International Telecommunication Union

ITU-T

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

Z.107

(10/2019)

SERIES Z: LANGUAGES AND GENERAL SOFTWARE ASPECTS FOR TELECOMMUNICATION SYSTEMS

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

Specification and Description Language – Object-oriented data in SDL-2010

Recommendation ITU-T Z.107

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.107

Specification and Description Language – Object-oriented data in SDL-2010

Summary

Recommendation ITU-T Z.107 defines the object-oriented data features of the Specification and Description Language building on the foundation of the data definitions and expressions defined in Recommendation ITU-T Z.104. Together with Recommendations 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.106, this Recommendation is part of a reference manual for the language. The language defined in this Recommendation partially overlaps features of the language included in Basic SDL-2010 in Recommendation ITU-T Z.101 and used in Comprehensive SDL-2010 in Recommendation ITU-T Z.102, and the features of Recommendations ITU-T Z.103 and ITU-T Z.104.

Coverage

The Specification and Description Language has concepts for behaviour, data description and (particularly for larger systems) structuring. The basis of behaviour description is extended finite state machines communicating by messages. Data description is based on data types for values. The basis for structuring is hierarchical decomposition and type hierarchies. Though a distinctive feature of the Specification and Description Language is the graphical representation, the data and expression language is textual. Features to encode and decode data communicated by channels, define data types with values and operations, variables (including parameters) based on data types and expression actions that use the data types have been defined in previous Recommendations. This Recommendation covers the features of the language used to provide object-oriented, hierarchical abstraction to allow the representation of larger and data-intensive systems. This Recommendation does not always provide a canonical syntax, but by applying the *Model* descriptions given, a specification can be transformed to Basic SDL-2010 defined in Recommendation ITU-T Z.101, except in those cases where additional abstract syntax is added in this Recommendation.

Applications

The Specification and Description Language is applicable within standards bodies and industry. The main application areas for which the Specification and Description Language has been designed are stated in Recommendation ITU-T Z.100, but the language is generally suitable for describing reactive systems. The range of applications is from requirement description to implementation. The features of the language defined in Recommendation ITU-T Z.107 are essential for scalable description of data-intensive systems using object-oriented abstractions.

History

Edition	Recommendation	Approval	Study Group	Unique ID*
1.0	ITU-T Z.107	2012-04-29	17	11.1002/1000/11595
2.0	ITU-T Z.107	2016-04-29	17	11.1002/1000/12861
3.0	ITU-T Z.107	2019-10-14	17	11.1002/1000/14058

Keywords

Object-oriented data, SDL-2010, Specification and Description Language.

---

* 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.

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/.

Table of Contents

Introduction

Status/Stability

This Recommendation is part of the ITU-T Z.100 to ITU-T Z.107 series of Recommendations that give the complete language reference manual for SDL-2010. The text of this Recommendation is stable. For more details see Recommendation ITU-T Z.100.

Recommendation ITU-T Z.107

Specification and Description Language – Object-oriented data in SDL-2010

1 Scope

This Recommendation defines the object-oriented data features of the Specification and Description Language. The features defined in this document include: polymorphic object-oriented data including virtual methods, multiple inheritance of abstract types, run-time type identification and type coercion, as well as accessing data values by references.

Without these features the language would afford limited scalability and be unsuitable for the definition of real-time embedded software applications. For this reason, specifications in the Specification and Description Language use ITU-T Z.107 features. Together with [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.106], this Recommendation forms a reference manual for the language.

1.1 Objective

The objective of this Recommendation is to fully define the object-oriented data related features of the Specification and Description Language. Some of the ITU-T Z.107 material is also in [ITU-T Z.100], [ITU-T Z.101] and [ITU-T Z.104]. These definitions should be entirely consistent. The descriptions in [ITU-T Z.100] and [ITU-T Z.101] are overviews to enable data to be used with other features of SDL-2010. The ITU-T Z.104 definition is intended to be complete (with the exception of definition of data types in ASN.1 modules – see [ITU-T Z.105]). The ITU-T Z.107 definition extends the language with object-oriented data.

