

ITU-T

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

Z.100

Annex F1
(10/2016)

SERIES Z: LANGUAGES AND GENERAL SOFTWARE
ASPECTS FOR TELECOMMUNICATION SYSTEMS

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

Specification and Description Language – Overview
of SDL-2010

**Annex F1: SDL-2010 formal definition: General
overview**

Recommendation ITU-T Z.100 – Annex F1

ITU-T

ITU-T Z-SERIES RECOMMENDATIONS

LANGUAGES AND GENERAL SOFTWARE ASPECTS FOR TELECOMMUNICATION SYSTEMS

FORMAL DESCRIPTION TECHNIQUES (FDT)	
Specification and Description Language (SDL)	Z.100–Z.109
Application of formal description techniques	Z.110–Z.119
Message Sequence Chart (MSC)	Z.120–Z.129
User Requirements Notation (URN)	Z.150–Z.159
Testing and Test Control Notation (TTCN)	Z.160–Z.179
PROGRAMMING LANGUAGES	
CHILL: The ITU-T high level language	Z.200–Z.209
MAN-MACHINE LANGUAGE	
General principles	Z.300–Z.309
Basic syntax and dialogue procedures	Z.310–Z.319
Extended MML for visual display terminals	Z.320–Z.329
Specification of the man-machine interface	Z.330–Z.349
Data-oriented human-machine interfaces	Z.350–Z.359
Human-machine interfaces for the management of telecommunications networks	Z.360–Z.379
QUALITY	
Quality of telecommunication software	Z.400–Z.409
Quality aspects of protocol-related Recommendations	Z.450–Z.459
METHODS	
Methods for validation and testing	Z.500–Z.519
MIDDLEWARE	
Processing environment architectures	Z.600–Z.609

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

Recommendation ITU-T Z.100

Specification and Description Language – Overview of SDL-2010

Annex F1

SDL-2010 formal definition: General overview

Summary

Annex F1 provides the motivation for and the main objectives of a formal semantics definition for SDL-2010. It gives an overview of the structure of the formal semantics, and it also contains an introduction to the Abstract State Machine (ASM) formalism, which is used to define the SDL-2010 semantics.

History

Edition	Recommendation	Approval	Study Group	Unique ID*
1.0	ITU-T Z.100	1984-10-19		11.1002/1000/2222
1.1	ITU-T Z.100 Annex A	1984-10-19		11.1002/1000/6664
1.2	ITU-T Z.100 Annex B	1984-10-19		11.1002/1000/6665
1.3	ITU-T Z.100 Annex C1	1984-10-19		11.1002/1000/6666
1.4	ITU-T Z.100 Annex C2	1984-10-19		11.1002/1000/6667
1.5	ITU-T Z.100 Annex D	1984-10-19		11.1002/1000/6668
2.0	ITU-T Z.100	1987-09-30	X	11.1002/1000/10954
2.1	ITU-T Z.100 Annex A	1988-11-25		11.1002/1000/6669
2.2	ITU-T Z.100 Annex B	1988-11-25		11.1002/1000/6670
2.3	ITU-T Z.100 Annex C1	1988-11-25		11.1002/1000/6671
2.4	ITU-T Z.100 Annex C2	1988-11-25		11.1002/1000/6672
2.5	ITU-T Z.100 Annex D	1988-11-25	X	11.1002/1000/3646
2.6	ITU-T Z.100 Annex E	1988-11-25		11.1002/1000/6673
2.7	ITU-T Z.100 Annex F1	1988-11-25	X	11.1002/1000/3647
2.8	ITU-T Z.100 Annex F2	1988-11-25	X	11.1002/1000/3648
2.9	ITU-T Z.100 Annex F3	1988-11-25	X	11.1002/1000/3649
3.0	ITU-T Z.100	1988-11-25		11.1002/1000/3153
3.1	ITU-T Z.100 Annex C	1993-03-12	X	11.1002/1000/3155
3.2	ITU-T Z.100 Annex D	1993-03-12	X	11.1002/1000/3156
3.3	ITU-T Z.100 Annex F1	1993-03-12	X	11.1002/1000/3157
3.4	ITU-T Z.100 Annex F2	1993-03-12	X	11.1002/1000/3158
3.5	ITU-T Z.100 Annex F3	1993-03-12	X	11.1002/1000/3159

* To access the Recommendation, type the URL <http://handle.itu.int/> in the address field of your web browser, followed by the Recommendation's unique ID. For example, <http://handle.itu.int/11.1002/1000/11830-en>

