International Telecommunication Union

ITU-T

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

Z.106

(06/2021)

SERIES Z: LANGUAGES AND GENERAL SOFTWARE ASPECTS FOR TELECOMMUNICATION SYSTEMS

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

Specification and Description Language – Common interchange format for SDL-2010

Recommendation ITU-T Z.106

Documents
Document Sharing Hub

Full document details: ITU-T Z.106 2021

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.

Documents
Document Sharing Hub

Full document details: ITU-T Z.106 2021

Recommendation ITU-T Z.106

Specification and Description Language – Common interchange format for SDL-2010

Summary

Recommendation ITU-T Z.106 defines the common interchange format of Specification and Description Language (SDL-CIF). The SDL-CIF is intended for the interchange of graphical SDL-2010 specifications (SDL-GR) made on different tools that do not use the same storage format. Prior to the definition of SDL-CIF, the textual phrase representation of SDL-2010 (SDL-PR) was used to interchange specifications with the disadvantage that all graphical information was lost, making the same specifications often look very dissimilar in different environments. With the SDL-CIF, this disadvantage is reduced to a minimum, as it contains most of the graphical information. The SDL-CIF improves the independence from specific tool vendors and allows standards bodies to accept specifications in SDL-CIF irrespective of the tool they use for their internal work. This also improves productivity by allowing specifications to be made on the accustomed tool. All SDL-2010 tool vendors are encouraged to provide facilities for importing and exporting SDL-CIF.

This Recommendation defines how SDL-2010 descriptions are stored in order to be interchanged between tools coming from different vendors. It does not take into account the message sequence chart (MSC) notation. SDL-CIF is an optional part of SDL-2010. SDL-CIF is based on the SDL-PR syntax, the textual phrase representation of SDL-2010 also defined in this Recommendation. SDL-CIF is readable and written by tools as well as users. All the constructs available in SDL-2010 are able to be expressed in graphical form or in the purely textual SDL-PR form. Constraints on graphical presentation are expressed in SDL-CIF by adding specific annotations to SDL-PR. As a result, most SDL-PR descriptions are legal SDL-CIF descriptions. SDL-CIF is an open storage format as it includes a mechanism of tool-specific directives. This mechanism allows an SDL-CIF-compliant tool to extend the format by adding specific information. SDL-CIF is also easily implemented and provides tool vendors with two levels of tool conformance and concepts of mandatory and optional directives.

SDL-PR is an alternative text-only syntax for the Specification and Description Language. Before 2002, SDL-PR was published as part of ITU-T Z.100, but as the main use of this notation is for communication within and between tools the definition has been moved to this Recommendation. SDL-PR is Level 0 SDL-CIF and allows the interchange of syntactically complete SDL-2010 descriptions, usually as a single file per system. Conformance to SDL-PR requires the model to conform to the corresponding semantics defined in Recommendations ITU-T Z.101, ITU-T Z.102, ITU-T Z.103, ITU-T Z.104, ITU-T Z.105 and ITU-T Z.107.

This Recommendation introduces two further levels of SDL-CIF. Two further conformance levels are defined, one at a more liberal SDL-PR level and the second including graphical information. The complete grammar is described with the related semantics. Mandatory and optional directives are described, as well as the format for tool-specific directives. Current tool-specific directives are described in Appendix I.

Two levels of SDL-CIF conformance are defined as level 1 and level 2. Level 1 is very close to SDL-PR, but it supports incomplete descriptions in SDL-2010. Level 2 includes level 1 and is able to capture most of the graphical information of SDL-GR diagrams. An SDL-CIF specification shall identify which of these two levels it complies with. Similarly, tool vendors that use the SDL-CIF should also identify the SDL-CIF level they comply with for their import and export functions.

Documents
Document Sharing Hub
Full document details: ITU-T Z.106 2021

History

Edition	Recommendation	Approval	Study Group	Unique ID*
1.0	ITU-T Z.106	1996-10-18	10	11.1002/1000/3918
2.0	ITU-T Z.106	2000-11-24	10	11.1002/1000/5240
3.0	ITU-T Z.106	2002-08-06	17	11.1002/1000/6030
4.0	ITU-T Z.106	2011-12-22	17	11.1002/1000/11393
5.0	ITU-T Z.106	2016-04-29	17	11.1002/1000/12860
6.0	ITU-T Z.106	2019-10-14	17	11.1002/1000/14057
7.0	ITU-T Z.106	2021-06-13	17	11.1002/1000/14676

Keywords

Common Interchange Format, directives, SDL-2010, SDL-CIF, SDL-PR, Specification and Description Language, textual phrase representation.

* 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.

Documents
Document Sharing Hub
Full document details: ITU-T Z.106 2021

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/software copyrights, 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 appropriate ITU-T databases available via the ITU-T website at http://www.itu.int/ITU-T/ipr/.