1.2 Application

This Recommendation is part of the reference manual for the Specification and Description Language.

The use of object-oriented data is fundamental to the description of scalable and efficient real-time embedded applications. Thus, it is almost certain that some features defined in Recommendation ITU-T Z.107 will appear in any specification of applications in SDL-2010.

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 T.50] Recommendation ITU-T T.50 (1992), *International Reference Alphabet (IRA) (Formerly International Alphabet No. 5 or IA5) – Information technology – 7-bit coded character set for information interchange*.

[ITU-T Z.100] Recommendation ITU-T Z.100 (2019), *Specification and Description Language – Overview of SDL-2010.*

[ITU-T Z.101] Recommendation ITU-T Z.101 (2019), *Specification and Description Language – Basic SDL-2010.*

[ITU-T Z.102] Recommendation ITU-T Z.102 (2019), *Specification and Description Language – Comprehensive SDL-2010.*

[ITU-T Z.103] Recommendation ITU-T Z.103 (2019), *Specification and Description Language – Shorthand notation and annotation in SDL-2010.*

[ITU-T Z.104] Recommendation ITU-T Z.104 (2019), *Specification and Description Language – Data and action language in SDL-2010.*

[ITU-T Z.105] Recommendation ITU-T Z.105 (2019), *Specification and Description Language – SDL-2010 combined with ASN.1 modules.*

[ITU-T Z.106] Recommendation ITU-T Z.106 (2019), *Specification and Description Language – Common interchange format for SDL-2010.*

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

[ISO/IEC 10646] ISO/IEC 10646:2017, *Information technology – Universal Coded Character Set (UCS).*

3 Definitions

3.1 Terms defined elsewhere

This Recommendation uses the terms defined in [ITU-T Z.100], with the additions given in [ITU-T Z.101], [ITU-T Z.102], [ITU-T Z.103], [ITU-T Z.104], [ITU-T Z.105] and [ITU-T Z.106].

3.2 Terms defined in this Recommendation

None.

4 Abbreviations and acronyms

The abbreviations and acronyms given in [ITU-T Z.100] apply.

5 Conventions

The conventions defined in [ITU-T Z.100] apply, which includes the conventions defined in [ITU-T Z.111].

Where an abstract or concrete syntax rule is defined in this Recommendation with the same name as a rule in [ITU-T Z.101], [ITU-T Z.102], [ITU-T Z.103] or [ITU-T Z.104], the rule given here replaces the rule in [ITU-T Z.101], [ITU-T Z.102], [ITU-T Z.103] or [ITU-T Z.104]. Any *Abstract grammar* or *Concrete grammar* condition, *Semantics* and *Model* defined on a named rule in [ITU-T Z.101], [ITU-T Z.102], [ITU-T Z.103] or [ITU-T Z.104] apply to the redefined rule, unless specifically defined otherwise in this Recommendation.

Some numbered clauses are included so that this Recommendation has the general structure and numbering of [ITU-T Z.104], and in the text states "See [ITU-T Z.104]". In these cases the clause includes features defined for the correspondingly numbered clause (including any subordinate clauses if not also included in this Recommendation) in [ITU-T Z.101], [ITU-T Z.102], [ITU-T Z.103] or [ITU-T Z.104].

6 General rules

See [ITU-T Z.104].

7 Organization of specification and Description Language specifications

See [ITU-T Z.104].

8 Structural concepts

See [ITU-T Z.104].

9 Agents

See [ITU-T Z.104].

10 Communication

See [ITU-T Z.104].

11 Behaviour

See [ITU-T Z.104].

12 Data

The concept of data in SDL-2010 is extended from [ITU-T Z.104] by the support for object-oriented data.

