



DPC: 25/30482256 DC

BSI Group Headquarters

389 Chiswick High Road London W4 4AL

Tel: +44 (0)20 8996 9000

Fax: +44 (0)20 8996 7400

www.bsigroup.com

Date: 22 July 2025

Origin: International

Latest date for receipt of comments: 14 September 2025

Project No. 2023/02583

Responsible committee: ICT/1 Information systems co-ordination

Interested committees:

Title: Draft BS ISO/IEC 26300-4 Information technology — Open Document Format for Office Applications (OpenDocument) v1.3. Part 4: Recalculated Formula (OpenFormula) Format.

Please notify admin.start@bsigroup.com if you are aware of any keywords that might assist in clarifying or identifying the standard or of the content of this standard:

- i) has any issues related to 3rd party IPR, patent or copyright
- ii) affects other national standard(s)
- iii) requires additional national guidance or information

WARNING: THIS IS A DRAFT AND MUST NOT BE REGARDED OR USED AS A BRITISH STANDARD. THIS DRAFT IS NOT CURRENT BEYOND 14 September 2025

This draft is issued to allow comments from interested parties; all comments will be given consideration prior to publication. No acknowledgement will normally be sent. **See overleaf for information on the submission of comments.**

No copying is allowed, in any form, without prior written permission from BSI except as permitted under the Copyright, Designs and Patent Act 1988 or for circulation within a nominating organization for briefing purposes. Electronic circulation is limited to dissemination by e-mail within such an organization by committee members.

Further copies of this draft may be purchased from BSI Shop <http://shop.bsigroup.com> or from BSI Customer Services, Tel: +44(0) 20 8996 9001 or email cservices@bsigroup.com. British, International and foreign standards are also available from BSI Customer Services.

Information on the co-operating organizations represented on the committees referenced above may be obtained from <https://standardsdevelopment.bsigroup.com/>.

Introduction

This draft standard is based on national and international discussions. Your comments on this draft are invited and will assist in the preparation of the consequent standard.

For international standards, comments will be reviewed by the relevant UK national committee before sending the consensus UK vote and comments to the international committee, which will then decide appropriate action. If the international standard is approved, it is usual for the text to be published as a British Standard.

For national standards, comments will be reviewed by the relevant UK national committee and the resulting standards published as a British Standard.

UK Vote

Please indicate whether you consider the UK should submit a negative (with supporting technical reasons) or positive vote on this draft. Please indicate if you are aware of any reason why this draft standard should not be published as a British Standard.

Submission of Comments

- **Annotated drafts are not acceptable and will be rejected.**
- All comments should be submitted online at <https://standardsdevelopment.bsigroup.com/>. You will need to register in order to comment.

Template for comments and secretariat observations

Date: xx/xx/20xx	Document: ISO/DIS xxxx
------------------	------------------------

1	2	(3)	4	5	(6)	7
MB	Clause No./ Subclause No./Annex (e.g. 3.1)	Paragraph/ Figure/Table/Note	Type of comment	Comment (justification for change) by the MB	Proposed change by the MB	Secretariat observations on each comment submitted
	3.1	Definition 1	ed	Definition is ambiguous and needs clarifying.	Amend to read '...so that the mains connector to which no connection...'	
	6.4	Paragraph 2	te	The use of the UV photometer as an alternative cannot be supported as serious problems have been encountered in its use in the UK.	Delete reference to UV photometer.	



DRAFT International Standard

ISO/IEC DIS 26300-4

Information technology — Open Document Format for Office Applications (OpenDocument) v1.3 —

Part 4: Recalculated Formula (OpenFormula) Format

ICS: 35.240.30

ISO/IEC JTC 1

Secretariat: **ANSI**

Voting begins on:
2025-07-22

Voting terminates on:
2025-10-14

This document is circulated as received from the committee secretariat.

THIS DOCUMENT IS A DRAFT CIRCULATED FOR COMMENTS AND APPROVAL. IT IS THEREFORE SUBJECT TO CHANGE AND MAY NOT BE REFERRED TO AS AN INTERNATIONAL STANDARD UNTIL PUBLISHED AS SUCH.

IN ADDITION TO THEIR EVALUATION AS BEING ACCEPTABLE FOR INDUSTRIAL, TECHNOLOGICAL, COMMERCIAL AND USER PURPOSES, DRAFT INTERNATIONAL STANDARDS MAY ON OCCASION HAVE TO BE CONSIDERED IN THE LIGHT OF THEIR POTENTIAL TO BECOME STANDARDS TO WHICH REFERENCE MAY BE MADE IN NATIONAL REGULATIONS.

RECIPIENTS OF THIS DRAFT ARE INVITED TO SUBMIT, WITH THEIR COMMENTS, NOTIFICATION OF ANY RELEVANT PATENT RIGHTS OF WHICH THEY ARE AWARE AND TO PROVIDE SUPPORTING DOCUMENTATION.

Reference number
ISO/IEC DIS 26300-4:2025(en)

Full document details: 25/30482256 DC

© ISO/IEC 2025  **Documents**
Document Sharing Hub



COPYRIGHT PROTECTED DOCUMENT

© ISO/IEC 2025

All rights reserved. Unless otherwise specified, or required in the context of its implementation, no part of this publication may be reproduced or utilized otherwise in any form or by any means, electronic or mechanical, including photocopying, or posting on the internet or an intranet, without prior written permission. Permission can be requested from either ISO at the address below or ISO's member body in the country of the requester.

ISO copyright office
CP 401 • Ch. de Blandonnet 8
CH-1214 Vernier, Geneva
Phone: +41 22 749 01 11
Email: copyright@iso.org
Website: www.iso.org

Published in Switzerland

Notices

Copyright © OASIS Open 2021. All Rights Reserved.

All capitalized terms in the following text have the meanings assigned to them in the OASIS Intellectual Property Rights Policy (the "OASIS IPR Policy"). The full [Policy](#) may be found at the OASIS website.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published, and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this section are included on all such copies and derivative works. However, this document itself may not be modified in any way, including by removing the copyright notice or references to OASIS, except as needed for the purpose of developing any document or deliverable produced by an OASIS Technical Committee (in which case the rules applicable to copyrights, as set forth in the OASIS IPR Policy, must be followed) or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by OASIS or its successors or assigns.

This document and the information contained herein is provided on an "AS IS" basis and OASIS DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY OWNERSHIP RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

OASIS requests that any OASIS Party or any other party that believes it has patent claims that would necessarily be infringed by implementations of this OASIS Committee Specification or OASIS Standard, to notify OASIS TC Administrator and provide an indication of its willingness to grant patent licenses to such patent claims in a manner consistent with the IPR Mode of the OASIS Technical Committee that produced this specification.

OASIS invites any party to contact the OASIS TC Administrator if it is aware of a claim of ownership of any patent claims that would necessarily be infringed by implementations of this specification by a patent holder that is not willing to provide a license to such patent claims in a manner consistent with the IPR Mode of the OASIS Technical Committee that produced this specification. OASIS may include such claims on its website, but disclaims any obligation to do so.

OASIS takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Information on OASIS' procedures with respect to rights in any document or deliverable produced by an OASIS Technical Committee can be found on the OASIS website. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this OASIS Committee Specification or OASIS Standard, can be obtained from the OASIS TC Administrator. OASIS makes no representation that any information or list of intellectual property rights will at any time be complete, or that any claims in such list are, in fact, Essential Claims.

The name "OASIS" is a trademark of [OASIS](#), the owner and developer of this specification, and should be used only to refer to the organization and its official outputs. OASIS welcomes reference to, and implementation and use of, specifications, while reserving the right to enforce its marks against misleading uses. Please see <https://www.oasis-open.org/policies-guidelines/trademark> for above guidance.

Table of Contents

1	Introduction.....	19
1.1	Introduction.....	19
1.2	Terminology.....	19
1.3	Purpose.....	19
1.4	Normative References.....	19
1.5	Non-Normative References.....	20
2	Expressions and Evaluators.....	21
2.1	Introduction.....	21
2.2	OpenDocument Formula Expression.....	21
2.3	Evaluator Conformance.....	21
2.3.1	OpenDocument Formula Evaluator.....	21
2.3.2	OpenDocument Formula Small Group Evaluator.....	21
2.3.3	OpenDocument Formula Medium Group Evaluator.....	22
2.3.4	OpenDocument Formula Large Group Evaluator.....	23
2.4	Variances (Implementation-defined, Unspecified, and Behavioral Changes).....	23
3	Formula Processing Model.....	25
3.1	General.....	25
3.2	Expression Evaluation.....	25
3.2.1	General.....	25
3.2.2	Expression Calculation.....	25
3.2.3	Operator and Function Evaluation.....	25
3.3	Non-Scalar Evaluation (aka 'Array expressions').....	26
3.4	Host-Defined Behaviors.....	28
3.5	When recalculation occurs.....	28
3.6	Numerical Models.....	29
3.7	Basic Limits.....	29
4	Types.....	30
4.1	General.....	30
4.2	Text (String).....	30
4.3	Number.....	30
4.3.1	General.....	30
4.3.2	Time.....	30
4.3.3	Date.....	31
4.3.4	DateTime.....	31
4.3.5	Percentage.....	31
4.3.6	Currency.....	31

4.3.7 Logical (Number).....	31
4.4 Complex Number.....	31
4.5 Logical (Boolean).....	32
4.6 Error.....	32
4.7 Empty Cell.....	32
4.8 Reference.....	32
4.9 ReferenceList.....	32
4.10 Array.....	33
4.11 Pseudotypes.....	33
4.11.1 General.....	33
4.11.2 Scalar.....	33
4.11.3 DateParam.....	33
4.11.4 TimeParam.....	33
4.11.5 Integer.....	33
4.11.6 TextOrNumber.....	33
4.11.7 Basis.....	33
4.11.8 Criterion.....	35
4.11.9 Database.....	36
4.11.10 Field.....	36
4.11.11 Criteria.....	36
4.11.12 Sequences (NumberSequence, NumberSequenceList, DateSequence, LogicalSequence, and ComplexSequence).....	36
4.11.13 Any.....	37
5 Expression Syntax.....	38
5.1 General.....	38
5.2 Basic Expressions.....	38
5.3 Constant Numbers.....	38
5.4 Constant Strings.....	39
5.5 Operators.....	39
5.6 Functions and Function Parameters.....	40
5.7 Nonstandard Function Names.....	40
5.8 References.....	41
5.9 Reference List.....	42
5.10 Quoted Label.....	42
5.10.1 General.....	42
5.10.2 Lookup of Defined Labels.....	42
5.10.3 Automatic Lookup of Labels.....	42
5.10.4 Implicit Intersection.....	43
5.10.5 Automatic Range.....	43

5.10.6	Automatic Intersection.....	44
5.11	Named Expressions.....	44
5.12	Constant Errors.....	45
5.13	Inline Arrays.....	46
5.14	Whitespace.....	46
6	Standard Operators and Functions.....	47
6.1	General.....	47
6.2	Common Template for Functions and Operators.....	47
6.3	Implicit Conversion Operators.....	48
6.3.1	General.....	48
6.3.2	Conversion to Scalar.....	48
6.3.3	Implied intersection.....	48
6.3.4	Force to array context (ForceArray).....	48
6.3.5	Conversion to Number.....	48
6.3.6	Conversion to Integer.....	49
6.3.7	Conversion to NumberSequence.....	49
6.3.8	Conversion to NumberSequenceList.....	49
6.3.9	Conversion to DateSequence.....	49
6.3.10	Conversion to Complex Number.....	49
6.3.11	Conversion to ComplexSequence.....	50
6.3.12	Conversion to Logical.....	50
6.3.13	Conversion to LogicalSequence.....	50
6.3.14	Conversion to Text.....	50
6.3.15	Conversion to DateParam.....	51
6.3.16	Conversion to TimeParam.....	51
6.4	Standard Operators.....	51
6.4.1	General.....	51
6.4.2	Infix Operator "+".....	51
6.4.3	Infix Operator "-".....	51
6.4.4	Infix Operator "*".....	51
6.4.5	Infix Operator "/".....	52
6.4.6	Infix Operator "^".....	52
6.4.7	Infix Operator "=".....	52
6.4.8	Infix Operator "<>".....	52
6.4.9	Infix Operator Ordered Comparison ("<", "<=", ">", ">=").....	53
6.4.10	Infix Operator "&".....	53
6.4.11	Infix Operator Reference Range (":").....	53
6.4.12	Infix Operator Reference Intersection ("!").....	54

6.4.13	Infix Operator Reference Concatenation ("~") (aka Union).....	54
6.4.14	Postfix Operator "%".....	54
6.4.15	Prefix Operator "+".....	55
6.4.16	Prefix Operator "-".....	55
6.5	Matrix Functions.....	55
6.5.1	General.....	55
6.5.2	MDETERM.....	55
6.5.3	MINVERSE.....	56
6.5.4	MMULT.....	56
6.5.5	MUNIT.....	56
6.5.6	TRANSPOSE.....	57
6.6	Bit operation functions.....	57
6.6.1	General.....	57
6.6.2	BITAND.....	57
6.6.3	BITLSHIFT.....	57
6.6.4	BITOR.....	58
6.6.5	BITRSHIFT.....	58
6.6.6	BITXOR.....	58
6.7	Byte-position text functions.....	58
6.7.1	General.....	58
6.7.2	FINDB.....	58
6.7.3	LEFTB.....	59
6.7.4	LENB.....	59
6.7.5	MIDB.....	59
6.7.6	REPLACEB.....	59
6.7.7	RIGHTB.....	59
6.7.8	SEARCHB.....	60
6.8	Complex Number Functions.....	60
6.8.1	General.....	60
6.8.2	COMPLEX.....	60
6.8.3	IMABS.....	60
6.8.4	IMAGINARY.....	60
6.8.5	IMARGUMENT.....	60
6.8.6	IMCONJUGATE.....	61
6.8.7	IMCOS.....	61
6.8.8	IMCOSH.....	61
6.8.9	IMCOT.....	61
6.8.10	IMCSC.....	61