© ITU 2021

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

Documents
Document Sharing Hub
Full document details: ITU-T Z.106 2021

Table of Contents

		Page
1	Scope	1
2	References	1
3	Definitions and abbreviations	1
3.1	Definitions	1
3.2	Abbreviations and acronyms	1
4	Conventions	2
5	Level 0 SDL-CIF (SDL-PR)	2
5.1	General principles	2
5.2	General rules	3
5.3	Organization of SDL-2010 specifications	3
5.4	Structural concepts	5
5.5	Agents	7
5.6	Communication	9
5.7	Behaviour	11
5.8	Data	17
5.9	Generic system definition	18
6	Level 1 SDL-CIF (CIF-PR)	19
6.1	General principles	19
6.2	CIF-PR syntax	20
7	Level 2 SDL-CIF (CIF-GR)	20
7.1	General principles	20
7.2	General principles of graphical information	21
7.3	CIF-GR lexical rules	23
7.4	CIF-GR syntax: SDL-CIF A rules	24
7.5	CIF-GR Syntax – SDL-CIF B rules	44
7.6	Tool-specific SDL-CIF comments	47
8	Examples	47
8.1	DemonGame	48
8.2	Tricky SDL-2010 constructs	50
8.3	Situations SDL-CIF is not able to handle	56
9	SDL-CIF conformance criteria	57
9.1	About tools reading a SDL-CIF file	57
9.2	Automatic versus forced layout	57
9.3	Retention and use of tool-specific information	58
Appendix I – Tool-specific SDL-CIF comments		59
I.1	Maintenance of SDL-CIF	59
I.2	Current tool-specific SDL-CIF comments	59

Documents
Document Sharing Hub
Full document details: ITU-T Z.106 2021

Introduction

For a number of years, the Specification and Description Language has been increasingly used, both in industry and for standards and Recommendations. Whereas, in industry, the Specification and Description Language is often used in an environment with a single tool, environments for standards and Recommendation creation often require the integration of Specification and Description Language specifications from many tools used by different organizations. This is often also a requirement in international projects.

Until the time Recommendation ITU-T Z.106 had been agreed, the only way to interchange specifications in the Specification and Description Language had been to interchange SDL-PR (the textual representation of the language). This has led to the loss of graphical information. Though not necessary from the formal viewpoint, graphical information often had a significant impact on readability and comprehensibility. With the common interchange format, this Recommendation fulfilled a long-expressed need for the interchange of specifications of the Specification and Description Language without the loss of graphical information.

Documents
Document Sharing Hub
Full document details: ITU-T Z.106 2021

Documents
Document Sharing Hub

Full document details: ITU-T Z.106 2021

Recommendation ITU-T Z.106

Specification and Description Language – Common interchange format for SDL-2010

1 Scope

This Recommendation defines the common interchange format (SDL-CIF) for specifications written in Specification and Description Language. It is intended for tool vendors as an export and import format to allow the interchange of specifications with tools offered by other tool vendors. Even though the format allows writing specifications in SDL-CIF directly, it is not intended for this purpose, but rather should be generated from an existing specification written according to the Specification and Description Language as defined in [ITU-T Z.100], [ITU-T Z.101], [ITU-T Z.102], [ITU-T Z.103], [ITU-T Z.104], [ITU-T Z.105] and [ITU-T Z.107].

2 References

The following ITU-T Recommendations and other references contain provisions which, through reference in this text, constitute provisions of this Recommendation. At the time of publication, the editions indicated were valid. All Recommendations and other references are subject to revision; users of this Recommendation are therefore encouraged to investigate the possibility of applying the most recent edition of the Recommendations and other references listed below. A list of the currently valid ITU-T Recommendations is regularly published. The reference to a document within this Recommendation does not give it, as a stand-alone document, the status of a Recommendation.

[ITU-T Z.100] Recommendation ITU-T Z.100 (2021), *Specification and Description Language – Overview of SDL-2010*.

[ITU-T Z.101] Recommendation ITU-T Z.101 (2021), *Specification and Description Language – Basic SDL-2010*.

[ITU-T Z.102] Recommendation ITU-T Z.102 (2021), *Specification and Description Language – Comprehensive SDL-2010*.

[ITU-T Z.103] Recommendation ITU-T Z.103 (2021), *Specification and Description Language – Shorthand notation and annotation in SDL-2010*.

[ITU-T Z.104] Recommendation ITU-T Z.104 (2021), *Specification and Description Language – Data and action language in SDL-2010*.

[ITU-T Z.105] Recommendation ITU-T Z.105 (2021), *Specification and Description Language – SDL-2010 combined with ASN.1 modules*.

