node	::	*State-name*
		[*Nextstate-parameters*]
Nextstate-parameters	::	[*Expression*]*
		[State-entry-point-name]
		[**HISTORY**]

to

Nextstate-node	=	*Dash-nextstate* | *Named-nextstate*
Named-nextstate	::	*State-name*
		[*Nextstate-parameters*]
Nextstate-parameters	::	[*Expression*]*
		[*State-entry-point-name*]
Dash-nextstate	::	[**HISTORY**]

The paragraph in the *Semantics*
"A dash nextstate for a composite state implies that the next state is the composite state."

is replaced by the three paragraphs
"An empty *Dash-nextstate* means that the state is entered again. An empty *Dash-nextstate* for a composite state implies that the next state is the composite state.

NOTE – if there is only one state that can lead to the *Dash-nextstate*, the *Dash-nextstate* has the same meaning as a *Nextstate-node* that has the *State-name* of this state.

When a *Dash-nextstate* with HISTORY is interpreted, the next state is the one in which the current transition was activated. If interpretation re-enters a composite state, its entry procedure is invoked."

The paragraph in the *Semantics*
"When a Nextstate-node with HISTORY is interpreted, the next state is the one in which the current transition was activated. If interpretation re-enters a composite state, its entry procedure is invoked."

The *Model* subsection is deleted.

B.4.29 Error correction – 11.12.2.5 Raise, Model; (COM 17 TD3108 Rev 1 Question 2 Geneva Oct 2005)

There should not be a model section here. Replace the *Model* section (including the heading) by the Note:

NOTE: A <raise area> of a remote procedure call is according to the model (of remote procedure calls. The model for transition terminators in 11.12.1 applies after this transformation.

B.4.30 Extension – 11.13.1 Task, Concrete grammar (COM 17 TD3091, TD3092, TD3119 October 2005)

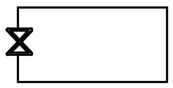
To add symbols for starting and stopping times, the syntax for <task area> is extended. Replace:
“ <task symbol> ***contains*** <task body>”

By
“ { <task symbol> ***contains*** <task body> | <start timer area> | <stop timer area> }”

Before the paragraph “The trailing <end> in <statement list> of a <task body> may be omitted.” add the syntax

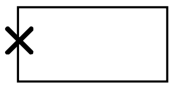
<start timer area> ::=
<start timer symbol> contains <set body>

<start timer symbol> ::=



<stop timer area> ::=
<stop timer symbol> contains <reset body>

<stop timer symbol> ::=



B.4.31 Clarification – 11.13.1 Task, Concrete grammar (COM 17 TD3174 Jeju April 2006)

Delete the sentence “A <task area> containing a single <statement> in its <statement list> represents whatever that statement represented.” because it is little confusing and is not strictly necessary because a *Compound-node* with a single statement gives the right semantics. However, to make this clear, after the paragraph where the sentence is deleted, add the following NOTE paragraph:

“NOTE: If the <statement list> contains a single <statement>, in the *Compound-node* the *Variable-definition-**set*** is empty, there is no *Exception-handler-node,* and the *Init-graph-node* and *Step-graph-node* lists are empty. The *Transition* of the *Compound-node* consists of either the *Terminator* for the statement (for example for a <stop statement>) or *Graph-node* for the <statement> (typically *Task-node* that is an *Assignment*) followed by a *Break-node* with the *Connector-name* for the *Compound-node*. This is therefore equivalent to replacing the *Compound-node* by the *Terminator* or *Graph-node* for the <statement>.”

B.4.32 Extension – 11.13.1 Task, Model (COM 17 TD3091, TD3092, TD3119 October 2005)

To add symbols for starting and stopping timers, the *Model* section is extended to include the new symbols. At the end of the *Model* section insert a new paragraph:
“A <start timer area> is transformed into a <task area> defined by a <task symbol> containing a <set statement> with the same <set body> as the <start timer area>. A <stop timer area> is transformed into a <task area> defined by a <task symbol> containing a <reset statement> with the same <reset body> as the <stop timer area>.”