6.8.11	IMCSCH.....	62
6.8.12	IMDIV.....	62
6.8.13	IMEXP.....	62
6.8.14	IMLN.....	62
6.8.15	IMLOG10.....	63
6.8.16	IMLOG2.....	63
6.8.17	IMPOWER.....	63
6.8.18	IMPRODUCT.....	63
6.8.19	IMREAL.....	63
6.8.20	IMSIN.....	64
6.8.21	IMSINH.....	64
6.8.22	IMSEC.....	64
6.8.23	IMSECH.....	64
6.8.24	IMSQRT.....	64
6.8.25	IMSUB.....	65
6.8.26	IMSUM.....	65
6.8.27	IMTAN.....	65
6.9	Database Functions.....	65
6.9.1	General.....	65
6.9.2	DAVERAGE.....	65
6.9.3	DCOUNT.....	66
6.9.4	DCOUNTA.....	66
6.9.5	DGET.....	66
6.9.6	DMAX.....	66
6.9.7	DMIN.....	66
6.9.8	DPRODUCT.....	67
6.9.9	DSTDEV.....	67
6.9.10	DSTDEVP.....	67
6.9.11	DSUM.....	67
6.9.12	DVAR.....	68
6.9.13	DVARP.....	68
6.10	Date and Time Functions.....	68
6.10.1	General.....	68
6.10.2	DATE.....	68
6.10.3	DATEDIF.....	68
6.10.4	DATEVALUE.....	69
6.10.5	DAY.....	69
6.10.6	DAYS.....	69

6.10.7	DAYS360.....	70
6.10.8	EDATE.....	70
6.10.9	EOMONTH.....	71
6.10.10	HOUR.....	71
6.10.11	ISOWEEKNUM.....	71
6.10.12	MINUTE.....	71
6.10.13	MONTH.....	72
6.10.14	NETWORKDAYS.....	72
6.10.15	NOW.....	72
6.10.16	SECOND.....	72
6.10.17	TIME.....	73
6.10.18	TIMEVALUE.....	73
6.10.19	TODAY.....	73
6.10.20	WEEKDAY.....	73
6.10.21	WEEKNUM.....	74
6.10.22	WORKDAY.....	75
6.10.23	YEAR.....	75
6.10.24	YEARFRAC.....	75
6.11	External Access Functions.....	76
6.11.1	General.....	76
6.11.2	DDE.....	76
6.11.3	HYPERLINK.....	76
6.12	Financial Functions.....	77
6.12.1	General.....	77
6.12.2	ACCRINT.....	77
6.12.3	ACCRINTM.....	78
6.12.4	AMORLINC.....	78
6.12.5	COUPDAYBS.....	79
6.12.6	COUPDAYS.....	79
6.12.7	COUPDAYSNC.....	80
6.12.8	COUPNCD.....	80
6.12.9	COUPNUM.....	81
6.12.10	COUPPCD.....	81
6.12.11	CUMIPMT.....	82
6.12.12	CUMPRINC.....	82
6.12.13	DB.....	83
6.12.14	DDB.....	84
6.12.15	DISC.....	85

6.12.16	DOLLARDE.....	86
6.12.17	DOLLARFR.....	86
6.12.18	DURATION.....	86
6.12.19	EFFECT.....	87
6.12.20	FV.....	87
6.12.21	FVSCHEDULE.....	87
6.12.22	INTRATE.....	88
6.12.23	IPMT.....	88
6.12.24	IRR.....	88
6.12.25	ISPMT.....	89
6.12.26	MDURATION.....	89
6.12.27	MIRR.....	90
6.12.28	NOMINAL.....	90
6.12.29	NPER.....	90
6.12.30	NPV.....	91
6.12.31	ODDFPRICE.....	91
6.12.32	ODDFYIELD.....	92
6.12.33	ODDLPRICE.....	92
6.12.34	ODDLYIELD.....	93
6.12.35	PDURATION.....	93
6.12.36	PMT.....	93
6.12.37	PPMT.....	94
6.12.38	PRICE.....	94
6.12.39	PRICEDISC.....	95
6.12.40	PRICEMAT.....	95
6.12.41	PV.....	96
6.12.42	RATE.....	96
6.12.43	RECEIVED.....	97
6.12.44	RRI.....	97
6.12.45	SLN.....	97
6.12.46	SYD.....	98
6.12.47	TBILLEQ.....	98
6.12.48	TBILLPRICE.....	98
6.12.49	TBILLYIELD.....	99
6.12.50	VDB.....	99
6.12.51	XIRR.....	100
6.12.52	XNPV.....	100
6.12.53	YIELD.....	101

6.12.54	YELDDISC.....	101
6.12.55	YELDMAT.....	102
6.13	Information Functions.....	102
6.13.1	General.....	102
6.13.2	AREAS.....	102
6.13.3	CELL.....	102
6.13.4	COLUMN.....	104
6.13.5	COLUMNS.....	104
6.13.6	COUNT.....	104
6.13.7	COUNTA.....	105
6.13.8	COUNTBLANK.....	105
6.13.9	COUNTIF.....	105
6.13.10	COUNTIFS.....	106
6.13.11	ERROR.TYPE.....	106
6.13.12	FORMULA.....	106
6.13.13	INFO.....	106
6.13.14	ISBLANK.....	107
6.13.15	ISERR.....	107
6.13.16	ISERROR.....	108
6.13.17	ISEVEN.....	108
6.13.18	ISFORMULA.....	108
6.13.19	ISLOGICAL.....	108
6.13.20	ISNA.....	109
6.13.21	ISNONTEXT.....	109
6.13.22	ISNUMBER.....	109
6.13.23	ISODD.....	109
6.13.24	ISREF.....	110
6.13.25	ISTEXT.....	110
6.13.26	N.....	110
6.13.27	NA.....	110
6.13.28	NUMBERVALUE.....	111
6.13.29	ROW.....	111
6.13.30	ROWS.....	111
6.13.31	SHEET.....	111
6.13.32	SHEETS.....	112
6.13.33	TYPE.....	112
6.13.34	VALUE.....	113
6.14	Lookup Functions.....	114

6.14.1	General.....	114
6.14.2	ADDRESS.....	114
6.14.3	CHOOSE.....	115
6.14.4	GETPIVOTDATA.....	115
6.14.5	HLOOKUP.....	116
6.14.6	INDEX.....	117
6.14.7	INDIRECT.....	117
6.14.8	LOOKUP.....	117
6.14.9	MATCH.....	118
6.14.10	MULTIPLE.OPERATIONS.....	119
6.14.11	OFFSET.....	120
6.14.12	VLOOKUP.....	121
6.15	Logical Functions.....	121
6.15.1	General.....	121
6.15.2	AND.....	121
6.15.3	FALSE.....	122
6.15.4	IF.....	122
6.15.5	IFERROR.....	123
6.15.6	IFNA.....	123
6.15.7	NOT.....	123
6.15.8	OR.....	123
6.15.9	TRUE.....	124
6.15.10	XOR.....	124
6.16	Mathematical Functions.....	124
6.16.1	General.....	124
6.16.2	ABS.....	124
6.16.3	ACOS.....	124
6.16.4	ACOSH.....	125
6.16.5	ACOT.....	125
6.16.6	ACOTH.....	125
6.16.7	ASIN.....	125
6.16.8	ASINH.....	126
6.16.9	ATAN.....	126
6.16.10	ATAN2.....	126
6.16.11	ATANH.....	126
6.16.12	BESSELI.....	127
6.16.13	BESSELJ.....	127
6.16.14	BESSELK.....	127

6.16.15	BESSELY.....	127
6.16.16	COMBIN.....	128
6.16.17	COMBINA.....	128
6.16.18	CONVERT.....	128
6.16.19	COS.....	135
6.16.20	COSH.....	136
6.16.21	COT.....	136
6.16.22	COTH.....	136
6.16.23	CSC.....	136
6.16.24	CSCH.....	137
6.16.25	DEGREES.....	137
6.16.26	DELTA.....	137
6.16.27	ERF.....	137
6.16.28	ERFC.....	138
6.16.29	EUROCONVERT.....	138
6.16.30	EVEN.....	139
6.16.31	EXP.....	139
6.16.32	FACT.....	139
6.16.33	FACTDOUBLE.....	140
6.16.34	GAMMA.....	140
6.16.35	GAMMALN.....	140
6.16.36	GCD.....	140
6.16.37	GESTEP.....	141
6.16.38	LCM.....	141
6.16.39	LN.....	141
6.16.40	LOG.....	141
6.16.41	LOG10.....	142
6.16.42	MOD.....	142
6.16.43	MULTINOMIAL.....	142
6.16.44	ODD.....	142
6.16.45	PI.....	142
6.16.46	POWER.....	143
6.16.47	PRODUCT.....	143
6.16.48	QUOTIENT.....	143
6.16.49	RADIANS.....	143
6.16.50	RAND.....	144
6.16.51	RANDBETWEEN.....	144
6.16.52	SEC.....	144

6.16.53	SERIESSUM.....	144
6.16.54	SIGN.....	145
6.16.55	SIN.....	145
6.16.56	SINH.....	145
6.16.57	SECH.....	145
6.16.58	SQRT.....	146
6.16.59	SQRTPI.....	146
6.16.60	SUBTOTAL.....	146
6.16.61	SUM.....	147
6.16.62	SUMIF.....	147
6.16.63	SUMIFS.....	147
6.16.64	SUMPRODUCT.....	148
6.16.65	SUMSQ.....	148
6.16.66	SUMX2MY2.....	148
6.16.67	SUMX2PY2.....	148
6.16.68	SUMXMY2.....	149
6.16.69	TAN.....	149
6.16.70	TANH.....	149
6.17	Rounding Functions.....	149
6.17.1	CEILING.....	149
6.17.2	INT.....	150
6.17.3	FLOOR.....	150
6.17.4	MROUND.....	150
6.17.5	ROUND.....	151
6.17.6	ROUNDDOWN.....	151
6.17.7	ROUNDUP.....	151
6.17.8	TRUNC.....	151
6.18	Statistical Functions.....	152
6.18.1	General.....	152
6.18.2	AVEDEV.....	152
6.18.3	AVERAGE.....	152
6.18.4	AVERAGEA.....	152
6.18.5	AVERAGEIF.....	152
6.18.6	AVERAGEIFS.....	153
6.18.7	BETADIST.....	153
6.18.8	BETAINV.....	154
6.18.9	BINOM.DIST.RANGE.....	154
6.18.10	BINOMDIST.....	154

6.18.11	LEGACY.CHIDIST.....	155
6.18.12	CHISQDIST.....	155
6.18.13	LEGACY.CHIINV.....	155
6.18.14	CHISQINV.....	156
6.18.15	LEGACY.CHITEST.....	156
6.18.16	CONFIDENCE.....	156
6.18.17	CORREL.....	157
6.18.18	COVAR.....	157
6.18.19	CRITBINOM.....	157
6.18.20	DEVSQ.....	158
6.18.21	EXPONDIST.....	158
6.18.22	FDIST.....	158
6.18.23	LEGACY.FDIST.....	159
6.18.24	FINV.....	159
6.18.25	LEGACY.FINV.....	159
6.18.26	FISHER.....	160
6.18.27	FISHERINV.....	160
6.18.28	FORECAST.....	160
6.18.29	FREQUENCY.....	161
6.18.30	FTEST.....	161
6.18.31	GAMMADIST.....	161
6.18.32	GAMMAINV.....	162
6.18.33	GAUSS.....	162
6.18.34	GEOMEAN.....	162
6.18.35	GROWTH.....	162
6.18.36	HARMEAN.....	163
6.18.37	HYPGEOMDIST.....	163
6.18.38	INTERCEPT.....	164
6.18.39	KURT.....	164
6.18.40	LARGE.....	165
6.18.41	LINEST.....	165
6.18.42	LOGEST.....	167
6.18.43	LOGINV.....	169
6.18.44	LOGNORMDIST.....	170
6.18.45	MAX.....	170
6.18.46	MAXA.....	170
6.18.47	MEDIAN.....	170
6.18.48	MIN.....	171

6.18.49	MINA.....	171
6.18.50	MODE.....	171
6.18.51	NEGBINOMDIST.....	172
6.18.52	NORMDIST.....	172
6.18.53	NORMINV.....	172
6.18.54	LEGACY.NORMSDIST.....	173
6.18.55	LEGACY.NORMSINV.....	173
6.18.56	PEARSON.....	173
6.18.57	PERCENTILE.....	174
6.18.58	PERCENTRANK.....	175
6.18.59	PERMUT.....	176
6.18.60	PERMUTATIONA.....	176
6.18.61	PHI.....	176
6.18.62	POISSON.....	176
6.18.63	PROB.....	177
6.18.64	QUARTILE.....	177
6.18.65	RANK.....	178
6.18.66	RSQ.....	178
6.18.67	SKEW.....	179
6.18.68	SKEWP.....	179
6.18.69	SLOPE.....	180
6.18.70	SMALL.....	180
6.18.71	STANDARDIZE.....	180
6.18.72	STDEV.....	180
6.18.73	STDEVA.....	181
6.18.74	STDEVP.....	181
6.18.75	STDEVPA.....	182
6.18.76	STEYX.....	182
6.18.77	LEGACY.TDIST.....	182
6.18.78	TINV.....	183
6.18.79	TREND.....	183
6.18.80	TRIMMEAN.....	184
6.18.81	TTEST.....	184
6.18.82	VAR.....	186
6.18.83	VARA.....	186
6.18.84	VARP.....	187
6.18.85	VARPA.....	187
6.18.86	WEIBULL.....	187