[ITU-T Z.107] Recommendation ITU-T Z.107 (2021), *Specification and Description Language – Object-oriented data in SDL-2010*.

[ITU-T Z.111] Recommendation ITU-T Z.111 (2016), *Notations and guidelines for the definition of ITU-T languages*.

3 Definitions and abbreviations

3.1 Definitions

None.

3.2 Abbreviations and acronyms

This Recommendation uses the following abbreviation and acronyms:

Documents
Document Sharing Hub
Full document details: ITU-T Z.106 2021

CIF-GR	Common Interchange Format for Specification and Description Language Graphic Representation
CIF-PR	Common Interchange Format for Specification and Description Language textual Phrase Representation
SDL-CIF	Specification and Description Language Common Interchange Format
SDL-GR	Specification and Description Language Graphic Representation
SDL-PR	Specification and Description Language textual Phrase Representation

4 Conventions

For the definition of properties and syntaxes of SDL-CIF, the meta-language defined in clause 5.4.2 of [ITU-T Z.111] has been used with the following addition: the last alphanumeric part of a rule name gives the rules category (A, B or C) followed by an identifying number in that category. Non-terminal symbols refer to non-terminals of [ITU-T Z.101], [ITU-T Z.102], [ITU-T Z.103], [ITU-T Z.104] or [ITU-T Z.107] or to non-terminals introduced in this Recommendation.

The following non-terminal rules are intentionally defined but not used because they are starting rules for syntax:

<sdl specification>,
<cif description a1>,
<cif level 1 file>.

The following non-terminal rule is intentionally not used, because it is a general descriptive rule for SDL-CIF comments:

<cif directive>.

5 Level 0 SDL-CIF (SDL-PR)

5.1 General principles

SDL-CIF level 0 introduces an additional syntactic form for representing a system: the textual phrase representation (SDL-PR). It is allowed to use SDL-PR instead of the graphic representation of [ITU-T Z.101], [ITU-T Z.102], [ITU-T Z.103], [ITU-T Z.104] or [ITU-T Z.107], which is referred to as SDL-GR in this document. As both are concrete representations of the same abstract grammar, they are equivalent. In particular, they are both equivalent to the abstract grammar for the corresponding concepts.

A subset of SDL-GR is common with SDL-PR. This subset is called common textual grammar.

Although SDL-2010 is allowed to be written in either SDL-PR or SDL-GR, the language has been designed with the knowledge that SDL-PR will be used infrequently for purposes such as interchanging between tools, albeit the common interchange format specified in the remainder of this document further diminishes the use of SDL-PR. Most users use SDL-GR.

Figure 1 shows the relationships between SDL-PR, SDL-GR, the concrete grammars and the abstract grammar.

Documents
Document Sharing Hub
Full document details: ITU-T Z.106 2021

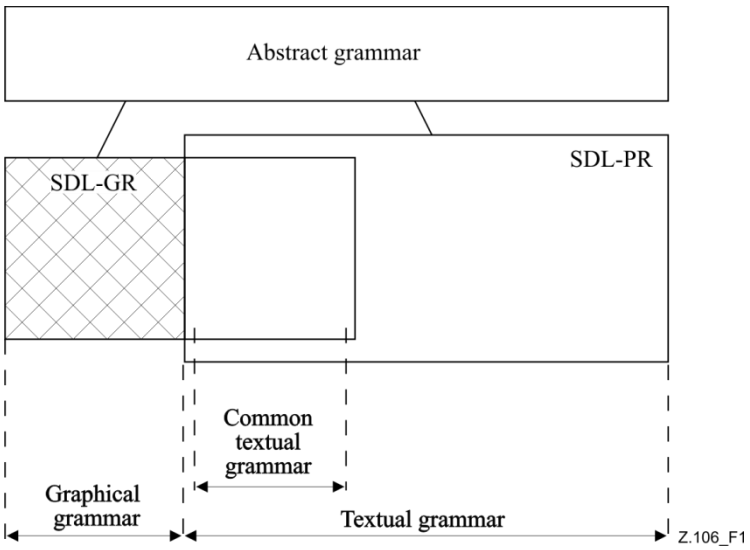

Figure 1 – SDL-2010 grammars

Each of the concrete grammars has a definition of its own syntax and of its relationship to the abstract grammar (that is, how to transform into the abstract syntax). Using this approach, there is only one definition of the semantics of SDL; each of the concrete grammars inherits the semantics via its relationship to the abstract grammar (in [ITU-T Z.101], [ITU-T Z.102], [ITU-T Z.104] and [ITU-T Z.107]). This approach also ensures that SDL-PR and SDL-GR are equivalent.

For some constructs there is no directly equivalent abstract syntax. In these cases, a model is given for the transformation from concrete syntax into the concrete syntax of other constructs that (directly or indirectly via further models) have an abstract syntax. Items that have no mapping to the abstract syntax (such as comments) do not have any formal meaning.