12.1 Data definitions

Data definitions are extended from [ITU-T Z.104] to support object-oriented data and multiple inheritance of abstract types.

Abstract grammar

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

Concrete grammar

The *Null-literal-signature* is the unique element of any value sort denoted by `null`.

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

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

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

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

Semantics

Every value sort contains a unique named element, the *Null-literal-signature* that does not identify any value. An attempt to obtain a value from the *Null-literal-signature* raises the predefined exception `InvalidReference`, except in the case the access is the *First-operand* or *Second-operand* of an *Equality-expression*, or the access is the *Expression* of an *Assignment*.

The *Null-literal-signature* of a *Value-data-type-definition* is the signature for the null literal operator.

12.1.1 Data type definition

See [ITU-T Z.104].

12.1.2 Interface definition

See [ITU-T Z.104].

12.1.3 Operation signature

[ITU-T Z.104] is extended to support virtual methods.

Abstract grammar

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

Concrete grammar

A virtual method is a method of a data type having **virtual** or **redefined** as <virtuality>. A method that has **finalized** as <virtuality>, or that does not have <virtuality> is not a virtual method. A redefined method is a method having **redefined** or **finalized** as <virtuality>. Redefinition is only allowed for virtual methods. Every redefined method shall be directly or indirectly (via another redefined method) a redefinition of a virtual method that is not redefined (that is, with <virtuality>**virtual**).

If the derived type contains only an <operation signature> but no <operation definition>, <operation reference>, or <external operation definition> for the redefined method, then only the signature of the redefined method is changed.

When a method is redefined in a specialization, its signature shall be sort compatible with the corresponding signature in the base type, and further, if the *Result* in the *Operation-signature* denotes a sort A, then the *Result* of the redefined method shall only denote a sort B such that B is sort compatible with A.

A redefinition of a virtual method shall not change the <parameter kind> in any <argument> of the inherited <operation signature>. A redefinition of a virtual method shall not change the <aggregation kind> of the <operation result> of the inherited <operation definition>.

Concrete grammar

The <operation preamble> defined in [ITU-T Z.104] is extended to include an optional <virtuality> to define a method to be a virtual method.

<operation preamble> ::=
[<visibility> [<virtuality>]] | [<virtuality> [<visibility>]]

A <virtuality> item is only allowed in an <operation preamble> of an <operation signature> that is a method. If <virtuality> is present in an <operation signature>, none of the <argument> items of the <operation signature> shall contain a <sort> that is an <anchored sort>.

NOTE – Because of the above constraints, it is not possible to define co-variantly redefined methods.

An <operation signature> with <virtuality> **virtual** or **redefined** in the <operation preamble> represents a *Dynamic-operation-signature*. An <operation signature> with <virtuality> **finalized** in the <operation preamble> represents a *Static-operation-signature*.

Semantics

Virtual methods do not have a <virtuality constraint> which, in this case only, does not limit redefinition.

12.1.4 Generic data type operations

An operation is added to the generic data type operations defined in [ITU-T Z.101].

Concrete grammar

For a value data type with the <data type name>`S`, there is an item in the *Operation-signature-**set*** items equivalent to including the following explicit <operation signature> definition in the <operator list> of its <operation signatures>:

```
Null    -> S;
```

where the result corresponds to an *Aggregation-kind* of **REF**.

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

Semantics

For any sort, the operator `Null` returns the *Null-literal-signature* of that sort.

12.1.5 Pid and pid sorts

See [ITU-T Z.104].

12.1.6 Data type constructors

12.1.6.1 Literals

See [ITU-T Z.104].

12.1.6.2 Structure data types

See [ITU-T Z.104].

12.1.6.3 Choice data types

See [ITU-T Z.104].

12.1.7 Behaviour of operations

The specification of the behaviour of operations is extended from [ITU-T Z.104] because <virtuality> is allowed for methods.