6.18.87 ZTEST.....	188
6.19 Number Representation Conversion Functions.....	188
6.19.1 General.....	188
6.19.2 ARABIC.....	189
6.19.3 BASE.....	189
6.19.4 BIN2DEC.....	189
6.19.5 BIN2HEX.....	190
6.19.6 BIN2OCT.....	190
6.19.7 DEC2BIN.....	190
6.19.8 DEC2HEX.....	191
6.19.9 DEC2OCT.....	191
6.19.10 DECIMAL.....	192
6.19.11 HEX2BIN.....	192
6.19.12 HEX2DEC.....	192
6.19.13 HEX2OCT.....	193
6.19.14 OCT2BIN.....	193
6.19.15 OCT2DEC.....	193
6.19.16 OCT2HEX.....	194
6.19.17 ROMAN.....	194
6.20 Text Functions.....	195
6.20.1 General.....	195
6.20.2 ASC.....	195
6.20.3 CHAR.....	197
6.20.4 CLEAN.....	198
6.20.5 CODE.....	198
6.20.6 CONCATENATE.....	198
6.20.7 DOLLAR.....	198
6.20.8 EXACT.....	198
6.20.9 FIND.....	199
6.20.10 FIXED.....	199
6.20.11 JIS.....	199
6.20.12 LEFT.....	201
6.20.13 LEN.....	201
6.20.14 LOWER.....	202
6.20.15 MID.....	202
6.20.16 PROPER.....	202
6.20.17 REPLACE.....	203
6.20.18 REPT.....	203

6.20.19	RIGHT.....	203
6.20.20	SEARCH.....	204
6.20.21	SUBSTITUTE.....	204
6.20.22	T.....	204
6.20.23	TEXT.....	204
6.20.24	TRIM.....	205
6.20.25	UNICHAR.....	205
6.20.26	UNICODE.....	205
6.20.27	UPPER.....	205
7	Other Capabilities.....	207
7.1	General.....	207
7.2	Inline constant arrays.....	207
7.3	Inline non-constant arrays.....	207
7.4	Year 1583.....	207
8	Non-portable Features.....	208
8.1	General.....	208
8.2	Distinct Logical.....	208
Appendix A.	Changes From Previous Specification Versions (Non Normative).....	209
A.1.	Changes from “Open Document Format for Office Applications (OpenDocument) v1.2”	209

1 Introduction

1.1 Introduction

This document is part of the Open Document Format for Office Applications (OpenDocument) Version 1.3 specification. It defines a formula language for OpenDocument documents, which is also called OpenFormula.

OpenFormula is a specification of an open format for exchanging recalculated formulas between office applications, in particular, formulas in spreadsheet documents. OpenFormula defines data types, syntax, and semantics for recalculated formulas, including predefined functions and operations.

OpenFormula is intended to be a supporting document to the Open Document Format for Office Applications (OpenDocument) format, particularly for defining its attributes `table:formula` and `text:formula`. It can also be used in other circumstances where a simple, easy-to-read infix text notation is desired for exchanging recalculated formulas.

Note: Using OpenFormula allows document creators to change the office application they use, exchange formulas with others (who may use a different application), and access formulas far in the future, with confidence that the recalculated formulas in their documents will produce equivalent results if given equivalent inputs.

1.2 Terminology

All text is normative unless otherwise labeled.

Within the normative text of this specification, the terms “shall”, “shall not”, “should”, “should not”, “may” and “need not” are to be interpreted as described in rules for the interpretation of international standards.

1.3 Purpose

OpenFormula defines:

1. data types
2. syntax
3. semantics

for recalculated formulas. 3.5

OpenFormula also defines functions.

OpenFormula does not define:

1. a user interface
2. a general notation for mathematical expressions

1.4 Normative References

[CharModel] Martin J. Dürst, et. al., Character Model for the World Wide Web 1.0: Fundamentals, <http://www.w3.org/TR/2005/REC-charmod-20050215/>, W3C, 2005.

[ISO4217] ISO 4217:2008 Codes for the representation of currencies and funds, International Organization for Standardization and International Electrotechnical Commission, 2008.

[ISO8601] ISO 8601:2004 Data elements and interchange formats -- Information interchange -- Representation of dates and times, International Organization for Standardization and International Electrotechnical Commission, 2004.

[RFC3986] T. Berners-Lee, R. Fielding, L. Masinter, Uniform Resource Identifier (URI): Generic Syntax, <http://www.ietf.org/rfc/rfc3986.txt>, IETF, 2005.

[RFC3987] M. Duerst, M. Suignard, Internationalized Resource Identifiers (IRIs), <http://www.ietf.org/rfc/rfc3987.txt>, IETF, 2005.

[UNICODE] The Unicode Consortium. The Unicode Standard, Version 5.2.0, defined by: The Unicode Standard, Version 5.2 (Mountain View, CA, The Unicode Consortium, 2009. ISBN 978-1-936213-00-9). (<http://www.unicode.org/versions/Unicode5.2.0/>).

[UTR15] Mark Davis, Martin Dürst, Unicode Normalization Forms, Unicode Technical Report #15, <http://www.unicode.org/reports/tr15/tr15-25.html>, 2005.

[XML1.0] Tim Bray, Jean Paoli, C. M. Sperberg-McQueen, Eve Maler, François Yergeau , Extensible Markup Language (XML) 1.0 (Fourth Edition), <http://www.w3.org/TR/2006/REC-xml-20060816/>, W3C, 2006.

1.5 Non-Normative References

[JISX0201] The Unicode Consortium., JIS X 0201 (1976) to Unicode 1.1 Table, 1994, <http://www.unicode.org/Public/MAPPINGS/OBSOLETE/EASTASIA/JIS/JIS0201.TXT>.

[JISX0208] The Unicode Consortium., JIS X 0208 (1990) to Unicode, 1994, <http://www.unicode.org/Public/MAPPINGS/OBSOLETE/EASTASIA/JIS/JIS0208.TXT>.

[UAX11] Asmus Freytag, East Asian Width, Unicode Standard Annex #11, <http://www.unicode.org/reports/tr11/tr11-19.html>, 2009.

2 Expressions and Evaluators

2.1 Introduction

The OpenDocument specification defines conformance for formula expressions and evaluators. For evaluators, there are three groups of features that an evaluator may support. This chapter defines the basic requirements for the individual conformance targets.

2.2 OpenDocument Formula Expression

An *OpenDocument formula expression* shall adhere to the expression syntax defined in chapter 4. It may use subsets or supersets of OpenFormula.

2.3 Evaluator Conformance

2.3.1 OpenDocument Formula Evaluator

An *OpenDocument Formula Evaluator* is a program that can parse and recalculate OpenDocument formula expressions, and that meets the following additional requirements:

- A) It may implement subsets or supersets of this specification.
- B) It shall conform to one of: (C16) OpenDocument Formula Small Group Evaluator, (C17) OpenDocument Formula Medium Group Evaluator or (C18) OpenDocument Formula Large Group Evaluator
- C) It may implement additional functions beyond those defined in this specification. It may further implement additional formula syntax, additional operations, additional optional parameters for functions, or may consider function parameters to be optional when they are required by this specification.
- D) Applications should clearly document their extensions in their user documentation, both online and paper, in a manner so users would be likely to be aware when they are using a non-standard extension.

Note 1: An expression may reference a function not defined by this specification by name, or depend on implementation-defined behavior, or on semantics not guaranteed by this specification. Reference to or dependence upon functions or behavior not defined by this standard may impair the interoperability of the resulting expression(s).

Note 2: This specification defines formulas in terms of a canonical text representation used for exchange. If formulas are contained in XML attributes some characters shall be escaped as required by the XML specification (e.g., the character & shall be escaped in XML attributes using notations such as &). All string and character literals references by this specification are in the value space defined by [UNICODE] thus, "A" is U+0041, "Z" is U+005A, and the range of characters "A-Z" is the range U+0041 through U+005A inclusive.

2.3.2 OpenDocument Formula Small Group Evaluator

An *OpenDocument Formula Small Group Evaluator* is an OpenDocument Formula Evaluator that meets the following additional requirements:

- A) It shall implement at least the limits defined in the "Basic Limits" section. 3.7

B) It shall implement the syntax defined in these sections on syntax: Criteria 4.11.11; Basic Expressions 5.2; Constant Numbers 5.3; Constant Strings 5.4; Operators 5.5; Functions and Function Parameters 5.6; Nonstandard Function Names 5.7; References 5.8; Simple Named Expressions ; Errors 5.12; Whitespace 5.14.

C) It shall implement all implicit conversions for the types it implements, at least Text 6.3.14, Conversion to Number 6.3.5, Reference , Conversion to Logical 6.3.12, and when an expression returns an Error.

D) It shall implement the following operators (which are all the operators except reference union (~)): Infix Operator Ordered Comparison (" $<$ ", " $<=$ ", " $>$ ", " $>=$ ") 6.3.5; Infix Operator "&" 6.4.10; Infix Operator "+" 6.4.2; Infix Operator "-" 6.4.3; Infix Operator "*" 6.4.4; Infix Operator "/" 6.4.5; Infix Operator "^" 6.4.6; Infix Operator "=" 6.4.7; Infix Operator "<>" 6.4.8; Postfix Operator "%" 6.4.14; Prefix Operator "+" 6.4.15; Prefix Operator "-" 6.4.16; Infix Operator Reference Intersection ("!") 6.4.12; Infix Operator Range (":") 6.4.11.

E) It shall implement at least the following functions as defined in this specification: ABS 6.16.2 ; ACOS 6.16.3 ; AND 6.15.2 ; ASIN 6.16.7 ; ATAN 6.16.9 ; ATAN2 6.16.10 ; AVERAGE 6.18.3 ; AVERAGEIF 6.18.5 ; CHOOSE 6.14.3 ; COLUMNS 6.13.5 ; COS 6.16.19 ; COUNT 6.13.6 ; COUNTA 6.13.7 ; COUNTBLANK 6.13.8 ; COUNTIF 6.13.9 ; DATE 6.10.2 ; DAVERAGE 6.9.2 ; DAY 6.10.5 ; DCOUNT 6.9.3 ; DCOUNTA 6.9.4 ; DDB 6.12.14 ; DEGREES 6.16.25 ; DGET 6.9.5 ; DMAX 6.9.6 ; DMIN 6.9.7 ; DPRODUCT 6.9.8 ; DSTDEV 6.9.9 ; DSTDEVP 6.9.10 ; DSUM 6.9.11 ; DVAR 6.9.12 ; DVARP 6.9.13 ; EVEN 6.16.30 ; EXACT 6.20.8 ; EXP 6.16.31 ; FACT 6.16.32 ; FALSE 6.15.3 ; FIND 6.20.9 ; FV 6.12.20 ; HLOOKUP 6.14.5 ; HOUR 6.10.10 ; IF 6.15.4 ; INDEX 6.14.6 ; INT 6.17.2 ; IRR 6.12.24 ; ISBLANK 6.13.14 ; ISERR 6.13.15 ; ISERROR 6.13.16 ; ISLOGICAL 6.13.19 ; ISNA 6.13.20 ; ISNONTEXT 6.13.21 ; ISNUMBER 6.13.22 ; ISTEXT 6.13.25 ; LEFT 6.20.12 ; LEN 6.20.13 ; LN 6.16.39 ; LOG 6.16.40 ; LOG10 6.16.41 ; LOWER 6.20.14 ; MATCH 6.14.9 ; MAX 6.18.45 ; MID 6.20.15 ; MIN 6.18.48 ; MINUTE 6.10.12 ; MOD 6.16.42 ; MONTH 6.10.13 ; N 6.13.26 ; NA 6.13.27 ; NOT 6.15.7 ; NOW 6.10.15 ; NPV 6.12.29 ; NPV 6.12.30 ; ODD 6.16.44 ; OR 6.15.8 ; PI 6.16.45 ; PMT 6.12.36 ; POWER 6.16.46 ; PRODUCT 6.16.47 ; PROPER 6.20.16 ; PV 6.12.41 ; RADIANS 6.16.49 ; RATE 6.12.42 ; REPLACE 6.20.17 ; REPT 6.20.18 ; RIGHT 6.20.19 ; ROUND 6.17.5 ; ROWS 6.13.30 ; SECOND 6.10.16 ; SIN 6.16.55 ; SLN 6.12.45 ; SQRT 6.16.58 ; STDEV 6.18.72 ; STDEVP 6.18.74 ; SUBSTITUTE 6.20.21 ; SUM 6.16.61 ; SUMIF 6.16.62 ; SYD 6.12.46 ; T 6.20.22 ; TAN 6.16.69 ; TIME 6.10.17 ; TODAY 6.10.19 ; TRIM 6.20.24 ; TRUE 6.15.9 ; TRUNC 6.17.8 ; UPPER 6.20.27 ; VALUE 6.13.34 ; VAR 6.18.82 ; VARP 6.18.84 ; VLOOKUP 6.14.12 ; WEEKDAY 6.10.20 ; YEAR 6.10.23

F) It need not evaluate references that contain more than one area.

G) It need not implement inline arrays 5.13, complex numbers 4.4, and the reference union operator 6.4.13.