The concrete textual syntax for SDL-PR is applied as defined in *Concrete grammar* of clause 5.3.2 of [ITU-T Z.111] and is specified in the extended Backus-Naur Form of syntax description defined in clause 5.4.2 of [ITU-T Z.111]. Underlined semantic sub-categories (see clause 5.4.2 of [ITU-T Z.111]) are significant.

Along with the productions for the concrete syntax of SDL-PR, *Model* clauses are given for constructs that are "derived concrete syntax" for other equivalent concrete syntax constructs as defined in *Model* of clause 5.3.2 of [ITU-T Z.111].

NOTE – To aid understanding, the following titles of the clauses often correspond to the titles of related clauses in [ITU-T Z.101], [ITU-T Z.102], [ITU-T Z.103], [ITU-T Z.104] and [ITU-T Z.107].

5.2 General rules

In SDL-PR, the optional name or identifier in a definition after the ending keywords (**endsystem**, **endblock**, etc.) in definitions shall be syntactically the same as the name or identifier following the corresponding commencing keyword (**system**, **block**, etc., respectively).

5.3 Organization of SDL-2010 specifications

5.3.1 Framework

The starting production rule of clause 7.1 of [ITU-T Z.103] <sdl specification>, is replaced by the following production:

```
<sdl specification> ::=
                    {       { <package diagram> | <package definition> | <module definition>
                                |       <system specification> | <textual system specification> }
                            <referenced definition>* }set
```


Documents
Document Sharing Hub
Full document details: ITU-T Z.106 2021

<module definition> ::=
ModuleDefinition

where the non-terminal **ModuleDefinition** is defined in ASN.1 as defined in clause 6 of [ITU-T Z.105].

<textual system specification> ::=
<agent definition>
| {<package use clause>}* <textual typebased agent definition>

The alternatives <package diagram> and <system specification> of <sdl specification> are not part of SDL-PR.

Model

A <textual system specification> being a <process definition> or a <textual typebased process definition> is derived syntax for a <system definition> having the same name as the process, containing implicit channels and containing the <process definition> or <textual typebased process definition> as the only definition.

A <textual system specification> being a <block definition> or a <textual typebased block definition> is derived syntax for a <system definition> having the same name as the block, containing implicit channels and containing the <block definition> or <textual typebased block definition> as the only definition.

A <package use clause> before a <textual typebased agent definition> of a <textual system specification> is derived syntax for a <package use clause> before the <system heading> in the <system definition> derived from the <textual typebased agent definition>.

5.3.2 Package

<package definition> ::=
{<package use clause>}*
<package heading> <end>
{<entity in package>}*
endpackage [<package name>] <end>

<entity in package> ::=
<agent type definition>
| <agent type reference>
| <package definition>
| <package reference>
| <signal definition list>
| <signal list definition>
| <remote variable definition>
| <data definition>
| <procedure definition>
| <procedure reference>
| <remote procedure definition>
| <composite state type definition>
| <composite state type reference>
| <select definition>
| <macro definition>

5.3.3 Referenced definition

The alternative <diagram> of <referenced definition> is not part of SDL-PR.

The production rule <definition> (see clause 7.3 of [ITU-T Z.104]), is replaced by the following production:

Documents
Document Sharing Hub
Full document details: ITU-T Z.106 2021

<definition> ::=
<package definition>
| <agent definition>
| <agent type definition>
| <composite state definition>
| <composite state type definition>
| <procedure definition>
| <operation definition>
| <macro definition>

Each <definition> item (except <macro definition> items) shall have a corresponding reference in the associated <package diagram> or <system specification>.

5.4 Structural concepts

5.4.1 Types, instances and gates

5.4.1.1 Structural type definitions

5.4.1.1.1 Agent type

<agent type definition> ::=
<system type definition> | <block type definition> | <process type definition>

5.4.1.1.2 System type

<system type definition> ::=
<package use clause>*
<system type heading> <end> <agent structure>
endsystem type [[<qualifier>] <system type name>] <end>

If there is a <system type name> after **endsystem type**, this shall be the same as the <system type name> in the <system type heading>. If there is a <qualifier> after **endsystem type**, this shall identify the qualifier of the identifier for the system type.

5.4.1.1.3 Block type

<block type definition> ::=
<package use clause>*
<block type heading> <end> <agent structure>
endblock type [[<qualifier>] <block type name>] <end>

If there is a <block type name> after **endblock type**, this shall be the same as the <block type name> in the <block type heading>. If there is a <qualifier> after **endblock type**, this shall identify the qualifier of the identifier for the block type.