Concrete grammar

A <virtuality> item is only allowed in an <operation preamble> of an <operation definition> that is a method. If <virtuality> is present in an <operation definition>, none of the <formal operation parameters> items of the <operation definition> shall contain a <sort> that is an <anchored sort>.

12.1.8 Additional data definition constructs

This clause covers further constructs for data.

12.1.8.1 Restricted visibility

See [ITU-T Z.104].

12.1.8.2 Syntypes

See [ITU-T Z.104].

12.1.8.3 Constraint

See [ITU-T Z.104].

12.1.8.4 Synonym definition

See [ITU-T Z.104].

12.1.9 Specialization of data types

Specialization of data types is extended from [ITU-T Z.104] to support object-oriented data.

Concrete grammar

<data type specialization> ::=
inherits<data type type expression> { **,** < data type type expression> }*
[<renaming> | <legacy data inheritance>] [**adding**]

At most one of the <data type type expression> is allowed to be not abstract.

NOTE – In other words, multiple inheritance is allowed only for abstract data types.

Semantics

Direct sort compatibility is extended by an additional clause (c):

Let Y and Z be different sorts. Y is directly sort compatible with Z if and only if any of the following applies:

a) Y is denoted by an <anchored sort> of the form **this** Z;

b) Y is denoted by a <pid sort> (see clause 12.1.2 of [ITU-T Z.101] and clause 12.1.2 of [ITU-T Z.104]) and Z is a supersort of Y;

c) Y is denoted by a <basic sort> and Z is a supersort of Y.

12.2 Use of data

The following clauses define how sorts, literals, operators, methods and synonyms are interpreted in expressions.

12.2.1 Expressions

Expressions are extended from [ITU-T Z.104] support object-oriented data. Due to the possibility of polymorphic assignment (see clause 12.3.3), the dynamic sort of an expression is permitted to differ from the (static) sort of an expression.

Abstract grammar

Constant-expression	::	*Literal*
	|	*Conditional-expression*
	|	*Equality-expression*
	|	*Operation-application*
	|	*Range-check-expression*
	|	*Agent-instance-pid-value*
	|	*Type-check-expression*
	|	*Type-coercion*

Active-expression = *Variable-access*
| *Conditional-expression*
| *Operation-application*
| *Equality-expression*
| *Imperative-expression*
| *Range-check-expression*
| *Value-returning-call-node*
| *Encoding-expression*
| *Decoding-expression*
| *Agent-instance-pid-value*
| *Type-check-expression*
| *Type-coercion*

The alternatives *Type-check-expression* and *Type-coercion* (described in clause 12.2.7 and clause 12.2.9 respectively) are added to both *Constant-expression* and *Active-expression* defined in [ITU-T Z.104].

Concrete grammar

<expression> ::=
<expression0>
| <range check expression>
| <type coercion>

The <expression> defined in [ITU-T Z.101] is extended to include <type coercion> (see clause 12.2.9) as an alternative.

Semantics

The dynamic sort of an expression is the sort of the result of the expression. The static and dynamic sort of active expressions are permitted to differ due to polymorphic assignments (see clause 12.3.3). For a constant expression, the dynamic sort of an expression is its static sort.

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

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

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

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

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

12.2.2 Literal

See [ITU-T Z.104].

12.2.3 Extended primary

See [ITU-T Z.104].

12.2.4 Equality expression

Equality expression as defined in Basic SDL-2010 is extended for aggregation kind **REF**.

Semantics

If, after interpretation, both of the operands are values and the aggregation kind of both of the operands is **REF**, the *Equality-expression* denotes equality of identities. The positive *Equality-*

expression returns the predefined `Boolean` value `true` if both operands are "undefined" or `null` (that is, either they access a variable which has no value associated or are associated with the *Null-literal-signature*), and predefined `Boolean` value `false` if only one operand is "undefined" or is associated with the *Null-literal-signature*. The positive *Equality-expression* returns the predefined `Boolean` value `true` if both operands have the same data item associated, and the predefined `Boolean` value `false` otherwise.