B.4.33 Error Correction – 11.14 Statement list, Concrete grammar (COM 17 TD3174 Jeju April 2006)

There is a missing “|” in the syntax rule <terminating statement>. Add “|” before <stop statement>.

B.4.34 Modification - 11.13.2 Create - (see COM 10-TD 10 item 7; COM 10-TD 41 item 5 (Geneva, 21-24 November 2000))

This change is necessary to correct a deficiency in the handling of the "**this**" construct because it cannot be transformed and to remove the unnecessary *Variable-identifier*.

The *Abstract grammar* is changed from

Create-request-node	::	[*Variable-identifier*] *Agent-identifier* [*Expression*]*

to

Create-request-node	::	{ *Agent-identifier* | **THIS** } [*Expression*]*

At the end of the *Semantics* subsection insert the paragraph:
"**THIS** identifies the *Agent-identifier* (which may be an anonymous implied identifier) of the set of instances of the agent in which the create is being interpreted."

At the end of the *Model* delete the first paragraph:
"Stating this is derived syntax for the implicit <process identifier> that identifies the set of instances of the agent in which the create is being interpreted."

B.4.35 Modification - 11.13.3 Procedure call - (see COM 10-TD 41 (Geneva, 21-24 November 2000) items 9, 10)

This change is necessary to correct a deficiency in the handling of the "**this**" construct because it cannot be transformed.

The *Abstract grammar* is changed from

Call-node	::	*Procedure-identifier* [*Expression*]*
Value-returning-call-node	::	*Procedure-identifier* [*Expression*]*

to

Call-node	::	[**THIS**] *Procedure-identifier* [*Expression*]*
Value-returning-call-node	::	[**THIS**] *Procedure-identifier* [*Expression*]*

At the end of the *Semantics* subsection insert the paragraph:
"If **THIS** is present and the procedure is specialized, the *Procedure-identifier* refers to the identifier of the specialized procedure. For a procedure that is not specialized or if **THIS** is absent, the *Procedure-identifier* refers to the identifier of the procedure that is not specialized."

At the end of the *Model* delete the paragraph:
"**this** implies that when the procedure is specialized, the <procedure identifier> is replaced by the identifier of the specialized procedure."

B.4.36 Modification - 11.13.4 Output - (see COM 10-TD 41 (Geneva, 21-24 November 2000) items 9, 10)

This change is necessary to correct a deficiency in the handling of the "**this**" construct because it cannot be transformed.

The *Abstract grammar* rule *Signal-destination* is changed from

Signal-destination	=	*Expression* | *Agent-identifier*

to

Signal-destination	=	*Expression* | *Agent-identifier* | **THIS**

After the first paragraph of the *Semantics* subsection insert the paragraph:

"Stating **THIS** in a *Signal-destination* refers to the set of instances of the agent in which the output is being interpreted."

At the end of the *Model* delete the paragraph:

"Stating **this** in <destination> is derived syntax for the implicit <agent identifier> that identifies the set of instances for the agent in which the output is being interpreted."

B.4.37 Modification – 12.1 Data definition, Abstract grammar (COM 17 TD3015 Moscow April 2005)

It needed to be defined where implicit procedure definitions for operations are placed. When the operation is in a data type defined within another data type, the procedure needs to be defined outside the enclosed data type but within the enclosing data type. The change is to allow implicit procedures in the abstract grammar at this point.

In the abstract syntax for both *Value-data-type-definition* and *Object-data-type-definition* insert after "*Dynamic-operation-signature-**set**"* insert the line (twice):

*Procedure-definition-**set***

B.4.38 Error correction - 12.1 Data definitions, Abstract grammar (COM 17 TD3174 Jeju April 2006)

Delete the alternative "*Data-type-definition-**set***" and replace the alternative "*Syntype-definition-**set***" by "*Signal-definition-**set***", to make the abstract syntax here consistent with Annex F (which is correct).