5.4.1.1.4 Process type

<process type definition> ::=
<package use clause>*
<process type heading> <end> <agent structure>
endprocess type [[<qualifier>] <process type name>] <end>

If there is a <process type name> after **endprocess type**, this shall be the same as the <process type name> in the <process type heading>. If there is a <qualifier> after **endprocess type**, this shall identify the qualifier of the identifier for the process type.

5.4.1.2 Composite state type

<composite state type definition> ::=
<composite state type graph> | <state aggregation type>

Documents
Document Sharing Hub
Full document details: ITU-T Z.106 2021

<composite state type graph> ::=
{<package use clause>}*
<composite state type heading> <end>
<composite state structure>
endsubstructure state type [[<qualifier>] <composite state type name>] <end>

<state aggregation type> ::=
{<package use clause>}*
<state aggregation type heading> <end>
<aggregation structure>
endsubstructure state type [[<qualifier>] <composite state type name>] <end>

In <composite state type graph> and <state aggregation type>, if there is a <composite state type name> after **endsubstructure state type**, this shall be the same as the <composite state type name> in the <composite state type heading>. If there is a <qualifier> after **endsubstructure state type**, this shall identify the qualifier of the identifier for the composite state type.

<composite state type heading> ::=
<type preamble>
state type [substructure] [<qualifier>] <composite state type name>
[<formal context parameters>] [<virtuality constraint>]
[<specialization>]
[<agent formal parameters>]

<state aggregation type heading> ::=
<type preamble>
state aggregation type [substructure] [<qualifier>] <composite state type name>
[<formal context parameters>] [<virtuality constraint>]
[<specialization>]
[<agent formal parameters>]

The optional keyword **substructure** is added to <composite state type heading> and <state aggregation type heading> to match the keyword **endsubstructure** needed to close the <composite state type definition>.

5.4.1.2.1 Definitions based on types

<textual typebased agent definition> ::=
<textual typebased system definition>
| <textual typebased block definition>
| <textual typebased process definition>

The agent type denoted by <base type> in the type expression of a <textual typebased agent definition> shall contain an unlabelled start transition in its state machine.

5.4.1.2.2 System definition based on system type

<textual typebased system definition> ::=
<typebased system heading> <end>

5.4.1.2.3 Block definition based on block type

<textual typebased block definition> ::=
block <typebased block heading> <end>

5.4.1.2.4 Process definition based on process type

<textual typebased process definition> ::=
process <typebased process heading> <end>

5.4.1.2.5 Composite state definition based on composite state type

<textual typebased state partition def> ::=
state aggregation <typebased state partition heading> <end>

Documents
Document Sharing Hub
Full document details: ITU-T Z.106 2021

5.4.1.3 Gate

<gate in definition> ::=
<textual gate definition> | <textual interface gate definition>

<textual gate definition> ::=
gate <gate> [<encoding rules>] [**adding**] <gate constraint> <end>

<textual interface gate definition> ::=
gate { **in** | **out** }
with <interface identifier> [<encoding rules>] [<textual endpoint constraint>] <end>

It is only allowed to specify **adding** in a subtype definition and only for a gate defined in the supertype. When **adding** is specified for a <gate>, any <textual endpoint constraint>s and <signal list>s are additions to the <gate constraint>s of the gate in the supertype.

5.5 Agents

<agent definition> ::=
<system definition> | <block definition> | <process definition>

<agent structure> ::=
{ <entity in agent>}*
{ <interaction> | <agent body> }

<interaction> ::=
{ <channel to channel connection>
| <channel definition>
| <agent definition>
| <block reference>
| <process reference>
| <textual typebased agent definition> }*
[<state machine>]

The <state machine> defines the state machine of the agent. If <agent structure> is able to be understood both as either <state machine> or an <agent body>, it is interpreted as <agent body>.

A <channel to channel connection> shall not be contained within an <agent type definition>.

<entity in agent> ::=
<block type definition>
| <block type reference>
| <composite state definition>
| <composite state type definition>
| <composite state type reference>
| <data definition>
| <gate in definition>
| <imported procedure specification>
| <imported variable specification>
| <macro definition>
| <procedure definition>
| <procedure reference>
| <process type definition>
| <process type reference>
| <remote procedure definition>
| <remote variable definition>
| <select definition>
| <signal definition list>
| <signal list definition>
| <timer definition>
| <valid input signal set>
| <variable definition>

NOTE 1 – A composite state reference is not allowed as an <entity in agent>. The composite state is instead referenced by an item in a <state> of a <agent body> or a <state machine> of an <interaction>.