Note: This specification does not mandate a user interface for international characters, so a resource-constrained application may choose to not show the traditional glyph (e.g., it may show the [UNICODE] numeric code instead).

2.3.3 OpenDocument Formula Medium Group Evaluator

An *OpenDocument Formula Medium Group Evaluator* is an OpenDocument Small Group Formula Evaluator that meets the following additional requirements:

A) It shall implement the following functions as defined in this specification: ACCRINT 6.12.2 ; ACCRINTM 6.12.3 ; ACOSH 6.16.4 ; ACOT 6.16.5 ; ACOTH 6.16.6 ; ADDRESS 6.14.2 ; ASINH 6.16.8 ; ATANH 6.16.11 ; AVEDEV 6.18.2 ; BESSELI 6.16.12 ; BESSELJ 6.16.13 ; BESSELK 6.16.14 ; BESSELY 6.16.15 ; BETADIST 6.18.7 ; BETAINV 6.18.8 ; BINOMDIST 6.18.10 ; CEILING 6.17.1 ; CHAR 6.20.3 ; CLEAN 6.20.4 ; CODE 6.20.5 ; COLUMN 6.13.4 ; COMBIN 6.16.16 ; CONCATENATE 6.20.6 ; CONFIDENCE 6.18.16 ; CONVERT 6.16.18 ; CORREL 6.18.17 ; COSH 6.16.20 ; COT 6.16.21 ; COTH 6.16.22 ; COUPDAYBS 6.12.5 ; COUPDAYS 6.12.6 ; COUPDAYSNC 6.12.7 ; COUPNCD 6.12.7 ; COUPNUM 6.12.9 ; COUPPCD 6.12.10 ; COVAR 6.18.18 ; CRITBINOM 6.18.19 ; CUMIPMT 6.12.11 ; CUMPRINC 6.12.12 ; DATEVALUE 6.10.4 ; DAYS360 6.10.7 ; DB 6.12.13 ; DEVSQ 6.18.20 ; DISC 6.12.15 ; DOLLARDE 6.12.16 ;

DOLLARFR 6.12.17 ; DURATION 6.12.18 ; EFFECT 6.12.19 ; EOMONTH 6.10.9 ; ERF 6.16.27 ;
 ERFC 6.16.28 ; EXPONDIS 6.18.21 ; FISHER 6.18.26 ; FISHERINV 6.18.27 ; FIXED 6.20.10 ;
 FLOOR 6.17.3 ; FORECAST 6.18.28 ; FTEST 6.18.30 ; GAMMADIST 6.18.31 ; GAMMAINV
 6.18.32 ; GAMMALN 6.16.35 ; GCD 6.16.36 ; GEOMEAN 6.18.34 ; HARMEAN 6.18.36 ;
 HYPGEOMDIST 6.18.37 ; INTERCEPT 6.18.38 ; INTRATE 6.12.22 ; ISEVEN 6.13.17 ; ISODD
 6.13.23 ; ISOWEENUM 6.10.11 ; KURT 6.18.39 ; LARGE 6.18.40 ; LCM 6.16.38 ;
 LEGACY.CHIDIST 6.18.11 ; LEGACY.CHIINV 6.18.13 ; LEGACY.CHITEST 6.18.15 ;
 LEGACY.FDIST 6.18.23 ; LEGACY.FINV 6.18.25 ; LEGACY.NORMSDIST 6.18.54 ;
 LEGACY.NORMSINV 6.18.55 ; LEGACY.TDIST 6.18.77 ; LINEST 6.18.41 ; LOGEST 6.18.42 ;
 LOGINV 6.18.43 ; LOGNORMDIST 6.18.44 ; LOOKUP 6.14.8 ; MDURATION 6.12.26 ; MEDIAN
 6.18.47 ; MINVERSE 6.5.3 ; MIRR 6.12.27 ; MMULT 6.5.4 ; MODE 6.18.50 ; MROUND 6.17.4 ;
 MULTINOMIAL 6.16.43 ; NEGBINOMDIST 6.18.51 ; NETWORKDAYS 6.10.14 ; NOMINAL
 6.12.28 ; ODDFPRICE 6.12.31 ; ODDFYIELD 6.12.32 ; ODDLPRICE 6.12.33 ; ODDLYIELD
 6.12.34 ; OFFSET 6.14.11 ; PEARSON 6.18.56 ; PERCENTILE 6.18.57 ; PERCENTRANK
 6.18.58 ; PERMUT 6.18.59 ; POISSON 6.18.62 ; PRICE 6.12.38 ; PRICEMAT 6.12.40 ; PROB
 6.18.63 ; QUARTILE 6.18.64 ; QUOTIENT 6.16.48 ; RAND 6.16.50 ; RANDBETWEEN 6.16.51 ;
 RANK 6.18.65 ; RECEIVED 6.12.43 ; ROMAN 6.19.17 ; ROUNDDOWN 6.17.6 ; ROUNDUP
 6.17.7 ; ROW 6.13.29 ; RSQ 6.18.66 ; SERIESSUM 6.16.53 ; SIGN 6.16.54 ; SINH 6.16.56 ;
 SKEW 6.18.67 ; SKEWP 6.18.68 ; SLOPE 6.18.69 ; SMALL 6.18.70 ; SQRTPI 6.16.59 ;
 STANDARDIZE 6.18.71 ; STDEVA 6.18.73 ; STDEVPA 6.18.75 ; STEYX 6.18.76 ; SUBTOTAL
 6.16.60 ; SUMPRODUCT 6.16.64 ; SUMSQ 6.16.65 ; SUMX2MY2 6.16.66 ; SUMX2PY2 6.16.67 ;
 SUMXMY2 6.16.68 ; TANH 6.16.70 ; TBILLEQ 6.12.47 ; TBILLPRICE 6.12.48 ; TBILLYIELD
 6.12.49 ; TIMEVALUE 6.10.18 ; TINV 6.18.78 ; TRANSPOSE 6.5.6 ; TREND 6.18.79 ;
 TRIMMEAN 6.18.80 ; TTEST 6.18.81 ; TYPE 6.13.33 ; VARA 6.18.83 ; VDB 6.12.50 ; WEEKNUM
 6.10.21 ; WEIBULL 6.18.86 ; WORKDAY 6.10.22 ; XIRR 6.12.51 ; XNPV 6.12.52 ; YEARFRAC
 6.10.24 ; YIELD 6.12.53 ; YIELDDISC 6.12.54 ; YIELDMAT 6.12.55 ; ZTEST 6.18.87

B) It shall implement the Infix Operator Reference Union ("~") 6.4.13

C) It shall evaluate references with more than one area.

2.3.4 OpenDocument Formula Large Group Evaluator

An *OpenDocument Formula Large Group Evaluator* is an OpenDocument Medium Group Formula Evaluator that meets the following additional requirements:

A) It shall implement the syntax defined in these sections on syntax: Inline Arrays 5.13; Automatic Intersection 5.10.6; External Named Expressions 5.11.

B) It shall implement the complex number type as discussed in the section on Complex Number 4.4, array formulas, and Sheet-local Named Expressions.

It shall implement the following functions as defined in this specification: AMORLINC 6.12.4 ;
 ARABIC 6.19.2 ; AREAS 6.13.2 ; ASC 6.20.2 ; AVERAGEA 6.18.4 ; AVERAGEIFS 6.18.6 ; BASE
 6.19.3 ; BIN2DEC 6.19.4 ; BIN2HEX 6.19.5 ; BIN2OCT 6.19.6 ; BINOM.DIST.RANGE 6.18.9 ;
 BITAND 6.6.2 ; BITLSHIFT 6.6.3 ; BITOR 6.6.4 ; BITRSHIFT 6.6.5 ; BITXOR 6.6.6 ; CHISQDIST
 6.18.12 ; CHISQINV 6.18.14 ; COMBINA 6.16.17 ; COMPLEX 6.8.2 ; COUNTIFS 6.13.10 ; CSC
 6.16.23 ; 6.16.23CSCH 6.16.24 ; DATEDIF 6.10.3 ; DAYS 6.10.6 ; DDE 6.11.2 ; DEC2BIN 6.19.7 ;
 DEC2HEX 6.19.8 ; DEC2OCT 6.19.9 ; DECIMAL 6.19.10 ; DELTA 6.16.26 ; EDATE 6.10.8 ;
 ERROR.TYPE 6.13.11 ; EUROCONVERT 6.16.29 ; FACTDOUBLE 6.16.33 ; FDIST 6.18.22 ;
 FINDB 6.7.2 ; FINV 6.18.24 ; FORMULA 6.13.12 ; FREQUENCY 6.18.29 ; FVSCHEDULE 6.12.21
 ; GAMMA 6.16.34 ; GAUSS 6.18.33 ; GESTEP 6.16.37 ; GETPIVOTDATA 6.14.4 ; GROWTH
 6.18.35 ; HEX2BIN 6.19.11 ; HEX2DEC 6.19.12 ; HEX2OCT 6.19.13 ; HYPERLINK 6.11.3 ;
 IFERROR 6.15.5 ; IFNA 6.15.6 ; IMABS 6.8.3 ; IMAGINARY 6.8.4 ; IMARGUMENT 6.8.5 ;
 IMCONJUGATE 6.8.6 ; IMCOS 6.8.7 ; IMCOT 6.8.9 ; IMCSC 6.8.10 ; IMCSCH 6.8.11 ; IMDIV
 6.8.12 ; IMEXP 6.8.13 ; IMLN 6.8.14 ; IMLOG10 6.8.15 ; IMLOG2 6.8.16 ; IMPOWER 6.8.17 ;
 IMPRODUCT 6.8.18 ; IMREAL 6.8.19 ; IMSEC 6.8.22 ; IMSECH 6.8.23 ; IMSIN 6.8.20 ; IMSQRT
 6.8.24 ; IMSUB 6.8.25 ; IMSUM 6.8.26 ; IMTAN 6.8.27 ; INDIRECT 6.14.7 ; INFO 6.13.13 ; IPMT
 6.12.23 ; ISFORMULA 6.13.18 ; ISPMT 6.12.25 ; ISREF 6.13.24 ; JIS 6.20.11 ; LEFTB 6.7.3 ;
 LENB 6.7.4 ; MAXA 6.18.46 ; MDETERM 6.5.2 ; MULTIPLE.OPERATIONS 6.14.10 ; MUNIT 6.5.5
 ; MIDB 6.7.5 ; MINA 6.18.49 ; NORMDIST 6.18.52 ; NORMINV 6.18.53 ; NUMBERTOVALUE

6.13.28 ; OCT2BIN 6.19.14 ; OCT2DEC 6.19.15 ; OCT2HEX 6.19.16 ; PDURATION 6.12.35 ; PERMUTATIONA 6.18.60 ; PHI 6.18.61 ; PPMT 6.12.37 ; PRICEDISC 6.12.39 ; REPLACEB 6.7.6 ; RIGHTB 6.7.7 ; RRI 6.12.44 ; SEARCH 6.20.20 ; SEARCHB 6.7.8 ; SEC 6.16.52 ; SECH 6.16.57 ; SHEET 6.13.31 ; SHEETS 6.13.32 ; SUMIFS 6.16.63 ; TEXT 6.20.23 ; UNICHAR 6.20.25 ; UNICODE 6.20.26 ; VARPA 6.18.85 ; XOR 6.15.10

Note: The following functions are documented by this specification, but not included even in the Large group: CELL 6.13.3 ; DOLLAR 6.20.7

2.4 Variances (Implementation-defined, Unspecified, and Behavioral Changes)

Applications should document all implementation-defined variances from this standard in a manner that enables application users to obtain the information.

In a few cases a specific approach is required (e.g., string indexes begin at one), which may vary in the user interfaces of different implementations.

Note 1: In practice, for nearly all documents the differences are irrelevant. The primary variances and differences from OpenFormula and some existing applications are:

- Some conversions between types are not required to be automatic. In particular, applications may, but need not, perform automatic conversion of text in a cell when it is to be used as a number.
- There need not be a distinguishable Logical type. Applications may have a Logical type distinct from Number and Text (see Distinct Logical 8.2), but Logical values may also be represented by the Number type using the values 1 (TRUE) and 0 (FALSE). This means that functions that take number sequences (such as SUM) may but need not include true and false values in the sequence.
- Applications vary on the set of Errors they support. In this specification the only distinguished Error is #N/A; all other errors are simply errors, allowing applications to choose the Error set that best meets their needs.
- In this specification, string index positions start from 1. Users of applications with string index positions starting from 0 shall add and subtract 1 on import/export of this format, as appropriate.
- Database criteria match patterns (such as the pattern matching language for text) have historically varied: Some support glob syntax (e.g., a*b is a, followed by 0 or more characters, followed by b), while others support traditional regular expression syntax (e.g., a*b is zero or more a's, followed by b). This specification supports both pattern languages.

Note 2: Interoperability is improved by the use of the DATE 6.10.2 and TIME 6.10.17 functions or the textual [ISO8601] date representation because dates in that format do not rely upon epoch or locale-specific settings.

In an OpenDocument file, calculation settings impact formula recalculation, which can be the same or different from a particular application's defaults. These include whether or not text comparisons are case-sensitive, or if search criteria apply to the whole cell.

3 Formula Processing Model

3.1 General

This section describes the basic formula processing model: how expressions are calculated, when recalculation occurs, and limits on formulas.

3.2 Expression Evaluation

3.2.1 General

OpenFormula defines rules for the evaluation of expressions as well as the functions and operators that appear in expressions.