B.4.39 Modification – 12.1 Data definition, Abstract grammar (COM 17 TD3015 Moscow April 2005)

It needed to be defined where implicit procedure definitions for operations are placed. For the abstract syntax change adding *Procedure-definition-**set**,* at the end of the *Abstract* grammar just before the heading *Concrete grammar* insert the paragraph:

Each *Procedure-definition* of the *Procedure-definition-**set*** of a *Value-data-type-definition* or *Object-data-type-definition* is a *Procedure-definition* associated with an *Operation-signature* according to the Model in 12.1.8, Behaviour of operations for an enclosed *Data-type-definition*. It is not allowed to define procedures directly within a data type definition.

B.4.40 Error correction - 12.1 Data definition, Semantics (COM 17 TD3174 Jeju April 2006)

Correct "*Expanded-sort-identifier*" to "*Expanded-sort*". Correct "*Referenced-sort-identifier*" to "*Referenced-sort*".

B.4.41 Modification - 12.1.1 Data type definitions, Concrete grammar (COM 17 TD3015 Moscow April 2005)

It needed to be defined where implicit procedure definitions for operations are placed. When the operation is in a data type defined within another data type, the procedure needs to be defined outside the enclosed data type but within the enclosing data type. The change is to allow implicit procedures in the abstract grammar at this point.

In the concrete syntax in the rule <entity in data type> insert after <synonym definition> the 2 lines:

```
        |       <procedure definition>
        |       <procedure reference>
```

At the end of the *Concrete grammar* just before the heading *Semantics* insert the paragraph:
A <procedure definition> or <procedure reference> is only allowed as an <entity in data type> if the <procedure definition> or <procedure reference> is the result of transforming an <operation definition> or <operation diagram> of an enclosed <data type definition> as described in 12.1.8, Behaviour of operations.

B.4.42 Clarification – 12.1.3 Specialization of data types; (COM 17 TD3108 Rev 1 Agreed 3 Geneva Oct 2005)

Delete from the beginning of 12.1.3
"The sort defined by the specialization is considered a subsort of the sort defined by the base type. The sort defined by the base type is a supersort of the sort defined by the specialization."

Insert a new paragraph after the heading *Semantics*
"The sort defined by the specialization is a subsort of the sort defined by the base type. The sort defined by the base type is a supersort of the sort defined by the specialization."

B.4.43 Clarification – 12.1.3 Specialization of data type, Semantics (COM 17 TD3108 Rev 1 Data 2 Geneva Oct 2005)

The changes clarifies that **value** X is considered as "defined by an object type" if X is defined by "**object type** X ...". The required change is to replace the text "T has been defined by an object type" in item (c) of first list in the *Semantics* by "T is denoted by a <sort identifier> or **object** <sort identifier> or **value** <sort identifier>, and the <sort identifier> is defined by an object type".

B.4.44 Clarification – 12.1.4 Operations, Abstract grammar; (COM 17 TD3108 Rev 1 Agreed 2 Geneva Oct 2005)

Insert a new second paragraph in the *Abstract grammar*:
A Virtual-argument shall reference the Sort of an Object-data-type-definition.

B.4.45 Clarification – 12.1.5 Any, Semantics (COM 17 TD3108 Rev 1 Data 9 Geneva Oct 2005)

The change clarifies the semantics of *is_equal*.

Replace the sentence starting "In general for the *is_equal*" by the sentence "If the sort of the actual parameter is not sort compatible to the sort of **this**, or **this** or the actual parameter is Null, the exception InvalidReference is raised."

In the item (a) for *is_equal* replace "are the same value" by "have the same value for a value data type, or reference the same value for an object data type".

B.4.46 Clarification – 12.1.5 Any, Semantics (COM 17 TD3108 Rev 1 Data 3 Geneva Oct 2005)

The change clarifies the semantics of *Is_equal* for structures with absent fields. Add to the end of the item (b) for *is_equal* the sentence "The method *is_equal* is able to return the predefined Boolean value true only if this field is absent in corresponding components of both Y and X, otherwise the predefined Boolean value false is returned."