3.6	ITU-T Z.100 App. I	1993-03-12	X	11.1002/1000/3160
3.7	ITU-T Z.100 App. II	1993-03-12	X	11.1002/1000/3161
4.0	ITU-T Z.100	1993-03-12	X	11.1002/1000/3154
4.1	ITU-T Z.100 (1993) Add. 1	1996-10-18	10	11.1002/1000/3917
5.0	ITU-T Z.100	1999-11-19	10	11.1002/1000/4764
5.1	ITU-T Z.100 (1999) Cor. 1	2001-10-29	17	11.1002/1000/5567
6.0	ITU-T Z.100	2002-08-06	17	11.1002/1000/6029
6.1	ITU-T Z.100 (2002) Amd. 1	2003-10-29	17	11.1002/1000/7091
6.2	ITU-T Z.100 (2002) Cor. 1	2004-08-29	17	11.1002/1000/356
7.0	ITU-T Z.100	2007-11-13	17	11.1002/1000/9262
8.0	ITU-T Z.100	2011-12-22	17	11.1002/1000/11387
8.1	ITU-T Z.100 Annex F1	2000-11-24	10	11.1002/1000/5239
8.2	ITU-T Z.100 Annex F2	2000-11-24	10	11.1002/1000/5576
8.3	ITU-T Z.100 Annex F3	2000-11-24	10	11.1002/1000/5577
8.4	ITU-T Z.100 Annex F1	2015-01-13	17	11.1002/1000/12354
8.5	ITU-T Z.100 Annex F2	2015-01-13	17	11.1002/1000/12355
8.6	ITU-T Z.100 Annex F3	2015-01-13	17	11.1002/1000/12356
9.0	ITU-T Z.100	2016-04-29	17	
9.1	ITU-T Z.100 Annex F1	2016-10-29	17	
9.2	ITU-T Z.100 Annex F2	2016-10-29	17	
9.3	ITU-T Z.100 Annex F3	2016-10-29	17	

Keywords

Abstract state machines, ASM, formal definition, overview, overview of semantics, SDL-2010, specification and description language.

FOREWORD

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

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

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

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

NOTE

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

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

INTELLECTUAL PROPERTY RIGHTS

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

As of the date of approval of this Recommendation, ITU had not received notice of intellectual property, protected by patents, which may be required to implement this Recommendation. However, implementers are cautioned that this may not represent the latest information and are therefore strongly urged to consult the TSB patent database at <http://www.itu.int/ITU-T/ipr/>.

© ITU 2017

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

Table of Contents

	Page
F1.1 Preface	1
F1.2 Overview of the semantics	3
F1.3 Abstract State Machines	7

iv

Recommendation ITU-T Z.100

Specification and Description Language – Overview of SDL-2010

Annex F1

SDL-2010 formal definition: General overview

F1.1 Preface

The formal definition of SDL-2010 provided in this annex is a precise language definition, which supplements the definition given in the Recommendation text. It is for use by those requiring a very precise definition of SDL-2010, such as maintainers of the SDL-2010 language, designers of SDL-2010 tools and users of the SDL-2010 language.

The formal definition consists of three annexes:

Annex F1 This annex provides the motivation for and the main objectives of a formal semantics definition for SDL-2010. It gives an overview of the structure of the formal semantics, and contains an introduction to the Abstract State Machine (ASM) formalism, which is used to define the SDL-2010 semantics.

Annex F2 This annex describes the static semantic constraints of SDL-2010, as well as the transformations identified by the 'Model' sections of the ITU-T Z.100 series for SDL-2010.

Annex F3 This annex defines the dynamic semantics of SDL-2010.

F1.1.1 Motivation

SDL-2010 has both a formal syntax and a formal semantics. Annexes F1, F2 and F3 define the formal semantics of SDL-2010. If there is any inconsistency between Annexes F1, F2 and F3 and other parts of the ITU-T Z.100 series for SDL-2010, then there is an error that needs correcting. Neither the other parts of the ITU-T Z.100 series for SDL-2010 nor Annexes F1, F2 and F3 take precedence in this case.

F1.1.2 Main objectives

A primary objective of a formal SDL-2010 semantics is intelligibility, a prerequisite for correctness, acceptance and maintainability. Intelligibility is supported by building on well-known mathematical formalisms and notations, a close correspondence between the specification technique and semantics to be formalized, and by concise and well-structured documentation.

Maintainability is another important objective because SDL-2010 is an evolving technical standard. Apart from the language extensions that are incorporated into this Recommendation, further language features under consideration. Therefore, the mathematical formalism has to be sufficiently rich and flexible such that the formal semantics can be adapted and extended with a reasonable effort.

