

UNISYSDBMS

1

Generated by Doxygen 1.8.1.1

Tue Aug 14 2012 16:26:10

Contents

1	UniSysDB library	1
1.1	Introduction	1
1.2	First Tutorial	1
1.3	General usage: How to include the XMLParser library inside your project.	1
2	Module Index	3
2.1	Modules	3
3	Namespace Index	5
3.1	Namespace List	5
4	Class Index	7
4.1	Class Hierarchy	7
5	Class Index	9
5.1	Class List	9
6	File Index	13
6.1	File List	13
7	Module Documentation	15
7.1	The XML parser	15
7.1.1	Detailed Description	16
7.2	Parsing XML files/strings to an XMLNode structure and Rendering XMLNode's to files/string.	17
7.2.1	Detailed Description	17
7.2.2	Function Documentation	17
7.2.2.1	createXMLString	17
7.2.2.2	getError	18
7.2.2.3	guessCharEncoding	18
7.2.2.4	openFileHelper	18
7.2.2.5	parseFile	18
7.2.2.6	parseString	19
7.2.2.7	setGlobalOptions	19
7.2.2.8	writeToFile	20

7.3	Navigate the XMLNode structure	21
7.3.1	Detailed Description	22
7.3.2	Function Documentation	22
7.3.2.1	deepCopy	22
7.3.2.2	emptyNode	22
7.3.2.3	enumContents	22
7.3.2.4	getAttribute	22
7.3.2.5	getAttribute	22
7.3.2.6	getAttribute	23
7.3.2.7	getAttributeName	23
7.3.2.8	getAttributeValue	23
7.3.2.9	getChildNode	23
7.3.2.10	getChildNode	23
7.3.2.11	getChildNode	23
7.3.2.12	getChildNodeByPath	23
7.3.2.13	getChildNodeByPathNonConst	23
7.3.2.14	getChildNodeWithAttribute	23
7.3.2.15	getClear	23
7.3.2.16	getName	23
7.3.2.17	getParentNode	24
7.3.2.18	getText	24
7.3.2.19	isAttributeSet	24
7.3.2.20	isDeclaration	24
7.3.2.21	isEmpty	24
7.3.2.22	nAttribute	24
7.3.2.23	nChildNode	24
7.3.2.24	nChildNode	24
7.3.2.25	nClear	24
7.3.2.26	nElement	24
7.3.2.27	nText	24
7.4	Create or Update the XMLNode structure	25
7.4.1	Detailed Description	25
7.5	Creating from scratch a XMLNode structure	26
7.5.1	Detailed Description	26
7.5.2	Function Documentation	26
7.5.2.1	addAttribute	26
7.5.2.2	addChild	26
7.5.2.3	addChild	26
7.5.2.4	addClear	27
7.5.2.5	addText	27

7.5.2.6	createXMLTopNode	27
7.6	Updating Nodes	28
7.6.1	Detailed Description	28
7.6.2	Function Documentation	28
7.6.2.1	updateAttribute	28
7.6.2.2	updateAttribute	28
7.6.2.3	updateAttribute	29
7.6.2.4	updateClear	29
7.6.2.5	updateClear	29
7.6.2.6	updateClear	29
7.6.2.7	updateName	29
7.6.2.8	updateText	29
7.6.2.9	updateText	29
7.7	Deleting Nodes or Attributes	30
7.7.1	Detailed Description	30
7.7.2	Function Documentation	30
7.7.2.1	deleteAttribute	30
7.7.2.2	deleteAttribute	30
7.7.2.3	deleteAttribute	31
7.7.2.4	deleteClear	31
7.7.2.5	deleteClear	31
7.7.2.6	deleteClear	31
7.7.2.7	deleteNodeContent	31
7.7.2.8	deleteText	31
7.7.2.9	deleteText	31
7.8	???_WOSD functions.	32
7.8.1	Detailed Description	33
7.8.2	Function Documentation	33
7.8.2.1	addAttribute_WOSD	33
7.8.2.2	addChild_WOSD	33
7.8.2.3	addClear_WOSD	33
7.8.2.4	addText_WOSD	33
7.8.2.5	createXMLTopNode_WOSD	33
7.8.2.6	updateAttribute_WOSD	34
7.8.2.7	updateAttribute_WOSD	34
7.8.2.8	updateAttribute_WOSD	34
7.8.2.9	updateClear_WOSD	34
7.8.2.10	updateClear_WOSD	34
7.8.2.11	updateClear_WOSD	34
7.8.2.12	updateName_WOSD	34