3.2.2 Expression Calculation

Expressions in OpenFormula shall be evaluated by application of the following rules:

- 1) If an expression consists of a constant Number (5.3), a constant String (5.4), a Reference (5.8), constant Error (per section 5.12), the value of that type is returned.
- 2) If an expression consists of one or more operations, apply the operators in order of precedence and associativity as defined by Table 1 in 5.5 (Operators). Precedence of operators may be altered by the use of "(" (LEFT PARENTHESES, U+0028) and ")" (RIGHT PARENTHESES, U+0029) to group operators. Evaluate the operator as described in Operator and Function Evaluation, 3.2.3.
- 3) If an expression consists of a function call (5.6, 5.7), evaluate the function as described in Operator and Function Evaluation, 3.2.3.
- 4) If an expression consists of a named expression (5.11), the result of evaluating the named expression is returned.
- 5) If an expression consists of a QuotedLabel (5.10), AutomaticIntersection (5.10.6), or Array (5.13), its value is returned. Expression Syntax 5

Once evaluation has completed:

1. If the result is a Reference and a single non-reference value is needed, it is converted to the referenced value, using the rules of Non-Scalar Evaluation, 3.3, 1.2.
2. If the result is an Array, for the display area, apply the rules of Non-Scalar Evaluation, 3.3, 1.1.

3.2.3 Operator and Function Evaluation

Operators and functions in OpenFormula **shall** be evaluated according to their definitions by applying the following rules:

1. The value of all expression arguments are computed. Exceptions to computation of all arguments are noted in a function's specification.

Note: The practice of computing all argument expressions is known as "eager" evaluation. The IF function is an example of a function that does not require computation of all arguments.

2. If an argument expression evaluates to Error, calculation of the operator or function may short-circuit and return the Error if the function does not suppress error propagation as noted in the function's specification.

3. If an operator or function is passed a value of incorrect type, call the appropriate implicit conversion function to convert the value to the correct type. If conversion is not possible, generate an Error.
4. The function or operation is called with its argument expressions' results, and the result of the function or operation is the evaluation of the expression.

3.3 Non-Scalar Evaluation (aka 'Array expressions')

Non-scalar values passed as arguments to functions are evaluated by intersection or iteration.

- 1) Evaluation as an implicit intersection of the argument with the expression's evaluation position.

1.1) Inline Arrays

Element (0;0) of the array is used in place of the array.

Note 1:

```
=ABS({-3;-4})      => ABS(-3)      // row vector
=ABS({-3|-4})      => ABS(-3)      // column vector
=ABS({-3;-4|-6;-8}) => ABS(-3)      // matrix
={1;2;3|4;5;6}      => 1           // simple display
```

1.2) References

- 1.2.1) If the target reference is a row-vector (Nx1), use the value at the intersection of the evaluation position's column and the reference's row.

Note 2:

in cell B2 : =ABS(A1:C1) => ABS(B1)

If there is no intersection the result is #N/A or a more specific Error value.

Note 3: in cell D4 : =ABS(A1:C1) => #N/A or a more specific Error value.

- 1.2.2) If the target reference is a column-vector (1xM), the value at the intersection of the evaluation position's row and the reference's column.

Note 4:

in cell B2 : =ABS(A1:A3) => ABS(A2)

in cell D4 : =ABS(A1:A3) => #N/A or a more specific Error value.

- 2) Matrix evaluation.

If an expression is being evaluated in a cell flagged as a being part of a 'Matrix' (OpenDocument Part 3, 19.683 table:number-matrix-columns-spanned):

- 2.1) The portion of a non-scalar result to be displayed need not be co-extensive with a specified display area. The portion of the non-scalar result to be displayed is determined by:

- 2.1.1) If the position to be displayed exists in the result, display that position.

- 2.1.2) If the non-scalar result is 1 column wide, subsequent columns in the display area display the value in the first column. This applies to

- scalars '3'
- singletons '{3}'
- column vectors '{1|2|3}'

- 2.1.3) If the non-scalar result is 1 row high, subsequent rows in the display area use the value of the first row. This applies to

- scalars '3'
- singletons '{3}'
- row vectors '{1;2;3}'

- 2.1.4) If none of the other rules apply #N/A

Note 5:

in matrix A1:B3 with $=\{1;2|3;4|5;6\}$: cell B2 contains 4. [Rule 2.1.1]
 in matrix A1:B3 with $=\{1|3|5\}$: cell B2 contains 3. [Rule 2.1.1 for
 row, and Rule 2.1.2 column]
 in matrix A1:B3 with $=\{2;4\}$: cell B2 contains 4. [Rule 2.1.3 for
 row, and Rule 2.1.1 column]
 in matrix A1:C4 with $=\{1;2|3;4|5;6\}$: cell C1,A4 contain #N/A. [Rule 2.1.4]

Note 6: if a value is not requested it is not displayed

in matrix A1:B2 with $=\{1;2|3;4|5;6\}$: the value '6' is not displayed because
 B3 is not part of the display matrix.

2.2) Calculations with non-scalar inputs are a generalization of (2.1).

When evaluating a formula in 'matrix' mode, and a non-scalar value is passed to a function argument that expects a scalar, the function is evaluated multiple times, iterating over the non-scalar input(s) and putting the function result into a matrix at the position corresponding to the input. Unary/Binary operators, other than range and union, follow the rules for scalar functions when passed non-scalar values.

Inline arrays and references are interchangeable.

2.2.1) Functions returning arrays are not eligible for implicit iteration. When evaluated in 'matrix' mode the {0;0}th element is used.

Note 7:

e.g. $=\text{SUM}(\text{INDIRECT}(\{"A1";"A2"\}))$ would produce the value in A1 when evaluated in array mode.

2.2.2) The result matrix is rectangular, sized with the maximum number of rows and columns from all non-scalar arguments.

Note 8:

$=\{1;2\}+\{3;4;5\} \Rightarrow \{4;6;\#N/A\}$
 $=\{1\}+\{1;2\} \Rightarrow \{2;3\}$

2.2.3) The result matrix is populated by extracting the corresponding value from each of the non-scalar arguments based on the following rules, and evaluating the function with that set of arguments.

2.2.3.1) If the argument data is a singleton array or a scalar the value is repeated for each evaluation.

Note 9:

$= 1 + \{1;2;3|4;5;6\} \Rightarrow \{2;3;4|5;6;7\}$
 $= \{1\} + \{1;2;3|4;5;6\} \Rightarrow \{2;3;4|5;6;7\}$

2.2.3.2) If the argument data is 1 column wide the value in the corresponding row is used to evaluate all columns in the result matrix.

Note 10:

$= \{1|2\} + \{10;20|30;40\} \Rightarrow \{11;21|32;42\}$

2.2.3.3) If the argument data is 1 row height the value in the corresponding column is used to evaluate all rows in the result matrix.

Note 11:

$= \{1;2\} + \{10;20|30;40\} \Rightarrow \{11;22|31;42\}$

2.2.3.4) If one argument data is 1 column wide and another argument data is 1 row height the value of the corresponding row respectively column is used to evaluate all elements in the result matrix.

Note 12:

$=\{1;2\} + \{10|20\} \Rightarrow \{11;12|21;22\}$

2.2.3.5) If an argument is a 2d matrix the argument value in the position corresponding to the current output position is used if it is within range of the supplied argument, otherwise #N/A is used in the calculation.

Note 13:

=MID("abcd";{1;2};{1;2;3}) => {"a";"bc";#N/A}

3.4 Host-Defined Behaviors

A Formula Evaluator operates in an execution environment (a "host"). The behavior of the Formula Evaluator is parametrized by host-defined properties and functions.

The following properties are host-defined:

1. **HOST-CASE-SENSITIVE:** if true, text comparisons are case-sensitive. This influences the operators =, <>, <, <=, >, and >=, as well as database query functions that use them. Note that the EXACT function is always case-sensitive, regardless of this calculation setting.
2. **HOST-PRECISION-AS-SHOWN:** If true, calculations are performed using rounded values of those displayed; otherwise, calculations are performed using the precision of the underlying numeric representation.
Note: This does not impose a particular numeric model. Since implementations may use binary representations, this rounding may be inexact for decimal values.
3. **HOST-SEARCH-CRITERIA-MUST-APPLY-TO-WHOLE-CELL** If true, the specified search criteria shall apply to the entire cell contents if it is a text match using = or <>; if not, only a subpart of the cell content needs to match the text.
4. **HOST-AUTOMATIC-FIND-LABELS:** if true, row and column labels are automatically found.
5. **HOST-USE-REGULAR-EXPRESSIONS:** If true, regular expressions are used for character string comparisons and when searching.
6. **HOST-USE-WILDCARDS:** If true, wildcards question mark '?' and asterisk '*' are used for character-string comparisons and when searching. Wildcards may be escaped with a tilde '~' character.
7. **HOST-NULL-YEAR:** This defines how to convert a two-digit year into a four-digit year. Each two-digit year value is interpreted as a year that equals or follows this year.
8. **HOST-NULL-DATE:** Defines the beginning of the epoch; a numeric date of 0 equals this date.
9. **HOST-LOCALE:** The locale to be used for locale-dependent operations, such as conversion of text to dates, or text to numbers.
10. **HOST-ITERATION-STATUS:** If enabled, iterative calculations of cyclic references are performed.
11. **HOST-ITERATION-MAXIMUM-DIFFERENCE:** If iterative calculations of cyclic references are enabled, the maximum absolute difference between calculation steps that all involved formula cells must yield for the iteration to end and yield a result.
12. **HOST-ITERATION-STEPS:** If iterative calculations of cyclic references are enabled, the maximum number of steps iterations that are performed if the results are not within HOST-ITERATION-MAXIMUM-DIFFERENCE.

The function HOST-REFERENCE-RESOLVER(Reference) is implementation-defined. This function takes as input a Unicode string containing a Reference according to section 4.8 and returns a resolved value.

3.5 When recalculation occurs

Implementations of OpenFormula typically recalculate formulas when its information is needed. Typical implementations will note what values a formula depends on, and when those dependent values are changed and the formula's results are displayed, it will re-execute the formulas that depend on them to produce the new results (choosing the formulas in the right order based on

their dependencies). Implementations may recalculate when a value changes (this is termed *automatic recalculation*) or on user command (this is termed *manual recalculation*).

Some functions' dependencies are difficult to determine and/or should be recalculated more frequently. These include functions that return today's date or time, random number generator functions (such as RAND 6.16.50), or ones that indirectly determine the cells to act on. Many implementations *always* recalculate formulas including such functions whenever a recalculation occurs. Functions that are *always* recalculated whenever a recalculation occurs are termed *volatile* functions. Functions that are often volatile functions include CELL 6.13.3, HYPERLINK 6.11.3, INDIRECT 6.14.7, INFO 6.13.13, NOW 6.10.15, OFFSET 6.14.11, RAND 6.16.50 and TODAY 6.10.19. Functions that depend on the cell position of the formula they are contained in or the position of a cell they reference need to be recalculated whenever that cell is moved, such functions are COLUMN 6.13.4, ROW 6.13.29 and SHEET 6.13.31. In addition, formulas may indicate that they should *always* be recalculated during a recalculation process by including a forced recalculation marker, as described in the syntax below.

3.6 Numerical Models

This specification does not, by itself, specify a numerical implementation model, though it does imply some minimal levels of accuracy for most functions. For example, an application cannot say that it implements the infix operator “/” as specified in this document if it implements integer-only arithmetic.

3.7 Basic Limits

Evaluators which claim to support “basic limits” shall support at least the following limits:

1. formulas up to at least 1024 characters long, as measured when in OpenDocument interchange format not counting the square brackets around cell addresses, or the “.” in a cell address when the sheet name is omitted.
2. at least 30 parameters per function when the function prototype permits a list of parameters.
3. permit strings of ASCII characters of up to 32,767 ($2^{15}-1$) characters.
4. support at least 7 nesting levels of functions.

4 Types

4.1 General

All values defined by OpenFormula have a type. OpenFormula defines Text 4.2, Number 4.3, Complex Number 4.4, Logical 4.5, Error 4.6, Reference 4.8, ReferenceList 4.9 and Array 4.10 types.

4.2 Text (String)

A Text value (also called a string value) is a Character string as specified in [CharModel].

A text value of length zero is termed the empty string.

Index positions in a text value begin at 1.

Whether or not Unicode Normalization [UTR15] is performed on formulas, formula results or user inputs is implementation-defined. Some functions defined in OpenFormula are labeled as "normalization-sensitive", meaning that the results of the formula evaluation may differ depending on whether normalization occurs, and which normalization form is used. Mixing operands of different normalization forms in the same calculation is undefined.

4.3 Number

4.3.1 General

A number is a numeric value.

Numbers shall be able to represent fractional values (they shall not be limited to only integers). Evaluators may implement Number with a fixed or with a variable bit length. A cell with a constant numeric value has the Number type.

Implementations typically support many subtypes of Number, including Date, Time, DateTime, Percentage, fixed-point arithmetic, and arithmetic supporting arbitrarily long integers, and determine the display format from this. All such Number subtypes shall yield TRUE for the ISNUMBER 6.13.22 function.

Note: This specification does not require that specific subtypes be distinguishable from each other, or that the subtype be tracked, but in practice most implementations do such tracking because requiring users to manually format every cell appropriately becomes tedious very quickly. Automatically determining the most likely subtype is especially important for a good user interface when generating OpenDocument format, since some subtypes (such as date, time, and currency) are stored in a different manner depending on their subtype. Thus, this specification identifies some common subtypes and identifies those subtypes where relevant in some function definitions, as an aid to implementing good user interfaces. Many applications vary in the subtype produced when combining subtypes (e.g., what is the result when percentages are multiplied together), so unless otherwise noted these are unspecified. Typical implementations try to heuristically determine the "right" format for a cell when a formula is first created, based on the operations in the formula. Users can then override this format, so as a result the heuristics are not important for data exchange (and thus outside the scope of this specification).