B.4.47 Clarification – 12.1.5 Any, Semantics (COM 17 TD3108 Rev 1 Data 13 Geneva Oct 2005)

The change clarifies the semantics of *copy*. To the end of the paragraph on the *copy* method add the sentence "The method *copy* returns **this.**"

B.4.48 Clarification – 12.1.5 Any, Semantics; (COM 17 TD3108 Rev 1 Agreed 1 Geneva Oct 2005)

Remove item (b) for *is_equal*. This condition is not needed, because the dynamic method lookup will pick the appropriate *is_equal*. Delete the text "if the sort of X is an object sort, let X be the result of interpreting the *Make* operator for the data type that defined the sort of Y. The result is obtained by interpreting *x.copy*(*y*), where *x* and *y* represent X and Y, respectively; otherwise" and make item (c) item (b).

B.4.49 Clarification – 12.1.7.2 Structure data type, Model; (COM 17 TD3108 Rev 1 Data 14 Geneva Oct 2005)

Replace
"**virtual** *field-modify-operation-name* (<field sort>) -> S;"

by
"**virtual** *field-modify-operation-name* (<field sort>) -> **this** S;"

B.4.50 Clarification – 12.1.7.3 Choice data type, Model; (COM 17 TD3108 Rev 1 Data 14 Geneva Oct 2005)

In item (d) of the list, add after
"Otherwise the <data type definition> is a <value type definition>"

the text
"of the same form but with the keyword **value** and **endvalue** (instead of **object** and **endobject)**"

B.4.51 Clarification – 12.1.8 Behaviour of operations, Concrete syntax (COM 17 TD3108 Rev 1 Open 6 Geneva Oct 2005)

The reason for the positioning of <operator preamble> in the syntax, is explained by placing before the rule <operation identifier> the Note paragraph:

NOTE: <operation preamble> is placed after the keyword **operator** or **method** to avoid ambiguity with the optional <operation signatures> which can also start with an <operation preamble>. The initial keyword of <operation definitions> is never the same as the initial keyword of an <operation signature>.

B.4.52 Modification - 12.1.8 Behaviour of operations, Model (COM 17 TD3015 Moscow April 2005)

It needed to be defined where implicit procedure definitions for operations are placed. Replace the paragraph starting "An <operation definition> or <operation diagram> is transformed" by
An <operation definition> or <operation diagram> is transformed into a <procedure definition> or <procedure diagram> (with a <procedure reference> to the diagram) respectively in the enclosing context of the directly

enclosing <data definition> for the data type. The <procedure definition> or <procedure diagram> has a <procedure name> derived from the <operation name>, <procedure formal parameters> derived from the <formal operation parameters>, and a <result> derived from the <operation result>. In the case of an <operation diagram>, the <procedure body area> is derived from the <operation body area>. The <procedure name> derived from the <operation name> is not known directly in the concrete grammar, therefore the procedure can only called by the invocation of the operation. When an <operation definition> is transformed, the <operation definition> is removed and the <procedure definition> is placed after the <data definition>. When an <operation diagram> is transformed, the <procedure diagram> replaces the <operation diagram>. A <procedure reference> that references the <procedure diagram> is placed after the <data definition>.

B.4.53 Error correction – 12.1.9.1 Name class, Model (COM 17 TD3108 Rev 1 Open 12 Geneva Oct 2005)

Alphabetical sorting does not always work, so the sorting is changed as follows.

In the last paragraph replace "in the alphabetical" by "first by length (a shorter literal comes before any longer ones) then in lexicographical" and delete ", and a true prefix of a word is considered less than the whole word".

B.4.54 Clarification – 12.2.1 Expressions; (COM 17 TD3108 Rev 1 Question 53 Geneva Oct 2005)

Move the interpretation of constant expressions from the *Concrete grammar* to the semantics. Delete from the *Concrete grammar*:
"Each <constant expression> is interpreted once during initialization of the system, and the result of the interpretation is preserved. Whenever the value of the <constant expression> is needed during interpretation, a complete copy of that computed value is used."

In the *Semantics* replace "A simple expression is a *Constant-expression*." (which is not needed) by:
"Each *Constant-expression* is interpreted once during initialization of the system, and the result of the interpretation is preserved. Whenever the value of the *Constant-expression* is needed during interpretation, a complete replicate of that computed value is used."