SDL-2010 can be classified as a model-oriented formal description technique (FDT) for the specification of distributed and concurrent systems, which means that an SDL-2010 specification explicitly defines a set of computations. This calls for an operational semantics in order to achieve a close correspondence with the specification, and thus improve its intelligibility. In addition, operational semantics lends itself naturally to executability, see [b-Eschbach], [b-Eschbach 2001] and [b-Glässer], given the availability of tools, which is another explicit objective.

F1.1.3 References and definitions

The references and definitions of the main body of Recommendation ITU-T Z.100 apply throughout Annexes F1, F2 and F3.

F1.1.4 Bibliographical references (for this annex only)

- [b-ASM] www.eecs.umich.edu/gasm/ (accessed 12 September 2016)
- [b-Blass] Blass, A. and Gurevich, Y. (2003), *Abstract State Machines capture parallel algorithms*, ACM Transactions on Computational Logic, Vol. 4, No. 4, ACM.
NOTE – The axiomatic definition of abstract state machines for sequential algorithms is modified to capture parallel algorithms. Specifically, Bounded Exploration is replaced by Background, Proclet (subprocess of a parallel algorithm that contains no unbounded parallelism) and Bounded Sequentiality to ensure that the number of state elements involved in a given computation step is bounded, with the bound depending only on the algorithm and not on the state.
- [b-Blass 2008] Blass, A. and Gurevich, Y. (2008), *Abstract State machines capture parallel algorithms: Correction and extension*. ACM Transactions on Computational Logic, Vol. 9, No. 3, ACM.
NOTE – The postulates presented in [b-Blass] do not allow proclets to be created on the fly. On the fly creation of proclets is required to correct one of the flaws identified in the examples in Section 8 of [b-Blass]. Other flaws in the earlier article have also been corrected.
- [b-Börger] Börger, E. (2003), *The ASM Refinement Method*, Formal Aspects of Computing Vol. 15, pp. 237-257, BCS.
- [b-Börger & Stärk] Börger, E., and Stärk, R. S. (2003), *Abstract State Machines: A Method for High-Level System Design and Analysis*, Springer-Verlag.
NOTE – Design and analysis for multi-agent as well as single agent abstract state machines. Used here to clarify the coherence condition.
- [b-Eschbach] Eschbach, R., Glässer, U., Gotzhein, R., and Prinz, A. (2000), *On the Formal Semantics of SDL-2000: A Compilation Approach Based on an Abstract SDL Machine*, in: Y. Gurevich, M. Odersky, P. Kutter, L. Thiele (Eds.), *Abstract State Machines – Theory and Applications*, Lecture Notes in Computer Science, Vol. 1912, Springer-Verlag.
- [b-Eschbach 2001] Eschbach, R., Glässer, U., Gotzhein, R., von Löwis, M., and Prinz, A. (2001), *Formal Definition of SDL-2000: Compiling and Running SDL Specifications as ASM Models*, Journal of Universal Computer Science Vol. 7, No. 11, pp. 1024-1049, Springer.
- [b-Glässer] Glässer, U., Gotzhein, R., Prinz, A. (2003), *The formal semantics of SDL-2000 – Status and perspectives*, Computer Networks, Vol. 42, No. 3, pp. 343-358, Elsevier Sciences.
NOTE – The design objectives of the SDL semantics include executability, intelligibility, conciseness and flexibility as well as the ideals of correctness and completeness (for which indisputable evidence cannot be inferred from the SDL grammars and textual description). The decision to base the SDL formal semantics on ASM is documented, and the ITU-T approach is described. This approach entails analysis of an SDL model and synthesis of an ASM program that defines the behaviour of SDL agents. Execution is defined in terms of the SDL virtual machine, which provides operating system functionality that controls the execution of ASM programs on the logical hardware of the SDL abstract machine (SAM).