All Number subtypes shall yield TRUE for the ISNUMBER function.

4.3.2 Time

Time is a subtype of Number.

Time is represented as a fraction of a day.

4.3.3 Date

Date is a subtype of Number.

Date is represented by an integer value.

A serial date is the expression of a date as the number of days elapsed from a start date called the epoch.

Evaluators shall support all dates from 1904-01-01 through 9999-12-31 (inclusive) in calculations, should support dates from 1899-12-30 through 9999-12-31 (inclusive) and may support a wider date range.

Note 1: Using expressions that assume serial numbers are based on a particular epoch may cause interoperability issues.

Evaluators shall support positive serial numbers. Evaluators may support negative serial numbers to represent dates before an epoch.

Note 2: It is implementation-defined if the year 1900 is treated as a leap year.

Note 3: Evaluators that treat 1900 as a non-leap year can use the epoch date 1899-12-30 to compensate for serial numbers that originate from evaluators that treat 1900 as a leap year and use 1899-12-31 as an epoch date.

4.3.4 DateTime

DateTime is a subtype of Number. It is a Date plus Time.

4.3.5 Percentage

A percentage is a subtype of Number that may be displayed by multiplying the number by 100 and adding the sign “%” or with other formatting depending upon the number format assigned to the cell where it appears.

4.3.6 Currency

A currency is a subtype of Number that may appear with or without a currency symbol or with other formatting depending upon the number format assigned to the cell where it appears.

4.3.7 Logical (Number)

Applications may have a Logical type distinct from both Number and Text (see 4.5 Logical (Boolean)), but Logical values may also be represented by the Number type using the values 1 (True) and 0 (False). (see 8.2 Distinct Logical for details)

4.4 Complex Number

A complex number (sometimes also called an imaginary number) is a pair of real numbers including a *real part* and an *imaginary part*. In mathematics, complex numbers are often written as $x + iy$, where x (the real part) and y (the imaginary part) are real numbers and i is $\sqrt{-1}$. A complex number can also be written as $re^{i\theta} = r\cos(\theta) + ir\sin(\theta)$, where r is the *modulus* of the complex number (a real number) and θ is the *argument* or *phase* (a real number representing an angle in radians).

A complex number may, but need not be, represented as a Number or Text. The results of the functions ISNUMBER() 6.13.22 and ISTEXT() 6.13.25 are implementation-defined when applied to a complex number.

Functions and operators that accept complex numbers shall accept Text values as complex numbers (Conversion to Complex Number 6.3.10), as well as Numbers that are not complex numbers.

Note 1: IMSUM("3i";4) will produce the same result as COMPLEX(4;3).

Note 2: Expression authors should be aware that use of functions that are not defined by OpenFormula as accepting complex numbers as input may impair interoperability.

Equality can be tested using IMSUB to compute the difference, use IMABS to find the absolute difference, and then ensure the absolute difference is smaller than or equal to some nonnegative value (for exact equality, compare for equality with 0).

4.5 Logical (Boolean)

Applications may have a Logical type distinct from both Number and Text, but Logical values may also be represented by the Number type using the values 1 (True) and 0 (False) (see 4.3.7 Logical (Number)). (see 8.2 Distinct Logical for details)

4.6 Error

An Error is one of a set of possible error values. Implementations may have many different error values (see 5.12), but one error value in particular is distinct: #N/A, the result of the NA() function. Users may choose to enter some data values as #N/A, so that this error value propagates to any other formula that uses it, and may test for this using the function ISNA().

Functions and operators that receive one or more error values as an input shall produce one of those input error values as their result, except when the formula or operator is specifically defined to do otherwise.

In an OpenDocument document, if an error value is the result of a cell computation it shall be stored as if it was a string. That is, the `office:value-type` (OpenDocument Part 3, 19.389) of an error value is `string`; if the computed value is stored, it is stored in the attribute `office:string-value` (OpenDocument Part 3, 19.383).

Note: This does not change an Error into a string type (since the Error will be restored on recalculation); this enables applications which cannot recalculate values to display the error information.

4.7 Empty Cell

An empty cell is neither zero nor the empty string, and an empty cell can be distinguished from cells containing values (including zero and the empty string). An empty cell is not the same as an Error, in particular, it is distinguishable from the Error #N/A (not available).

4.8 Reference

A cell position is the location of a single cell at the intersection of a column and a row.

A cell strip consists of cell positions in the same row and in one or more contiguous columns.

A cell rectangle consists of cell positions in the same cell strips of one or more contiguous rows.

A cell cuboid consists of cell positions in the same cell rectangles of one or more contiguous sheets.

A reference is the smallest cuboid that (1) contains specifically-identified cell positions and/or specifically-identified complete columns/rows such that (2) removal of any cell positions either violates condition (1) or does not leave a cuboid.

Cell positions in a cell cuboid/rectangle/strip can resolve to empty cells (section 4.7).

The definitions of specific operations and functions that allow references as operands and parameters stipulate any particular limitations there are on forms of references and how empty cells, when permitted, are interpreted.

4.9 ReferenceList

A reference list contains 1 or more references, in order. A reference list can be passed as an argument to functions where passing one reference results in an identical computation as an arbitrary sequence of single references occupying the identical cell range.

Note 1: For example, SUM([.A1:.B2]) is identical to SUM([.A1]~[.B2]~[.A2]~[.B1]), but COLUMNS([.A1:.B2]), resulting in 2 columns, is not identical to COLUMNS([.A1]~[.B2]~[.A2]~[.B1]), where iterating over the reference list would result in 4 columns.

A reference list cannot be converted to an array.

Note 2: For example, in array context {ABS([.A1]~[.B2]~[.A2]~[.B1])} is an invalid expression, whereas {ABS([.A1:.B2])} is not.

Passing a reference list where a function does not expect one shall generate an Error. Passing a reference list in array iteration context to a function expecting a scalar value shall generate an Error.

4.10 Array

An array is a set of rows each with the same number of columns that contain one or more values. There is a maximum of one value per intersection of row and column. The intersection of a row and column may contain no value.

4.11 Pseudotypes

4.11.1 General

Many functions require a type or a set of types with special properties, and/or process them specially. For example, a "Database" requires headers that are the field names. These specialized types are called *pseudotypes*.

4.11.2 Scalar

A *Scalar* value is a value that has a *single* value. A reference to more than one cell is *not* a scalar (by itself), and is converted to one as described below. An array with more than one element is not a scalar. The types Number (including a complex number), Logical, and Text are scalars.

4.11.3 DateParam

A DateParam is a value that is either a Number (interpreted as a serial number; 4.3.3) or Text; text is automatically converted to a date value. 6.3.15

4.11.4 TimeParam

A TimeParam is a value that is either a Number (interpreted as a serial number; 4.3.2) or Text; text is automatically converted to a time value (fraction of a day). 6.3.16

4.11.5 Integer

An integer is a subtype of Number that has no fractional value. An integer X is equal to $\text{INT}(X)$. Division of one integer by another integer may produce a non-integer.

4.11.6 TextOrNumber

TextOrNumber is a value that is either a Number or Text.

4.11.7 Basis

4.11.7.1 General

A basis is a subtype of Integer that specifies the day-count convention to be used in a calculation.

This standard defines five day-count conventions, corresponding to widely used current and historical accounting conventions. Each of these five bases defines two things:

1. How to calculate the number of days between two dates, *date1* and *date2*.
2. How to calculate the number of days in each year between two dates, *date1* and *date2*.

Historically day-count bases used the naming convention x/y , which indicated that the convention assumed x days per month and y days per year. These names are given for reference purposes.

<i>Date Basis</i>	<i>Historical Name</i>	<i>Day Count</i>	<i>Days in Year</i>
0	US (NASD) 30/360	Procedure A, 4.11.7.3	Procedure D, 4.11.7.6
1	Actual/Actual	Procedure B, 4.11.7.4	Procedure E, 4.11.7.7
2	Actual/360	Procedure B, 4.11.7.4	Procedure D, 4.11.7.6
3	Actual/365	Procedure B, 4.11.7.4	Procedure F, 4.11.7.8
4	European 30/360	Procedure C, 4.11.7.5	Procedure D, 4.11.7.6

4.11.7.2 Procedural Notation

The day-count procedures are expressed using notations defined as:

- `day(date)` returns the day of the month for the given date value, an integer from 1 to 31
- `month(date)` returns the month of a given date value, an integer from 1-12
- `year(date)` returns the year of the given date value
- `truncate(date)` truncates any fractional (hours, minutes, seconds) of a date value and returns the whole date portion.
- Binary comparison operators `date1>date2` and `date1 == date2`
- `is-leap-year(year)` returns true if year is a leap year, otherwise false.

Note: Some of the day count procedures use intermediate results that contain counter-factual dates, such as February 30th. This is not an error. The above functions work on such dates as well, e.g., `day(February 30th) == 30`.

4.11.7.3 Procedure A

1. `truncate(date1), truncate(date2)`
2. If `date1==date2` return 0
3. If `date1> date2`, then swap the values of `date1` and `date2`.

4. If $\text{day}(\text{date1}) == 31$ then subtract 1 day from date1
5. If $\text{day}(\text{date1}) == 30$ and $\text{day}(\text{date2}) == 31$ then subtract 1 day from date2
6. If both date1 and date2 are the last day of February, change date2 to the 30th of the month.
7. If date1 is the last day of February, change it to the 30th of the month.
8. Return $(\text{year}(\text{date2}) * 360 + \text{month}(\text{date2}) * 30 + \text{day}(\text{date2})) - (\text{year}(\text{date1}) * 360 + \text{month}(\text{date1}) * 30 + \text{day}(\text{date1}))$.

4.11.7.4 Procedure B

1. $\text{truncate}(\text{date1})$, $\text{truncate}(\text{date2})$
2. If $\text{date1} > \text{date2}$, then swap the values of date1 and date2.
3. Return the actual numbers of days between date1 and date2, inclusive of date1, but not inclusive of date2.

4.11.7.5 Procedure C

1. $\text{truncate}(\text{date1})$, $\text{truncate}(\text{date2})$
2. If $\text{date1} == \text{date2}$ return 0
3. If $\text{date1} > \text{date2}$, then swap the values of date1 and date2.
4. If $\text{day}(\text{date1}) == 31$ then subtract 1 from date1
5. If $\text{day}(\text{date2}) == 31$ then subtract 1 from date 2
6. Return $(\text{year}(\text{date2}) * 360 + \text{month}(\text{date2}) * 30 + \text{day}(\text{date2})) - (\text{year}(\text{date1}) * 360 + \text{month}(\text{date1}) * 30 + \text{day}(\text{date1}))$.

4.11.7.6 Procedure D

1. Return 360

4.11.7.7 Procedure E

1. Evaluate A: $\text{year}(\text{date1}) \neq \text{year}(\text{date2})$
2. Evaluate B: $\text{year}(\text{date2}) \neq \text{year}(\text{date1}) + 1$
3. Evaluate C: $\text{month}(\text{date1}) < \text{month}(\text{date2})$
4. Evaluate D: $\text{month}(\text{date1}) == \text{month}(\text{date2})$
5. Evaluate E: $\text{day}(\text{date1}) < \text{day}(\text{date2})$
6. Evaluate F: (A and B) or (A and C) or (A and D and E)
7. If F is true then return the average of the number of days in each year between date1 and date2, inclusive.
8. Otherwise, if A and $\text{is-leap-year}(\text{year}(\text{date1}))$ then return 366
9. Otherwise, if a February 29 occurs between date1 and date2 then return 366
10. Otherwise, if date2 is a February 29, then return 366
11. Otherwise return 365

4.11.7.8 Procedure F

1. Return 365

4.11.8 Criterion

A *criterion* is a single cell Reference, Number or Text. It is used in comparisons with cell contents.

A reference to an empty cell is interpreted as the numeric value 0.

A matching expression can be:

- A Number or Logical value. A matching cell content equals the Number or Logical value.
- A value beginning with a comparator (<, <=, >, >=) 6.4.9 or an infix operator (=, <>). = 6.4.7, <> 6.4.8

For =, if the value is empty it matches empty cells. Empty cell 4.7, = 6.4.7

For <>, if the value is empty it matches non-empty cells. <> 6.4.8

For <>, if the value is not empty it matches any cell content except the value, including empty cells.

Note: "=0" does not match empty cells.

For = and <>, if the value is not empty and can not be interpreted as a Number type or one of its subtypes and the host-defined property HOST-SEARCH-CRITERIA-MUST-APPLY-TO-WHOLE-CELL is true, comparison is against the entire cell contents, if false, comparison is against any subpart of the field that matches the criteria. For = and <>, if the value is not empty and can not be interpreted as a Number type or one of its subtypes 3.4 applies.

- Other Text value. If the host-defined property HOST-SEARCH-CRITERIA-MUST-APPLY-TO-WHOLE-CELL is true, the comparison is against the entire cell contents, if false, comparison is against any subpart of the field that matches the criteria.

4.11.9 Database

A *database* is a rectangular organized set of data. Any database has a set of one or more *fields* that determine the structure of the database. A database has a set of zero or more *records* with data, and each record contains data for every field (though that field may be empty).

Evaluators shall support the use of ranges as databases if they support any database functions. The first row of a range is interpreted as a set of field names.

Note: Field names of type Text and unique without regard to case enhance the interoperability of data. It is also a common expectation that rows following the first row of data are data records that correspond to field names in the first row.