B.4.55 Clarification – 12.2.5 Equality expression, Semantics; (COM 17 TD3108 Rev 1 Agreed 1 Geneva Oct 2005)

There is no need to reorder the parameters for value equality, so the text can be simplified by replace the text from "denotes equality of values" to "<operand2>)" by:
If, after interpretation, either one of the operands is a value, the *Equality-expression* denotes equality of values and the <equality expression> returns the result of the application of the equal operator to *First-operand* and *Second-operand*, where equal is *Operation-identifier* represented by the <operation identifier> in the <operation application>:
    equal(<operand2>, <operand3>)

Model

B.4.56 Error correction – 12.3.4 Imperative expressions, Abstract grammar (COM 17 TD3108 Rev 1 Open 5 Geneva Oct 2005)

As a new last alternative for *Imperative-expression* add the missing item:

| *State-expression*

B.4.57 Extension – 12.3.4 Imperative expressions, Abstract grammar (COM 17 TD3108 Rev 1 Open 10 Geneva Oct 2005)

As a new alternative for *Imperative-expression* after *Timer-active-expression* add the new item:

| *Timer-remaining-duration*

B.4.58 Extension – 12.3.4 Imperative expressions, Concrete grammar (COM 17 TD3108 Rev 1 Open 10 Geneva Oct 2005)

As a new alternative for <imperative expression> after <timer active expression>add the new item:

```
                        |       <timer remaining expression>
```

B.4.59 Extension – 12.3.4.4 Timer active expression (COM 17 TD3108 Rev 1 Open 10 Geneva Oct 2005)

Extend the section title to "**Timer active expression and timer remaining duration**".

Add to the *Abstract grammar* the new rule:

Timer-remaining-duration	::	*Timer-identifier*
		*Expression**

In the sentence following the rules change "must" to "or *Timer-remaining-duration* shall".

In the *Concrete grammar* add the new rule:

<timer remaining duration> ::=
rem (<timer identifier> [**(** <expression list> **)**] **)**

Replace the whole of the *Semantics* (except the heading *Semantics*) by:

"The timer of a *Timer-active-expression* or *Timer-remaining-duration* is the timer identified by *Timer-identifier* and set with the same results as denoted by the *Expression* list (if any). The expressions are interpreted in the order given.

If a sort specified in a timer definition is a syntype, then the range check defined in 12.1.9.5 applied to the corresponding expression in <expression list> must be the predefined Boolean value true; otherwise, the predefined exception OutOfRange (see D.3.16) is raised.

A *Timer-active-expression* is an expression of the predefined Boolean sort, which has the result true, if the timer active (see 11.15). Otherwise, the *Timer-active-expression* has the result false.

A *Timer-remaining-duration* is an expression of the predefined Duration sort. The result value is the time the timer was last set to minus **now**. If the timer has never been set the value is undefined and the exception UndefinedVariable is raised."

B.4.60 Error correction – 12.3.4.6 State expression, Semantics (COM 17 TD3108 Rev 1 Open 5 Geneva Oct 2005)

The semantics are changed to reflect the fact the value is dynamic rather than static. The *Semantics* are changed to:

"A state expression interpreted as a Charstring that contains the spelling of the name of the most recently entered state of the nearest enclosing scope unit. If there is no such state, <state expression> denotes the empty string (")."

B.4.61 Error correction – 13 Generic system definition; (COM 17 TD3108 Rev 1 Question 1 Geneva Oct 2005)

There should not be a *Semantics* section here. In both 13.1 and 13.2 delete the *Model* heading and change the *Semantics* heading to *Model*.

B.4.62 Extension – Annex B Backwards compatibility (COM 17 TD3174 Jeju April 2006)

Add a new section "B.11 Task" with the following contents

Concrete grammar

The rule <task body> is extended to allow assignments to be separated as commas as in SDL-92.

```
<task body> ::=
                                <statement list>
                        |       <informal text>
                        |       <legacy task body>
```

```
<legacy task body> ::=
                        |       <assignment> { , <assignment> }*
```

B.5 Z.104 changes

There are no agreed changes to the Z.104 (07/05).