Documents
Document Sharing Hub
Full document details: ITU-T Z.106 2021

<block reference> ::=

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

<process reference> ::=

process <process name> [**adding**] [<number of instances>] **referenced** <end>

The <block reference> and <process reference> of [ITU-T Z.103] are extended to include the optional keyword **adding** for inherited agent definitions. If the keyword **adding** is present the agent reference shall be in a subtype and shall name an agent defined in the supertype. If **adding** is given, a <number of instances> item represents the *Number-of-instances* of the inherited *Agent-definition* in the subtype; otherwise the *Number-of-instances* is the same as the *Number-of-instances* of the inherited *Agent-definition* in the supertype.

<agent body> ::=

[<start>]
{ <state> | <free action> }*

<state machine> ::=

state substructure { <typebased composite state> | <composite state list item>}
referenced <end>

NOTE 2 – No notation is needed for inherited state machines, because any additional connection to the gates of the subtype state machine are given by a <channel definition> with a <channel path> where the <channel endpoint> is **this**.

5.5.1 System

<system definition> ::=

{<package use clause>}*
<system heading> <end> <agent structure>
endsystem [<system name>] <end>

5.5.2 Block

<block definition> ::=

{<package use clause>}*
<block heading> <end> <agent structure>
endblock [[<qualifier>] <block name>] <end>

If there is a <block name> after **endblock**, this shall be the same as the <block name> in the <block heading>. If there is a <qualifier> after **endblock**, this shall identify the qualifier of the identifier for the block.

5.5.3 Process

<process definition> ::=

{<package use clause>}*
<process heading> <end> <agent structure>
endprocess [[<qualifier>] <process name>] <end>

If there is a <process name> after **endprocess**, this shall be the same as the <process name> in the <process heading>. If there is a <qualifier> after **endprocess**, this shall identify the qualifier of the identifier for the process.

5.5.4 Procedure

<internal procedure definition> ::=

{<package use clause>}*
<procedure heading> [<end> <entity in procedure>+]
{ [<virtuality>] [<comment body>] <left curly bracket>
<statements> <end>*
<right curly bracket>
| [<procedure body>]
endprocedure [[<qualifier>] <procedure name>] <end> }

Documents
Document Sharing Hub
Full document details: ITU-T Z.106 2021

If there is a <procedure name> after **endprocedure**, this shall be the same as the <procedure name> in the <procedure heading>. If there is a <qualifier> after **endprocedure**, this shall identify the qualifier of the identifier for the procedure.

The syntax of <internal procedure definition> is extended to allow a <procedure body> as an alternative to <statements> enclosed in curly brackets.

<procedure body> ::=
[<start>] { <state> | <free action> }*

An **endprocedure** within an inner <internal procedure definition> in an <entity in procedure> of an outer <internal procedure definition> belongs to the inner <internal procedure definition>.

<entity in procedure> ::=
<variable definition>
| <data definition>
| <select definition>
| <macro definition>
| <procedure definition>
| <procedure reference>
| <composite state definition>
| <composite state type definition>
| <composite state type reference>

The syntax of <entity in procedure> is extended to allow composite state and state type definitions to be used in a <procedure body>.

5.5.5 Agent and composite state reference

<composite state reference> ::=
state substructure <composite state name> **referenced** <end>

5.6 Communication

5.6.1 Channel

<channel definition> ::=
channel [<channel name> [<encoding rules>]] [**nodelay**]
<channel path> [<channel path>]
endchannel [<channel name>] <end>

<channel path> ::=
from <channel endpoint>
to <channel endpoint> [**with** <signal list>] <end>

<channel endpoint> ::=
{ <agent identifier> | <state identifier> | **env** | **this** }[**via** <gate>]

It is only allowed to specify the ending <channel name> if the starting <channel name> is specified. If the starting <channel name> is not specified, the channel shall not be referred to by name.

<gate> shall be specified if:

a) <channel endpoint> with an <agent identifier> denotes a connection to an agent definition, in which case the <gate> shall be defined directly in the agent type for that agent; or

b) <channel endpoint> with a <state identifier> identifies the state machine of the enclosing agent, in which case the <gate> shall be defined directly in the state type for that state, and as there is only one state machine it can alternatively be identified by **this**; or

c) **env** is specified and the channel is defined in an agent type in which case the <gate> shall be defined in this agent type.

If <gate> is specified, the channel is connected to that gate. The gate and the channel shall have at least one common element in their signal lists in the same direction. If no <gate> is specified, the following applies: If the channel endpoint is an agent or state machine and that agent/state contains a

Documents
Document Sharing Hub
Full document details: ITU-T Z.106 2021

<channel to channel connection> for the channel, the channel is connected to the implicit gate introduced by the <channel to channel connection>. Otherwise, if the channel endpoint is a state, the channel is connected to the implicit gate of that state machine; the channel introduces an implicit gate on the agent or state mentioned in <channel endpoint>. This gate obtains the <signal list> of the respective <channel path>s as its corresponding gate constraint. The channel is connected to that gate.