12.2.5 Conditional expression

Conditional expression is extended from Basic SDL-2010 as follows.

Semantics

The dynamic sort of the conditional expression is the dynamic sort of the result of interpreting the conditional expression.

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

12.2.6 Operation application

Operation application is extended from Basic SDL-2010 and [ITU-T Z.104] to include the semantics of the interpretation of *Dynamic-operation-signature* items.

Semantics

If an *Operation-application* has an *Operation-identifier* that identifies a *Dynamic-operation-signature*, the *Operation-application* is interpreted as a *Value-returning-call-node* by invoking the *Procedure-definition* identified by the following steps:

a) the *Actual-parameters* are interpreted

b) if a *Procedure-identifier* exists in the *Dynamic-operation-signature* of the sort identified by the first *Argument*, then this *Procedure-identifier* identifies the *Procedure-definition*

c) otherwise, the sort identified by the parent sort of the first *Argument* is examined for a sort compatible *Dynamic-operation-signature* or *Static-operation-signature* with a *Procedure-identifier* and, if found, this *Procedure-identifier* identifies the *Procedure-definition*

d) otherwise, the search for a *Procedure-identifier* continues with the sort identified by the parent sort of the previously examined sort, if any

e) if no such *Procedure-identifier* is found, the operation application raises the predefined exception `InvalidCall`.

An *Operation-application* has a dynamic sort, which is the dynamic sort of the result obtained by the interpretation of the procedure.

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

12.2.7 Range check expression

Range check expression is extended from Basic SDL-2010 to apply to object-oriented data.

Abstract grammar

Type-check-expression :: *Expression Parent-sort-identifier*

Concrete grammar

<range check expression> ::=
<operand2> **in type** { <range check constrained sort>
| <syntype>
| <pid sort>
| <datatype type expression> }

If the form <datatype type expression> is used, the <range check expression> represents a *Type-check-expression*. The <operand2> represents the *Expression*. The <datatype type expression> determines the *Parent-sort-identifier*.

Semantics

The *Type-check-expression* has the predefined `Boolean` value `true` if and only if the dynamic sort of the *Expression* is sort compatible with the sort identified by the *Parent-sort-identifier*; otherwise, it has the predefined `Boolean` value `false`.

12.2.8 Synonym

See [ITU-T Z.104].

12.2.9 Type coercion

A type coercion allows changing the dynamic sort of an expression.

Abstract grammar

Type-coercion = *Expression*
Sort-reference-identifier

The static sort of *Expression* shall be the sort identified by *Sort-reference-identifier*, or a parent sort of that sort.

Concrete grammar

<type coercion> ::=
<expression> **as** <sort>

Semantics

If the sort identified by the *Sort-reference-identifier* is the dynamic sort of *Expression* or a parent sort of that sort, type-coercion changes the dynamic sort of *Expression* to the sort identified by *Sort-reference-identifier*. Otherwise, the predefined exception `InvalidSort` is returned.

12.3 Active use of data

This clause extends the active use of data of Basic SDL-2010 and [ITU-T Z.104].

12.3.1 Variable definition

Variable definition is extended from [ITU-T Z.104] to support object-oriented data.

Abstract grammar

Aggregation-kind = **PART** | **REF**

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

Semantics

The *Aggregation-kind* of a data holder (variables, procedure parameters, procedure results) determines the manner in which data referenced by the data holder are accessed or assigned.

Concrete grammar

<aggregation kind> ::=

[part | **ref]**

NOTE – The details of the impact of <aggregation kind> on the access or assignment to data holders are given in clause 12.3.2 and clause 12.3.3.

12.3.2 Variable access

Variable access is extended from [ITU-T Z.104] to account for its dynamic sort.

Semantics

A variable access has a dynamic sort, which is the dynamic sort of the data item associated with the identified variable.