B.6 Z.105 changes

There are no agreed changes to the Z.105 (07/03).

B.7 Z.106 changes

The following are changes to Z.106 (08/02):

B.7.1 Extension - A75 extended task symbol (see COM 17-TD 3202 and COM 17-TD 3225 (Geneva, 10-19 March 2004))

This change is to validate some existing CIF produced using an implementation of CIF prior to agreement of Z.106.

Add to the end of section 7.4.19 A19 CIF descriptor the alternative:

"| <extended task symbol: A75>"

Add to section 7.4.49 just before the example the note:

"NOTE <extended task symbol: A79> can be used as an alternative syntax."

After section 7.4.74 insert as a new section

"7.4.75 A75 extended task symbol:

/* CIF Extendedtask <position and size: B20> ***/**
[<text position: B21>]
<left curly bracket>
<statement list>
<right curly bracket>
<end>

Additional information:

See also <task symbol: A49> that describes an alternate syntax to represent task symbols.

Example:

```
/* CIF Task (800,550) */
TASK myVariable := 0;"
```

B.8 Z.107 changes

There are no agreed changes to Z.107 (11/99)

B.9 Z.108 status

Z.108 is no longer an open study.

B.10 Z.109 changes

The following changes are agreed to Z.109 (11/99).

B.10.1 Clarification - clause 3.2.8.2 Procedure

Add the following statement as the last statement of the introduction:

“A Procedure-definition is thereby represented both by an Operation in the class representing the type with the procedure and by a Class as part of the namespace of this class.”

Add the following statement right after the table:

“A procedure that is neither virtual, redefined nor finalized, will in its Class not have any of these tagged values and the Operation will have “isLeaf=true”.”

B.10.2 Clerical Error – clause 3.2.27 Operation

Change the following entry of the table

.isLeaf = true	• **finalized**

to

.isLeaf = true	• **finalized** or non-virtual

B.10.3 Clerical Error – clause 3.2.8.3 Data types

Replace *Static-operator* with *Static-operation-signature*

Replace *Dynamic-operator* with *Dynamic-operation-signature*

B.10.4 Clarifications – Appendix III

Delete “the” before “one of the duplicate transitions”.

Replace <textual procedure definition> with <procedure definition>.

B.10.5 Clerical Error – clause 3.2.21 Generalization

Replace *Parent-type-identifier* with *Type-identifier*

B.10.6 Clerical Error – clause 3.2.23 Method

Replace *Static-operator* with *Static-operation-signature*

Replace *Dynamic-operator* with *Dynamic-operation-signature*

B.10.7 Clerical Error – clause 3.2.27 Operation

Replace *Static-operator* with *Static-operation-signature*

Replace *Dynamic-operator* with *Dynamic-operation-signature*

B.10.8 Clerical Error – clause 3.5.2 Signal

Replace *Data-type-reference-identifier* with *Sort-reference-identifier*

B.10.9 Clerical Error – clause 3.6.2 AssociationRole

Replace *Channel-names* with *Channel-name*

B.10.10 Clerical Error – clause 3.8.33.8.3 CompositeState

Composite-state-part shall be changed according to eventual change at the meeting.

Replace *Multi-state* with *State-aggregation-node*

B.10.11 Clerical Error – clause 3.8.9 SimpleState

Composite-state-part shall be changed according to eventual change at the meeting.

B.10.12 Clerical Error – clause 3.8.10 State

Composite-state-part shall be changed according to eventual change at the meeting.

B.11 Open issues

The following is a list of issues classified as open items according to the rules for maintenance for SDL. The list below applied to the Recommendations Z.100, Z.105 and Z.107 approved by Study Group 10 on 19th November 1999. Please note that this list is overdue for revision.

To facilitate the tracking of open items each item is given an identifier of the form (month/year).<number> where month and year identify the meeting at which the item was first put onto the list. For example "(04/97).3".

To keep the list concise, the details given in relevant documents are not copied to the list, but the documents are referenced. However, a consequence of this procedure is that some references are to temporary documents of meetings, and therefore the Question rapporteur maintains copies of all referenced documents.