If a <channel definition> contains two <channel path>s, then the following conditions shall be satisfied.

a) The <channel endpoint> following **from** in the first <channel path> shall be the same as the <channel endpoint> following **to** in the second <channel path>.

b) The <channel endpoint> following **to** in the first <channel path> shall be the same as the <channel endpoint> following **from** in the second <channel path>.

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

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

5.6.2 Connection

<channel to channel connection> ::=
connect <external channel identifiers>
and <channel identifier> {, <channel identifier>}* <end>

It is not allowed to mention a channel after the keyword **and** in more than one <channel to channel connection> of a given scope unit.

For any pair of <channel to channel connection>s of a given scope unit, the <external channel identifiers>s shall either mention the same set of channels, or shall have no channels in common. If two or more <channel to channel connection>s of a given scope unit have the same set of external channels, this is derived syntax for a single <channel to channel connection> having one of the <external channel identifiers> as its <external channel identifiers>, and <channel identifier> list formed by listing all the internal <channel identifier> items.

NOTE – Because of the **connect** construct with an (external) channel, there is a need to have a name in the corresponding textual version, whereas the channel is allowed to be anonymous in the graphical version of a specification. This is completely analogous to the case of <merge area>s in process or procedure graphs. A tool that converts the graphical version of a specification to a textual version should thus be able to generate implicit channel names.

Model

Each different <channel to channel connection> in a given scope unit defines one implicit gate on the scope unit. All channels in the <channel to channel connection> are connected to that gate in their respective scope units. The gate constraints of the implicit gate are derived from the channels connected to the gate.

Documents
Document Sharing Hub
Full document details: ITU-T Z.106 2021

The name of the gate is a unique and unambiguous derived name. In the surrounding scope unit, the <channel definition> that is identified by the <channel identifier> is extended with a **via** <gate> part. The **via** <gate> part is added to the <channel endpoint> that references the current scope unit and it mentions the implicit gate. Inside the scope unit, the channels that are associated with the external channel by means of the <channel to channel connection> are modified by extending the <channel endpoint> that mentions **env** with a **via** <gate> part for the implicit gate.

5.7 Behaviour

5.7.1 Start

<start> ::=

start [<virtuality>] [<state entry point name>] <end> <transition>

If <state entry point name> is given in a <start>, the <start> shall be the <start> of a <composite state definition>.

5.7.2 State

<state> ::=

```
state <state list> <end>
        {       <input part>
        |       <priority input>
        |       <save part>
        |       <spontaneous transition>
        |       <continuous signal>
        |       <connect part> }*
        [ <state timer part> ]
[ endstate [<state name>] <end> ]
```

<state timer part> ::=

input <virtuality> <state timer> <end> <transition>

It is allowed to specify the optional <state name> ending a <state> only if the <state list> in the <state> consists of a single <state name>, in which case it shall be that <state name>.

The <connect part> is only allowed for a <state> with <state list> that contains a <typebased composite state> or <composite state list item>.

5.7.3 Input

<input part> ::=

input [<virtuality>] <input list> <end>
[<enabling condition>] <transition>

5.7.4 Priority input

<priority input> ::=

priority input [<virtuality>]
<priority input list> <end> <transition>

5.7.5 Continuous signal

<continuous signal> ::=

provided [<virtuality>]
<continuous expression> <end>
[**priority** <priority name> <end>] <transition>

5.7.6 Enabling condition

<enabling condition> ::=

provided <provided expression> <end>

Documents
Document Sharing Hub
Full document details: ITU-T Z.106 2021

5.7.7 Save

<save part> ::=
save [<virtuality>] <save list> <end>

5.7.8 Spontaneous transition

<spontaneous transition> ::=
input [<virtuality>] <spontaneous designator> <end>
[<enabling condition>] <transition>

5.7.9 Label

<label> ::=
<connector name>:

NOTE – In the abstract grammar, only free actions have labels; labels inside of a transition are transformed into separate free actions.

<free action> ::=
connection
<transition>
[**endconnection** [<label>] <end>]

If the <transition string> of the <transition> in <free action> is non-empty, the first <action> shall have a <label>; otherwise the <terminator> shall have a <label>. If present, the <label> ending the <free action> shall be the same as this <label>.

Model

If a <label> is not the first label of a <transition string>, the <transition string> is split into two parts. All <action>s preceding the <label> are preserved in the original transition, which is terminated with a <join> to the <label>. All action statements following <label> are copied to a new <free action>, which starts with the <label>. If the <transition string> is followed by a <terminator> with a label, the new <free action> is terminated with a <join> to the label of the <terminator> and another <free action> is made that contains the labelled <terminator>. If the <transition string> is followed by a <terminator> without a label, the new <free action> is terminated with the unlabelled <terminator>. If a <transition string> without any labels is followed by a labelled <terminator>, a <free action> is made that contains the labelled <terminator> and the <terminator> is replaced by a <join> to the label of the <terminator>.