7.8.2.13	updateText_WOSD	34
7.8.2.14	updateText_WOSD	34
7.9	Position helper functions (use in conjunction with the update&add functions)	35
7.9.1	Detailed Description	35
7.9.2	Function Documentation	35
7.9.2.1	positionOfChildNode	35
7.9.2.2	positionOfChildNode	35
7.9.2.3	positionOfChildNode	35
7.9.2.4	positionOfClear	35
7.9.2.5	positionOfClear	35
7.9.2.6	positionOfClear	35
7.9.2.7	positionOfText	35
7.9.2.8	positionOfText	36
7.10	String Allocation/Free functions	37
7.10.1	Detailed Description	37
7.10.2	Function Documentation	37
7.10.2.1	freeXMLString	37
7.10.2.2	stringDup	37
7.11	ato? like functions	38
7.11.1	Detailed Description	38
7.11.2	Function Documentation	38
7.11.2.1	xmltoa	38
7.11.2.2	xmltob	38
7.11.2.3	xmltoc	38
7.11.2.4	xmltof	38
7.11.2.5	xmltoi	38
7.11.2.6	xmltol	38
7.12	Helper class to create XML files using "printf", "fprintf", "cout",... functions.	39
7.12.1	Detailed Description	39
7.12.2	Typedef Documentation	39
7.12.2.1	ToXMLStringTool	39
7.13	Helper class to include binary data inside XML strings using "Base64 encoding".	40
7.13.1	Detailed Description	40
7.13.2	Typedef Documentation	40
7.13.2.1	XMLParserBase64Tool	40
8	Namespace Documentation	41
8.1	unisys Namespace Reference	41
8.1.1	Detailed Description	44
8.1.2	Enumeration Type Documentation	44

8.1.2.1	dbStatus	44
8.1.3	Function Documentation	44
8.1.3.1	BSONToset	44
8.1.3.2	BSONTovector	44
8.1.3.3	setMiriamURN	44
8.1.3.4	toBSONArray	44
9	Class Documentation	45
9.1	unisys::Annotation Class Reference	45
9.1.1	Detailed Description	48
9.1.2	Constructor & Destructor Documentation	48
9.1.2.1	Annotation	48
9.1.2.2	Annotation	48
9.1.3	Member Function Documentation	48
9.1.3.1	initField	48
9.1.3.2	setDefinition	48
9.1.3.3	setDefinition	48
9.1.3.4	setEvidence	48
9.2	unisys::BatchInsert Class Reference	49
9.2.1	Constructor & Destructor Documentation	51
9.2.1.1	BatchInsert	51
9.2.1.2	BatchInsert	51
9.2.2	Member Function Documentation	51
9.2.2.1	insert	51
9.2.2.2	insert	51
9.3	unisys::BiochemicalReaction Class Reference	51
9.3.1	Detailed Description	54
9.3.2	Constructor & Destructor Documentation	54
9.3.2.1	BiochemicalReaction	54
9.3.2.2	BiochemicalReaction	54
9.3.3	Member Function Documentation	54
9.3.3.1	addLeft	54
9.3.3.2	addRight	54
9.3.3.3	initField	55
9.3.3.4	setConversionDirection	55
9.3.3.5	toRXNString	55
9.4	unisys::BiochemicalReactionWithTransport Class Reference	55
9.4.1	Detailed Description	58
9.4.2	Constructor & Destructor Documentation	58
9.4.2.1	BiochemicalReactionWithTransport	58

9.4.2.2	BiochemicalReactionWithTransport	58
9.4.3	Member Function Documentation	58
9.4.3.1	addExport	58
9.4.3.2	addImport	58
9.4.3.3	addLeftIn	58
9.4.3.4	addLeftOut	59
9.4.3.5	addRightIn	59
9.4.3.6	addRightOut	59
9.4.3.7	initField	59
9.4.3.8	setInteractionKey	59
9.4.3.9	toRXNString	59
9.5	unisys::BioObject Class Reference	59
9.5.1	Detailed Description	62
9.5.2	Constructor & Destructor Documentation	62
9.5.2.1	BioObject	62
9.5.3	Member Function Documentation	62
9.5.3.1	addAnnotation	62
9.5.3.2	addDataXref	62
9.5.3.3	addEvidence	62
9.5.3.4	addSynonyms	62
9.5.3.5	createIdPair	62
9.5.3.6	initField	62
9.5.3.7	setComment	63
9.5.3.8	setDataPrimarySource	63
9.5.3.9	setMainName	63
9.5.3.10	setType	63
9.6	unisys::BIOPAX2 Class Reference	63
9.6.1	Detailed Description	65
9.6.2	Member Typedef Documentation	65
9.6.2.1	biopaxObjMap	65
9.6.3	Constructor & Destructor Documentation	65
9.6.3.1	BIOPAX2	65
9.6.3.2	BIOPAX2	65
9.6.4	Member Function Documentation	65
9.6.4.1	begin	65
9.6.4.2	end	65
9.6.4.3	getOWLNode	65
9.6.5	Member Data Documentation	65
9.6.5.1	biopax	65
9.6.5.2	objects	65