A single cell containing text can be used as a database; if it is, it is a database with a single field and no data records.

4.11.10 Field

A *field* is a value that selects a field in a database; it is also called a *field selector*. If the field selector is Text, it selects the field in the database with the same name.

Evaluators should match the database field name case-insensitively.

If a field selector is a Number, it is a positive integer and used to select the fields. Fields are numbered from left to right beginning with the number 1.

All functions that accept a field parameter shall, when evaluated, return an Error if the selected field does not exist.

4.11.11 Criteria

A *criteria* is a rectangular set of values, with at least one column and two rows, that selects matching records from a database. The first row lists fields against which expressions will be

matched. 4.11.10 Rows after the first row contain fields with expressions for matching against database records.

For a record to be selected from a database, all of the expressions in a criteria row shall match.

A reference to an empty cell is interpreted as the numeric value 0.

- Expressions are matched as per 4.11.8 Criterion.

4.11.12 Sequences (NumberSequence, NumberSequenceList, DateSequence, LogicalSequence, and ComplexSequence)

Some functions accept a sequence, i.e., a value that is to be treated as a sequential series of values. The following are sequences: NumberSequence, NumberSequenceList, DateSequence, LogicalSequence, and ComplexSequence.

When evaluating a function that accepts a sequence, the evaluator shall follow the rules for that sequence as defined in section 6.3. When processing a ReferenceList, the references are processed in order (first, second if any, and so on). In a cuboid, the first sheet is first processed, followed by later sheets (if any) in order. Inside a sheet, it is implementation-defined as to whether the values are processed row-at-a-time or column-at-a-time, but it shall be one of these two processing orders. If processing row-at-a-time, the sequence shall be produced by processing each row in turn, from smallest to largest column value (e.g., A1, B1, C1). If processing column-at-a-time, the sequence shall be produced by processing each column at a time, from the smallest to the largest row value (e.g., A1, A2, A3).

4.11.13 Any

Any represents a value of any type defined in this standard, including Error values.

5 Expression Syntax

5.1 General

The OpenFormula syntax is defined using the BNF notation of the XML specification, chapter 6 [XML1.0]. Each syntax rule is defined using "::<=".

Note: Formulas are typically embedded inside an XML document. When this occurs, characters (such as "<", ">", "'", and "&") shall be escaped, as described in section 2.4 of the XML specification [XML1.0]. In particular, the less-than symbol "<" is typically represented as "<"; the double-quote symbol as """; and the ampersand symbol as "&" (alternatively, a numeric character reference can be used).

5.2 Basic Expressions

Formulas may start with a '=' (EQUALS SIGN, U+003D), which if present may be followed by a "forced recalculate" marker '=' (EQUALS SIGN, U+003D), followed by an expression. If the second '=' (EQUALS SIGN, U+003D) is present, this formula is a "forced recalculation" formula. If a formula is marked as a "forced recalculation" formula, then it should be recalculated whenever one of its predecessors it depends on changes.

Expressed as a grammar in BNF notation, a formula is specified:

```
Formula ::= Intro? Expression
Intro ::= '=' ForceRecalc?
ForceRecalc ::= '='
```

The primary component of a formula is an `Expression`. Formulas are composed of `Expressions`, which may in turn be composed from other `Expressions`.

```
Expression ::=
  Whitespace* (
    Number |
    String |
    Array |
    PrefixOp Expression |
    Expression PostfixOp |
    Expression InfixOp Expression |
    '(' Expression ')' |
    FunctionName '(' ParameterList ')' |
    Reference |
    QuotedLabel |
    AutomaticIntersection |
    NamedExpression |
    Error
  ) Whitespace*
SingleQuoted ::= "'" ([^'] | ''')+ '"'
```

5.3 Constant Numbers

Constant numbers are written using '.' (FULL STOP, U+002E) dot as the decimal separator. Optional "E" or "e" denotes scientific notation. Syntactically, negative numbers are positive numbers with a prefix "-" (HYPHEN-MINUS, U+002D) operator. A constant number is of type `Number`.

```
Number ::= StandardNumber |
  '.' [0-9]+ ([eE] [-+]? [0-9]+)?
StandardNumber ::= [0-9]+ ('.' [0-9]+)? ([eE] [-+]? [0-9]+)?
```

Evaluators should be able to read the `Number` format, which accepts a decimal fraction that starts with decimal point '.' (FULL STOP, U+002E), without a leading zero. Evaluators shall write numbers only using the `StandardNumber` format, which requires a leading digit, and shall not write numbers with a leading '.' (FULL STOP, U+002E).

5.4 Constant Strings

Constant strings are surrounded by double-quote characters (QUOTATION MARK, U+0022); a literal double-quote character "" (QUOTATION MARK, U+0022) as string content is escaped by duplicating it. A constant string is of type `Text`.

```
String ::= '"' ([^"#x00] | '\"')* '"'
```

5.5 Operators

Operators are functions with one or more parameters.

```
PrefixOp ::= '+' | '-'
PostfixOp ::= '%'
InfixOp ::= ArithmeticOp | ComparisonOp | StringOp | ReferenceOp
ArithmeticOp ::= '+' | '-' | '*' | '/' | '^'
ComparisonOp ::= '=' | '<>' | '<' | '>' | '<=' | '>='
StringOp ::= '&'
```

There are three predefined reference operators: reference intersection, reference concatenation, and range. The result of these operators may be a 3-dimensional range, with front-upper-left and back-lower-right corners, or even a list of such ranges in the case of cell concatenation.

```
ReferenceOp ::= IntersectionOp | ReferenceConcatenationOp | RangeOp
IntersectionOp ::= '!'
ReferenceConcatenationOp ::= '~'
RangeOp ::= ':'
```

Table 1 - Operators defines the associativity and precedence of operators, from highest to lowest precedence.

Table 1 - Operators

Associativity	Operator(s)	Comments
left	:	Range.
left	!	Reference intersection ([.A1:.C4]![.B1:.B5] is [.B1:.B4]). Displayed as the space character in some implementations.
left	~	Reference union. Note: Displayed as the function parameter separator in some implementations.
right	+, -	Prefix unary operators, e.g., -5 or -[.A1]. Note that these have a different precedence than add and subtract.
left	%	Postfix unary operator % (divide by 100). Note that this is legal with expressions (e.g., [.B1]%).
left	^	Power (2 ^ 3 is 8).
left	*, /	Multiply, divide.
left	+, -	Binary operations add, subtract. Note that unary (prefix) + and - have a different precedence.
left	&	Binary operation string concatenation. Note that unary (prefix) + and - have a different precedence. Note that "&" shall be es-

		caped when included in an XML document
left	=, <>, <, <=, >, >=	Comparison operators equal to, not equal to, less than, less than or equal to, greater than, greater than or equal to

Note 1: Prefix “-” has a higher precedence than “^”, “^” is left-associative, and reference intersection has a higher precedence than reference union.

Note 2: Prefix “+” and “-” are defined to be right-associative. However, note that typical applications which implement at most the operators defined in this specification (as specified) may implement them as left-associative, because the calculated results will be identical.

Note 3: Precedence can be overridden by using parentheses, so “=2+3*4” computes to 14 but “=(2+3)*4” computes 20. Implementations *should* retain “unnecessary” parentheses and white space, since these are added by people to improve readability.

5.6 Functions and Function Parameters

Functions are called by name, followed by parentheses surrounding a list of parameters. Parameters are separated using the semicolon ‘;’ (SEMICOLON, U+003B) character:

```
FunctionName ::= LetterXML (LetterXML | DigitXML |
    '_' | '.' | CombiningCharXML)*
```

Where LetterXML, DigitXML, and CombiningCharXML are Letter, Digit, and CombiningChar as they are defined in [XML1.0].

Function names are case-insensitive.

Function calls shall be given a parameter list, though it may be empty. An empty list of parameters is considered a call with 0 parameters, not a call with one parameter that happens to be empty. TRUE() is syntactically a function call with 0 parameters. It is syntactically legitimate to provide empty parameters, though functions need not accept empty parameters unless otherwise noted:

```
ParameterList ::= /* empty */ |
    Parameter (Separator EmptyOrParameter)* |
    Separator EmptyOrParameter /* First param empty */
    (Separator EmptyOrParameter)*
EmptyOrParameter ::= /* empty */ Whitespace* | Parameter
Parameter ::= Expression
Separator ::= ';'

```

5.7 Nonstandard Function Names

When writing a document using function(s) not defined in this specification, an evaluator shall include a prefix in such function names to identify the original definer of the function's semantics. When the origin of a function cannot be determined, producers may omit a prefix. Producers may use the prefix to differentiate between different definition types. Evaluators that do not support a function should compute its result as some Error value other than #N/A.

Note: Examples of implementation-defined functions include extension functions included with an implementation, user-defined functions written by users, and 3rd party functions distributed in libraries.

Note: Examples of such names include COM.MICROSOFT.CUBEMEMBER, ORG.OPENOFFICE.STYLE, ORG.GNUMERIC.RANDRAYLEIGH, and COM.LOTUS.V98.FOO.

Evaluators should avoid defining evaluator-unique functions beginning with a top-level domain name followed by a period. Evaluators should avoid defining application functions beginning with “NET.”, “COM.”, “ORG.”, or a country code followed by a period.

Evaluators that do not support a function should compute its result as some Error value other than #N/A.

5.8 References

References refer to a specific cell or set of cells. The syntax for a constant reference:

```
Reference ::= '[' (Source? RangeAddress) | ReferenceError '['
RangeAddress ::=
  SheetLocatorOrEmpty '.' Column Row (':' '.' Column Row )? |
  SheetLocatorOrEmpty '.' Column ':' '.' Column |
  SheetLocatorOrEmpty '.' Row ':' '.' Row |
  SheetLocator '.' Column Row ':' SheetLocator '.' Column Row |
  SheetLocator '.' Column ':' SheetLocator '.' Column |
  SheetLocator '.' Row ':' SheetLocator '.' Row
SheetLocatorOrEmpty ::= SheetLocator | /* empty */
SheetLocator ::= SheetName ( '.' SubtableCell ) *
SheetName ::= QuotedSheetName | '$' ? [ ^ \ ] \ . # '$ ' ] +
QuotedSheetName ::= '$' ? SingleQuoted
SubtableCell ::= ( Column Row ) | QuotedSheetName
ReferenceError ::= "#REF!"
Column ::= '$' ? [ A-Z ] +
Row ::= '$' ? [ 1-9 ] [ 0-9 ] *
Source ::= " " " IRI " " " " # "
CellAddress ::= SheetLocatorOrEmpty '.' Column Row /* Not used
directly */
```

References always begin with '[' (LEFT SQUARE BRACKET, U+005B); this disambiguates cell addresses from function names and named expressions. *SheetNames* include single-quote"" (APOSTROPHE, U+0027) characters by doubling them and having the entire name surrounded by single-quotes. Column labels shall be in uppercase. The syntax supports whole-row and whole-column references. A reference is of type *Reference*.

A *ReferenceError* provides information that a formula evaluates to an Error because of a particular reference having been invalidated by actions that occurred after the formula was validly created.

Columns are named by a sequence of one or more uppercase letters A-Z (U+0041 through U+005A). Columns are named A, B, C, ... X, Y, Z, AA, AB, AC, ... AY, AZ, BA, BB, BC, ... ZX, ZY, ZZ, AAA, AAB, AAC, AAZ, ABA, ABB, and so on.

If a *RangeAddress* does not contain a *Column* element or does not contain a *Row* element, it specifies a cell rectangle (4.8 Reference). If it contains *Row* elements, the cell rectangle starts on the first column and ends on the last column the evaluator supports. If it contains *Column* elements, the cell rectangle starts on the first row and ends on the last row the evaluator supports.

If in a *RangeAddress* the first part (left of ':' colon) contains a *SheetLocator* and the second part (right of ':' colon) does not contain a *SheetLocator*, the second part inherits the *SheetLocator* from the first part.

If a *RangeAddress* contains two different *SheetLocators*, it specifies a cell cuboid (4.8 Reference).

If a *RangeAddress* contains no *SheetLocator*, the current sheet local to the position where the expression is evaluated is referred.

A reference with an explicit row or column value beyond the capabilities of an evaluator shall be computed as an Error, and not as a reference.

Note that references can include a single embedded ":" separator. Evaluators should use references with embedded ":" separators inside the [...] markers, instead of the general-purpose ":" operator, when saving files, and, where there is a choice of cells to join, evaluators should choose the leftmost pair.

The optional *Source* expresses that the reference is to sheets and/or cells in a different location (possibly in a same-document fragment) from that for the formula in which the reference occurs. The optional *Source* is also used for locating Named Expressions (section 5.11).

The IRI portion of Source shall be an IRI reference [RFC3987] conforming to the general syntax IRI-reference rule (section 2.2 of [RFC3987]) after each pair of consecutive single-quote characters (APOSTROPHE, U+0027) is replaced by one single single-quote character.

Note: The escaping of single-quotes as paired single-quotes is because the IRI is enclosed in single quote characters of the Source.

Resolution of the [RFC3987] IRI reference is host-defined behavior. 3.4

5.9 Reference List

A reference list is the result of the Infix Operator Reference Concatenation 6.4.13 '~', the syntax is:

```
ReferenceList ::= Reference (Whitespace* ReferenceConcatenationOp
    Whitespace* Reference)*
```

A reference list can be passed as an argument to functions expecting a reference parameter where passing one reference results in an identical computation as an arbitrary sequence of single references occupying the identical cell range. A reference list cannot be converted to an array.