B.11.1 Additional issues to be considered

Open issues for which no contributions are available:

- Removal of nested diagrams or restructuring Z.100 to make nested diagrams auxiliary features.
- Further restructuring of the specification to separate a core language from extension features.
- CORBA/IDL language mapping (binding) or binding to interface definition languages of other component models
- Data type library extensions (predefined object types, Standard Template Library analogue)
- Memory management issues
- Signals and exceptions as data
- Instance sets vs. container types and navigation into composite agents
- Broadcast mechanisms
- Better handling of priorities (e.g., multiple levels of priorities on inputs, process priorities)
- Multiple queues and "input via" (determine the gate at which a signal arrived)
- Interrupts
- Multi-party interfaces

- Integration with external action languages / further harmonization of action language with well-known external languages.

B.11.2 List of Open Items

The following is a list of issues classified as open items according to the rules for maintenance for SDL. The list below applies to the Recommendations Z.100, Z.105 and Z.107 as approved by Study Group 10 on 19th November 1999.

To facilitate the tracking of open items each item is given an identifier of the form (month/year).<number> where month and year identify the meeting at which the item was first put onto the list. For example "(04/97).3".

To keep the list concise, the details given in relevant documents are not copied to the list, but the documents are referenced. However, a consequence of this procedure is that some references are to temporary documents of meetings, and therefore all referenced documents will be available on the ITU FTP server
(<ftp://username:password@ties.itu.int/u/tsg17/sg17/archive/2001_xchange/wp3/q13/Retained_documents/>)
and on the SDL Forum ftp server at
<ftp:// username:password@ftp.sdl-forum.org/sg17/wp3/q13/Retained_documents/>.

(10/96).2	Extended alphabet for SDL.(C0M-10-1)
(04/97).13	output -, input *, reset * (TD 34, TD 37 SG 10 April 1997, note that this may also be related to signal as data);
(04/97).14	simplified timer handling timer on a state (TD 34 SG 10 April 1997); time supervised state
(04/97).17	signal parameter access (TD 34, TD 35 SG 10 April 1997, e.g., refer to signal parameters in enabling conditions);
(04/97).25	more flexible USE syntax – USE p1, p2, p3 (TD 35 SG 10 April 1997);
(03/98).1	generation of variants (TD 86 SG 10 March 1998);
(10/98).1	Simpler initialization of systems (TD 34 SG 10 April 1997, TDS 603 October 1998,TDN 631 November 1998) and dynamic routing;
(11/99).1	SuperType Method call (R 10 of the study period 1997-2000, SG 10, Annex 10.6, 2.2.13);
(11/99).2	Terminology for diagram or drawing in SDL (R10 of the study period 1997-2000, SG10, Annex 10.6, 2.1.1).
(11/99).3	UTF8String in Z.105 (TD 68 November 1999).
(06/00).1	Exit connection points for tasks (TDA03 6.6 June 2000)

Some text has been put into Z.100 for extended character sets in comments, but (10/96).2 has been left on the list of open items so that it is considered if this is sufficient, e.g., allow extended character sets in identifiers. Note that ASN.1 does not allow non-Latin character sets for identifiers.

B.11.3 List of Closed items

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.

(10/96).9 Allow algorithmic operators with external data - requirement not clear (COM-10-1);

(10/96).13 Operators returning sets of values (multivalued operators) - can adequately be handled with STRUCT (COM-10-1);

(04/97).18 signal Priority (TD 34 SG 10 April 1997);

(04/97).30 Relaxation of the rules for signals to services, because this would break the model for services and context parameters. (TD 35 SG10 April 1997);

NOTE: services are replaced by state aggregation, so it needs to be considered if similar rules should apply.

(04/97).31 **virtual** as default, because this had been extensively discussed (and rejected) when SDL-92 was formulated and has implications on the use of constraints. (TD 37 SG 10 April 1997);

(10/98).2 remote process creation, because this was added to the language, but the need can be satisfied by remote procedure and the state machine of a block and it was decided an additional construct made the language too complex (TDI 608 Internet meeting Autumn 1998);

____