- [b-Glässer 2007] Glässer, U., Gurevich, Y., and Veanes, M. (2007), *Abstract Communication Model for Distributed Systems*, IEEE Transactions on Software Engineering, Vol. 30, No. 7, pp. 458-472.
- NOTE – A high level abstract model for message based communication networks is presented. The model is based on distributed abstract state machines, has been implemented in AsmL and has been used for testing distributed systems.
- [b-Glausch] Glausch, A. and Reisig, W. (2007), *A Semantic Characterization of Unbounded-Nondeterministic Abstract State Machines*, in T Mossakowski et al. (Eds.) CALCO 2007, LNCS 4624, Springer-Verlag.
- NOTE – The axiomatic definition given by Gurevich for the sequential algorithms captured by ASMs is extended to nondeterministic ASMs. Unbounded nondeterminism means that there may be uncountably many update sets that could be produced by an algorithm in a given state. However, so long as each of these is bounded in size, the fact that only one of them is applied means that the number of state elements involved in a computation step is bounded.
- [b-Gurevich] Gurevich, Y. (1995), *Evolving algebras 1993: Lipari guide*, in Specification and validation methods, Börger, E. (ed.), pp. 9-36, Oxford University Press.
- [b-Gurevich 2000] Gurevich, Y. (2000), *Sequential Abstract State Machines Capture Sequential Algorithms*, Microsoft Research.

F1.2 Overview of the semantics

In order to define the formal semantics of SDL-2010, the language definition is decomposed into several parts:

- grammar
- well-formedness conditions
- transformation rules
- dynamic semantics.

The starting point for defining the formal semantics of SDL-2010 is a syntactically correct SDL-2010 specification, represented as an abstract syntax tree (AST).

The first three parts of the formal semantics are collectively referred to as *static semantics* or *static aspects* in the context of SDL-2010 (see Figure F1-1), and are described in Part 2 of the formal definition, i.e., Annex F2.

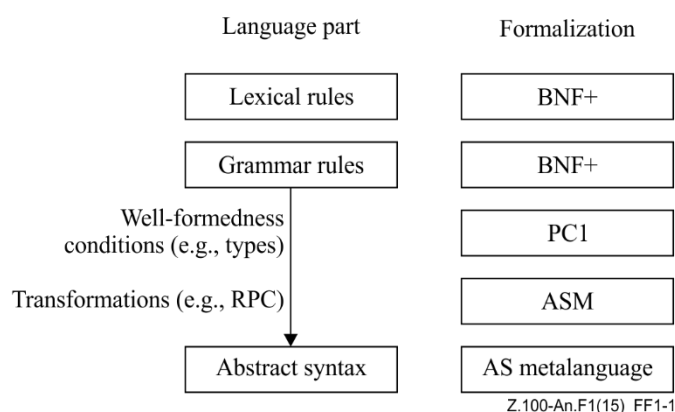


Figure F1-1 – Static aspects of SDL-2010

The *grammar* defines the set of syntactically correct SDL-2010 specifications. The ITU-T Z.100 series for SDL-2010 defines a concrete graphical grammar, a concrete textual grammar, and an

abstract grammar. The syntax of the concrete grammars is defined formally using the Backus-Naur form (BNF) with some extensions to capture the graphical language constructs (see clause 5.4.1 of [ITU-T Z.111]). The abstract grammar is obtained from the concrete grammars by removing irrelevant details such as separators and lexical rules, and by applying transformation rules (see below). The syntax of the abstract grammar is defined in the textual presentation metalanguage for abstract grammar (see clause 5.4.1.1 of [ITU-T Z.111]).

From the specifications that are correct with respect to the grammar syntax, the *well-formedness conditions* define the specifications that are also correct with respect to context information. For example, the well-formedness conditions define what names it is allowed to use at a given place, or what kind of values it is allowed to assign to variable. The well-formedness conditions are defined in terms of first order predicate calculus (PC1).

Furthermore, some language constructs appearing in the concrete grammars are replaced by other language elements in the abstract grammar using *transformation rules* to keep the set of core concepts small. These transformations are described in the 'Model' paragraphs of the ITU-T Z.100 series for SDL-2010, and are formally expressed as rewrite rules.

The *dynamic semantics* apply only to syntactically correct SDL-2010 specifications that satisfy the well-formedness conditions. The dynamic semantics defines the set of computations associated with a specification, and are described in Part 3 of the formal definition, i.e., Annex F3.

F1.2.1 Grammar

The *grammar* of SDL-2010 is formalized as described above. The primary concrete grammar is given for SDL-GR. Most of the grammar of SDL-GR is textual, but it has some graphical elements. To enable formalisation of SDL-2010 specifications into ASM, any SDL-GR graphical element is changed to the equivalent concrete textual representation (SDL-PR) defined in [ITU-T Z.106]. The grammar in the ITU-T Z.100 series for SDL-2010 is designed to be a presentation grammar: it is not made to generate a parser automatically. Moreover, some restrictions that finally guarantee uniqueness of the semantics cannot be expressed in BNF and have been stated in the text instead. Therefore, the grammar is defined using BNF and some text (mostly for the precedence rules). The translation from the concrete textual SDL-2010 representation to the abstract syntax representation of SDL-2010 (called AS1) consists of two steps. The first step from the concrete textual SDL-2010 representation to AS0 (the concrete syntax with irrelevant details such as separators and lexical rules removed) is not formally defined, but is derived from the correspondence between the two grammars, which is almost one-to-one. The second step, translating AS0 to AS1, is formally captured by a set of transformation rules (see Annex F2).