5.7.10 State machine and composite state

<composite state definition> ::=
<composite state graph> | <state aggregation>

5.7.10.1 Composite state graph

<composite state graph> ::=
{<package use clause>}*
<composite state heading> <end> <composite state structure>
endsubstructure [[<qualifier>] <composite state name>] <end>

<composite state heading> ::=
<virtuality> **state** [**substructure**] [<qualifier>] <state name>
[<specialization>] [<agent formal parameters>]

The optional keyword **substructure** is added to <composite state heading> to match the keyword **endsubstructure** needed to close the <composite state graph>.

If there is a <composite state name> after **endsubstructure**, this shall be the same as the <state name> in the <composite state heading>. If there is a <qualifier> after **endsubstructure**, this shall identify the qualifier of the identifier for the composite state.

Documents
Document Sharing Hub
Full document details: ITU-T Z.106 2021

<composite state structure> ::=
{ <state connection points> | <entity in composite state> }*
<composite state body>

<entity in composite state> ::=
<valid input signal set>
| <variable definition>
| <data definition>
| <select definition>
| <macro definition>
| <procedure definition>
| <procedure reference>
| <composite state definition>
| <composite state reference>
| <composite state type definition>
| <composite state type reference>
| <gate in definition>

An <entity in composite state> that is a <valid input signal set> is only valid if the corresponding *Composite-state-type-definition* is used for *State-machine* items of agent types. A <composite state reference> in an <aggregation structure> is never an <entity in composite state>, but is always a <state partition> and therefore ends the list of items (<state connection points> or <entity in composite state>) before the <state aggregation body>.

<composite state body> ::=
<start>* { <state> | <free action> }*

5.7.10.2 State aggregation

<state aggregation> ::=
{<package use clause>}*
<state aggregation heading> <end> <aggregation structure>
endsubstructure [[<qualifier>] <composite state name>] <end>

<state aggregation heading> ::=
<virtuality> **state aggregation** [**substructure**] [<qualifier>] <state name>
[<specialization>] [<agent formal parameters>]

The optional keyword **substructure** is added to <state aggregation heading> to match the keyword **endsubstructure** needed to close the <composite state graph>.

If there is a <composite state name> after **endsubstructure**, this shall be the same as the <state name> in the <state aggregation heading>. If there is a <qualifier> after **endsubstructure**, this shall identify the qualifier of the identifier for the composite state.

<aggregation structure> ::=
{ <state connection points> | <entity in state aggregation> }*
<state aggregation body>

<entity in state aggregation> ::=
<valid input signal set>
| <variable definition>
| <data definition>
| <select definition>
| <macro definition>
| <procedure definition>
| <procedure reference>
| <composite state type definition>
| <composite state type reference>
| <gate in definition>

NOTE 1 – An <entity in state aggregation> differs from an <entity in composite state> by excluding <composite state definition> and <composite state reference>, which in <aggregation structure> are <state partition> alternatives.

Documents
Document Sharing Hub
Full document details: ITU-T Z.106 2021

<state aggregation body> ::=
{ <state partition>
| <state partition connection> }*

<state partition> ::=
<textual typebased state partition def>
| <composite state reference>
| <composite state definition>

NOTE 2 – No notation is needed for inherited state partitions, because any additional connection to the gates of the subtype state partition is given by a <channel definition> with a <channel path> where the <channel endpoint> identifies the state partition.

<state partition connection> ::=
<state partition connection entry> | <state partition connection exit>

<state partition connection entry> ::=
connect [<composite state identifier> **via**] <outer entry points>
and <composite state identifier> **via** <inner entry point> <end>

<state partition connection exit> ::=
connect [<composite state identifier> **via**] <outer exit point>
and <composite state identifier> **via** <inner exit points> <end> }

In <state partition connection entry> and <state partition connection exit>, the optional <composite state identifier> after the keyword **connect** and the following keyword **via** has no SDL-2010 meaning. It is legacy syntax. However, if present the <composite state identifier> shall identify the state aggregation.

5.7.10.3 State connection point

<state connection points> ::=
{ **in** <state entry points> | **out** <state exit points> } <end>

5.7.10.4 Connect

<connect part> ::=
connect [<virtuality>] [<connect list>] <end>
<exit transition>

<exit transition> ::=
<transition>

5.7.11 Transition

5.7.11.1 Transition body

<transition> ::=
<transition action items>
| <terminator>

<transition action items> ::=
<transition string> [<terminator>]

<transition string> ::=
<action>+

<action> ::=
[<label>] <action item> <end>

Documents
Document Sharing Hub
Full document details: ITU-T Z.106 2021