If the variable is "undefined", the predefined exception `UndefinedVariable` is raised when the variable is accessed, except in the case where the *Variable-access* is the *First-operand* or *Second-operand* of an *Equality-expression*, or the *Variable-access* is the *Expression* of an *Assignment*.

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

12.3.3 Assignment

Assignment is extended from Basic SDL-2010 to account for aggregation kinds other than **PART**.

Abstract grammar

If the *Variable-identifier* refers to a *Variable-definition* with *Aggregation-kind* **REF** and the *Expression* has aggregation kind **PART**, the *Expression* shall be a *Variable-identifier* or an *Operation-application* which accesses a field of a data type defined in a <structure definition> or <choice definition>, as introduced in clause 12.1.6.2 and clause 12.1.6.3, respectively. If the *Variable-identifier* refers to a *Variable-definition* with *Aggregation-kind* **PART**, the *Expression* shall not be the *Null-literal-signature*.

Semantics

Where the association between the variable and the result of the *Expression* is established, the manner is extended from [ITU-T Z.101] to:

a) If the sort of the variable is the *Sort* of a *Value-data-type-definition*, then

   1) if the *Aggregation-kind* of variable is **PART**, a copy of the value returned by the result of the *Expression* is associated with the identified variable;

   2) if the *Aggregation-kind* of variable is **REF**, and the result of the *Expression* is not "undefined" (it does not access a variable that has no value associated), the value returned by the result of the *Expression* is associated with the identified variable. If the *Expression* is a *Variable-identifier* with aggregation kind **PART**, the variable is associated with the *Variable-identifier* of the *Expression*. If the *Expression* is a *Variable-identifier* with aggregation kind **REF**, the variable is associated with the (reference) value associated with the *Variable-identifier* of the *Expression*. If the *Expression* is a field access *Operation-application*, the variable is associated with a reference to the data item enclosing the field (for example a *Variable-identifier*) and the name of the field. If the *Expression* is a *Literal* that identifies the *Null-literal-signature* of a *Value-data-type-definition*, the variable is associated with the *Null-literal-*

signature. Otherwise, if the result of the *Expression* is "undefined", the variable is "undefined".

b) If the sort of the variable is the `Pid` sort or a pid sort (the *Sort* of an *Interface-definition*) and the result of the *Expression* is a pid (it identifies an agent instance), the *Variable-identifier* is associated with the pid that is the result of *Expression*.

12.3.4 Imperative expression

See [ITU-T Z.104].

12.3.5 Value returning procedure call

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

Semantics

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

SERIES OF ITU-T RECOMMENDATIONS

Series A	Organization of the work of ITU-T
Series D	Tariff and accounting principles and international telecommunication/ICT economic and policy issues
Series E	Overall network operation, telephone service, service operation and human factors
Series F	Non-telephone telecommunication services
Series G	Transmission systems and media, digital systems and networks
Series H	Audiovisual and multimedia systems
Series I	Integrated services digital network
Series J	Cable networks and transmission of television, sound programme and other multimedia signals
Series K	Protection against interference
Series L	Environment and ICTs, climate change, e-waste, energy efficiency; construction, installation and protection of cables and other elements of outside plant
Series M	Telecommunication management, including TMN and network maintenance
Series N	Maintenance: international sound programme and television transmission circuits
Series O	Specifications of measuring equipment
Series P	Telephone transmission quality, telephone installations, local line networks
Series Q	Switching and signalling, and associated measurements and tests
Series R	Telegraph transmission
Series S	Telegraph services terminal equipment
Series T	Terminals for telematic services
Series U	Telegraph switching
Series V	Data communication over the telephone network
Series X	Data networks, open system communications and security
Series Y	Global information infrastructure, Internet protocol aspects, next-generation networks, Internet of Things and smart cities
Series Z	**Languages and general software aspects for telecommunication systems**

Printed in Switzerland
Geneva, 2019