F1.2.2 Well-formedness conditions

The *well-formedness conditions* define additional constraints that a well-formed SDL-2010 specification has to satisfy. These constraints cannot be expressed using context-free grammar rules, but they are static, and can be defined and checked independently of the dynamic semantics of SDL-2010 (see Annex F2). An SDL-2010 specification is *valid* if and only if it satisfies the syntactical rules and the static conditions of SDL-2010. In fact, the well-formedness conditions refer to the syntax, but they have not been stated in the concrete syntax because they are not expressible in a context-free grammar.

There are five kinds of well-formedness conditions:

- *Scope/visibility rules*: The definition of an entity introduces an identifier used as the reference to the entity. Only the use of visible identifiers is allowed. The scope/visibility rules are applied to determine whether the corresponding definition of an identifier is visible or not.
- *Disambiguation rules*: Sometimes a name might refer to several identifiers. Rules are applied to find out the correct one.

- *Data type consistency rules*: These rules ensure that dynamically, no operation is applied to operands that do not match its argument types. More specifically, the data type of an actual parameter has to be compatible with that of the corresponding formal parameter; and the data type of an expression has to be compatible with that of the variable to which the expression is assigned.
- *Special rules*: There are some rules applicable to specific entities. For example, it is not allowed to export a procedure variable (that is, a variable defined within a procedure).
- *Plain syntax rules*: There are some rules that refer to the correctness of the concrete syntax, and that have no counterpart in the abstract syntax. For instance, the names at the beginning and at the end of a definition in SDL-PR have to match.

F1.2.3 Transformation rules

For a language with a rich syntax, it is important to identify the core concepts matching the intentions of the language designer. Further language constructs that are introduced for convenience, but do not add to the expressiveness of the language (such as shorthand notations), can be replaced using these core concepts. Since replacements, which are described by transformation rules, can be formalized, it suffices to define the dynamic semantics only for the core concepts, which adds to its conciseness and intelligibility. In Figure F1-1, the general approach is shown. The language is defined with its concrete grammar using lexical and syntax rules. Consistency constraints are defined on this concrete grammar.

The ITU-T Z.100 series for SDL-2010 prescribes the transformation of SDL-2010 specifications by a sequence of *transformation steps*. Each transformation step consists of a set of single transformations as stated in the *Model* clauses, and determines how to handle one special class of shorthand notations. The result of one step is used as input for the next step.

To formalize the transformation rules of SDL-2010, the rewrite rules in PC1 are used. These rules define patterns of the AST, which are to be replaced by other AST patterns. In fact, several groups of such rewrite rules are defined that are applied in turn. A single transformation is realized by the application of a rewrite rule to the concrete specification, which essentially means to replace parts of the specification by other parts as defined by the rule (see Annex F2).

F1.2.4 Dynamic semantics

The *dynamic semantics* (clauses F3.2 and F3.3) consists of the following parts (see Figure F1-2):

- a) The *SDL-2010 Abstract Machine (SAM)*: this is defined using ASM. The definition of the SAM is divided into three parts, corresponding to the abstract syntax:
 - i) basic signal flow concepts (such as signals, timers, gates, channels) defined in terms of an ASM model in clause F3.2.1.1;
 - ii) various types of ASM agents to model corresponding SDL-2010 agents in clause F3.2.1.2; and
 - iii) signal processing and behaviour primitives (the abstract machine instructions of the SAM) in clause F3.2.1.4 (that uses the interface to the data type part in clause F3.2.1.3).
- b) The *compilation function* (clause F3.2.2): this maps the AST of an SDL-2010 specification to SAM behaviour primitives that model the actions of the SDL-2010 agents. The compilation function amounts to an abstract compiler taking the AST of the state machines as input and transforming it to SAM instructions.
- c) The *SAM Programs* (clause F3.2.3): these define the set of computations. These programs consist of an initialization phase and an execution phase. SAM programs have fixed parts that are the same for all SDL-2010 specifications, and variable parts that are generated from the abstract syntax representation of a given SDL-2010 specification.