5.10 Quoted Label

5.10.1 General

A quoted label is Text contained in a table as cell content, either literally or as a formula result.

```
QuotedLabel ::= SingleQuoted
```

A quoted label identifies a column or a row, depending on the label range in which its text appears.

5.10.2 Lookup of Defined Labels

For a `QuotedLabel`, first the cells defined in column label ranges (cell ranges of the `table:label-cell-range-address` attribute (OpenDocument Part 3, 19.660) in the elements `<table:label-range>` (OpenDocument Part 3, 9.4.9) with attribute `table:orientation` (OpenDocument Part 3, 19.690.4) set to `column`) are searched for the string content of `QuotedLabel` (without the quotes). If found, the corresponding column's range of the data cell range of the `table:data-cell-range-address` attribute (OpenDocument Part 3, 19.612) is taken as a reference. If not found, the cells defined in row label ranges (attribute `table:orientation` set to `row`) are searched and if found the corresponding row's range of the data cell range is taken. Label ranges of the current formula's sheet take precedence over label ranges of other sheets if a name occurs in both.

5.10.3 Automatic Lookup of Labels

For a `QuotedLabel` where no defined label is found, an automatic lookup is performed on the sheet where the formula cell resides, if that document setting is enabled (HOST-AUTOMATIC-FIND-LABELS value true).

Matches to the upper left of the formula cell are preferred over other matches, followed by matches with a smaller distance. The following algorithm is used:

Cells on the same sheet as the formula cell are examined **column-wise** from left to right whether they contain the text of `QuotedLabel` (without the quotes). If more than one cell match, the distance and direction from the formula cell's position is taken into account. The distance is calculated by $\text{Distance} = \text{ColumnDifference} * \text{ColumnDifference} + \text{RowDifference} * \text{RowDifference}$ using an idealized layout of square cells. For the direction, during the run two independent match positions are remembered each time Distance is smaller than a previous Distance: *Match2* for positions right of and/or below the formula position

(FormulaColumn < MatchColumn || FormulaRow < MatchRow), *Match1* for all others (not right of and not below). *Match1* also holds the very first match, in case there is only one match or all matches are somewhere below or right of the formula cell. After having found the smallest distances the conditions are:

1. If *Match1* has the smallest distance, that match is taken.
2. Else, *Match2* (right and/or below) has the smallest or an equal distance:
 - 2.1 A match to the upper left (FormulaColumn >= Match1Column && FormulaRow >= Match1Row) takes precedence over matches to other directions.
 - 2.2 Else, if there is no match to the upper left:
 - 2.2.1 If *Match1* is somewhere right of the formula cell (FormulaColumn < Match1Column) it was the first match found in column-wise lookup.
 - 2.2.1.1 If *Match2* is above the formula cell (FormulaRow >= Match2Row) it is to the upper right of the formula cell and either nearer than *Match1* or *Match1* is below. *Match2* is taken.
 - 2.2.1.2 Else *Match2* is below the formula cell and *Match1* is taken.
 - 2.2.2 Else (*Match1* not right of the formula cell => two matches below or below and right) the match with the smallest distance is taken.

If the resulting cell is below or above another cell containing Text a row label is assumed, else a column label is assumed.

Note: Use of automatically looked up column or row labels in expressions impairs interoperability.

5.10.4 Implicit Intersection

For the reference resulting from a single *QuotedLabel* an implicit intersection is generated if the operator or function with which it is used expects a scalar value. The intersection is generated with the current formula's cell position, thus for a column label an intersection is generated with the formula cell's row, for a row label with the formula cell's column.

5.10.5 Automatic Range

When passed as a non-scalar argument (e.g. Array or NumberSequence) to a function, an automatically-looked-up column or row label (not defined label range) is converted to an automatic range reference that is adjusted each time the formula is interpreted. The range is generated from the column below a column label, or the row to the right of a row label, constructed by encompassing contiguous non-empty cells. An empty cell interrupts contiguousness, one empty cell directly below a column label cell or to the right of a row label cell is ignored.

Example:

Table 2 - Automatic Range

Row	Data	Expression	Result	Comment
1	Label	=SUM('Label')	3	Empty cell in row 2 is skipped (two empty cells in row 2 and 3 would not be skipped and would stop the automatic range), empty cell in row 5 stops the automatic range.
2				
3	1			
4	2			
5				
6	8			
7				
8	32			

If any cell content is entered in row 5 the range is regenerated as follows:

Table 3 - Automatic Range

Row	Data	Expression	Result	Comment
1	Label	=SUM('Label')	15	Empty cell in row 2 is skipped, empty cell in row 7 stops the automatic range.
2				
3	1			
4	2			
5	4			
6	8			
7				
8	32			

5.10.6 Automatic Intersection

An automatic intersection may be used to identify the intersection of two quoted labels. Note that this is different from the `IntersectionOp`, which takes two references instead of two labels:

```
AutomaticIntersection ::= QuotedLabel Whitespace* '!!!' Whitespace*
                        QuotedLabel
```

In an automatic intersection, one of the labels identifies a row, the other a column; they may be in either order. Each `QuotedLabel` is looked up as defined above under "Lookup of Defined Labels" and "Automatic Lookup of Labels". If two data cell ranges are found, the intersection of the column's data cell range and the row's data cell range is generated. If the intersection result is not exactly one cell, an Error is generated.

5.11 Named Expressions

A `NamedExpression` references another expression, possibly in a completely different spreadsheet or any other type of document that can be imported into a spreadsheet.

```
NamedExpression ::= SimpleNamedExpression |
                  SheetLocalNamedExpression | ExternalNamedExpression
```

```

SimpleNamedExpression ::= Identifier |
    '$$' (Identifier | SingleQuoted)
SheetLocalNamedExpression ::=
    QuotedSheetName '.' SimpleNamedExpression
ExternalNamedExpression ::=
    Source (SimpleNamedExpression | SheetLocalNamedExpression)

```

Evaluators supporting named expressions shall support Simple Named Expressions that are global to all the sheets in a (spreadsheet) document in the current document. This is a named expression without a *Source*, *QuotedSheetName*, or *SubtableCell*. The type of a named expression is the type of the value that the named expression returns.

Named expressions are case-consistent, meaning that matching is done case-insensitive and identifiers can not differ ONLY in their case. Evaluators should write identifiers with identical case in all locations.

Evaluators may support Sheet-local Named Expressions that are local (attached) to individual sheets. In that case, a non-empty *QuotedSheetName* can be used to reference a sheet-specific named expression. The most specific named expression for a given expression is used. If the *QuotedSheetName* is empty, the search for the named expression begins with the current sheet, then up through the container(s) of the sheet (the same is true if the *QuotedSheetName* rule fragment is not included at all). If there is a non-empty *QuotedSheetName*, search begins with that named sheet, then up through its container(s) for the given name.

Note: There is no syntax for referencing a named expression without first looking at the current sheet's named expressions; where this is a problem, a user can define a blank sheet and reference that sheet as the starting location for finding the named expression.

If a sheetname is not empty, it shall be quoted using the single-quote character "''" (APOSTROPHE, U+0027). While both *Source* and *QuotedSheetName* can begin with the single-quote character, they are distinguished: after the closing single-quote character, a non-empty source shall have the '#' (NUMBER SIGN, U+0023) character as the next non-whitespace character; a non-empty sheetname shall be followed by the '.' (FULL STOP, U+002E) character as the next non-whitespace character.

Expressions should limit the names of their identifiers to only ([UNICODE]) letters, underscores, and digits, not including patterns that look like cell references or the words True or False.

```

Identifier ::= ( LetterXML
    (LetterXML | DigitXML | '_' | CombiningCharXML)* )
    - ( [A-Za-z][0-9]+ )
    - ([Tt][Rr][Uu][Ee]) - ([Ff][Aa][Ll][Ss][Ee])

```

5.12 Constant Errors

Evaluators shall support the Error named #N/A. Evaluators may support other Errors. Evaluators may allow entry of errors directly, parse them and recognize them as Errors. Functions shall propagate Errors unless stated otherwise.

Error names shall have the following syntax:

```

Error ::= '#' [A-Z0-9]+ ([!?] | ('/' ([A-Z] | ([0-9] [!?])))

```

Specific Errors are indicated by their corresponding names.

Table 4 is a list of Errors that are used by several existing implementations.

Table 4 - Possible Other Constant Error Names

Name	Comments
#DIV/0!	Attempt to divide by zero, including division by an empty cell. ERROR.TYPE of 2 6.13.11
#NAME?	Unrecognized/deleted name. ERROR.TYPE of 5.

#N/A	Not available. ISNA() applied to this value will return TRUE. Lookup functions which failed, and NA(), return this value. ERROR.TYPE of 7.
#NULL!	Intersection of ranges produced zero cells. ERROR.TYPE of 1.
#NUM!	Failed to meet domain constraints (e.g., input was too large or too small). ERROR.-TYPE of 6.
#REF!	Reference to invalid cell (e.g., beyond the application's abilities). ERROR.TYPE of 4.
#VALUE!	Parameter is wrong type. ERROR.TYPE of 3.

Evaluators may implement other Errors.

An unknown Error name shall be mapped into an Error supported by the evaluator when read (e.g., the application's equivalent of #NAME?), though an evaluator may warn the user if this has or will take place. It is desirable to preserve the original specific Error name when writing an Error back out, where possible, but for Errors other than #N/A evaluators may write a different Error for a formula than they did when reading it. Whitespace shall not be included in an Error name.

Evaluators should use a human-comprehensible name, not a numeric id, for Error names they write.

5.13 Inline Arrays

Inline arrays are enclosed with curly braces. Inside, they contain one or more rows, with each row separated by a row separator:

```
Array ::= '{' MatrixRow ( RowSeparator MatrixRow )* '}'
MatrixRow ::= Expression ( ';' Expression )*
RowSeparator ::= '|'
```

Evaluators that support inline arrays shall accept a matrix with one or more rows, each with one or more columns, with the same number of columns in each row, with constant values for each expression. Evaluators that do not support inline arrays, or cannot support a particular use permitted by this syntax, should compute an Error value for such arrays. An inline array is of type Array.

Note: Expression authors should be aware that use of `Expression` other than constant `Number` or constant `String` may impair interoperability.

5.14 Whitespace

```
Whitespace ::= #x20 | #x09 | #x0a | #x0d
```

For calculation purposes, whitespace is ignored unless it is inside the contents of string constants or text surrounded by single quotes. Evaluators shall ignore any whitespace characters before and/or after any operators, constant numbers, constant strings, constant errors, inline arrays, parentheses used for controlling precedence, and the closing parenthesis of a function call. Whitespace shall be ignored following the initial equal sign(s). Whitespace shall be ignored just before a function name, but whitespace shall not separate a function name from its initial opening parenthesis. Whitespace shall not be used in the interior of a terminating grammar rule (a rule that references no other rule other than character sets, internally or externally-defined), unless specifically permitted by the terminating grammar rule, since these rules define the lexical properties of a component. Evaluators shall not write formulas with whitespace embedded in any unquoted identifier, constant `Number`, or constant `Error`. Evaluators shall treat SPACE (U+0020), CHARACTER TABULATION (U+0009), LINE FEED (U+000A), and CARRIAGE RETURN (U+000D) as whitespace characters.

An embedded line break shall be represented by a single LINE FEED character (U+000A), *not* by a carriage return-linefeed pair. When embedded in an XML attribute the linefeed character is represented as "
".

Evaluators should retain whitespace entered by the original formula creator and use it when saving or presenting the formula, and should not add additional whitespace unless directed to do so during the process of editing a formula.

6 Standard Operators and Functions

6.1 General

OpenFormula defines commonly used operators and functions.

Function names ignore case. Evaluators should write function names in all uppercase letters when writing OpenFormula formulas.

Unless otherwise noted, if any value being provided is an Error, the result is an Error; if more than one Error is provided, one of them is returned (evaluators should return the leftmost Error result).

6.2 Common Template for Functions and Operators

For every function or operator, the following are defined in this specification:

- **Name:** The function/operator name.
- **Summary:** One sentence briefly describing the function or operator.
- **Syntax:**
 - Parameter names are shown in order, with each parameter prefixed by the type or pseudotype of that parameter. If the type has multiple names separated by “|”, then any of those types are permitted.
 - A { ... } indicates a list of zero or more parameters, separated by the function parameter separator character.
 - A { ... } followed by a superscripted + indicates a list of one or more parameters, separated by the function parameter separator character.
 - Components surrounded by [...] are optional. Optional components may be omitted.
 - An optional parameter followed by the = symbol has the default value given after the equal sign.
 - Parameters are separated with a semicolon (“;”), as per the OpenFormula expression syntax 5.6.

When a function is given a value of a different type, the parameters are first converted using the implicit conversion rules before the function operates on its parameters.

Evaluators may extend functions by permitting fewer or additional parameters, which documents may use. Extended functions may result in a lack of interoperability.

- **Returns:** Return type (e.g., Number, Text, Logical, Reference).
- **Constraints:** A description of constraints, in addition to the constraints imposed by the parameter types. If there are no additional constraints beyond those imposed by the parameter types, this is “None”. If a constraint is not met, the function/operator shall return an Error unless otherwise noted.
- **Semantics:** This text describes what the function/operator does.

If a parameter is a pseudotype, but the provided value fails to meet the requirements for that type, the behavior is implementation-defined.

Note: Functions and operators are defined by mathematical formulas or by an OpenFormula formula. Formulas define the *correct result*, and *not* the algorithm for calculation. Since computing systems have limited precision and range of numbers, some functions *cannot* or *should not* be naively implemented as their formulas suggest. This specification defines the