9.7	unisys::BIOPAX_obj Class Reference	66
9.7.1	Detailed Description	67
9.7.2	Constructor & Destructor Documentation	67
9.7.2.1	BIOPAX_obj	67
9.7.2.2	BIOPAX_obj	67
9.7.3	Member Function Documentation	67
9.7.3.1	getAttribute	67
9.7.3.2	getAttributeName	67
9.7.3.3	getAttributeValue	67
9.7.3.4	getChildNode	67
9.7.3.5	getChildNode	67
9.7.3.6	getChildNode	67
9.7.3.7	getId	67
9.7.3.8	getPropertyValue	67
9.7.3.9	getType	68
9.7.3.10	nAttribute	68
9.7.3.11	nChildNode	68
9.7.3.12	nChildNode	68
9.7.4	Member Data Documentation	68
9.7.4.1	object	68
9.8	unisys::BioSource Class Reference	68
9.8.1	Detailed Description	71
9.8.2	Constructor & Destructor Documentation	71
9.8.2.1	BioSource	71
9.8.2.2	BioSource	71
9.8.2.3	BioSource	71
9.8.3	Member Function Documentation	71
9.8.3.1	initField	71
9.8.3.2	setCellState	72
9.8.3.3	setCellState	72
9.8.3.4	setCellType	72
9.8.3.5	setCellType	72
9.8.3.6	setSciName	72
9.8.3.7	setStain	72
9.8.3.8	setTaxonomicId	72
9.8.3.9	setTaxonomicId	72
9.8.3.10	setTissueType	72
9.8.3.11	setTissueType	72
9.9	unisys::CellularLocation Class Reference	72
9.9.1	Detailed Description	75

9.9.2	Constructor & Destructor Documentation	75
9.9.2.1	CellularLocation	75
9.9.2.2	CellularLocation	75
9.9.2.3	CellularLocation	75
9.9.2.4	CellularLocation	75
9.9.3	Member Function Documentation	75
9.9.3.1	initField	75
9.9.3.2	setEvidence	75
9.9.3.3	setLocation	75
9.9.3.4	setLocation	75
9.10	unisys::ChEBIOWL Class Reference	76
9.10.1	Detailed Description	77
9.10.2	Constructor & Destructor Documentation	77
9.10.2.1	ChEBIOWL	77
9.10.2.2	ChEBIOWL	77
9.10.3	Member Function Documentation	77
9.10.3.1	getClassList	77
9.10.3.2	getFileAnnotation	77
9.10.3.3	getVersion	77
9.10.3.4	parse	77
9.10.4	Member Data Documentation	77
9.10.4.1	classList	77
9.10.4.2	owlOntology	77
9.11	unisys::ChEBIOWLClass Class Reference	78
9.11.1	Detailed Description	79
9.11.2	Constructor & Destructor Documentation	80
9.11.2.1	ChEBIOWLClass	80
9.11.2.2	ChEBIOWLClass	80
9.11.3	Member Function Documentation	80
9.11.3.1	getAnnotationProperties	80
9.11.3.2	getFormula	80
9.11.3.3	getId	80
9.11.3.4	getInChi	80
9.11.3.5	getInChiKey	80
9.11.3.6	getLabel	80
9.11.3.7	getPropertyList	80
9.11.3.8	getSMILES	80
9.11.3.9	getSpAnnotationProperty	80
9.11.3.10	getSpSubClassOf	80
9.11.3.11	getSubClassOf	80

9.11.3.12 parse	80
9.11.3.13 parseAnnotation	80
9.11.3.14 parseSubClassOf	80
9.11.4 Member Data Documentation	80
9.11.4.1 AnnotationProperties	80
9.11.4.2 chebild	80
9.11.4.3 inchi	80
9.11.4.4 inchiKey	81
9.11.4.5 label	81
9.11.4.6 smiles	81
9.11.4.7 subClassOf	81
9.12 unisys::ClassBean Class Reference	81
9.12.1 Detailed Description	83
9.12.2 Member Typedef Documentation	83
9.12.2.1 relationStruct	83
9.12.3 Constructor & Destructor Documentation	84
9.12.3.1 ClassBean	84
9.12.3.2 ClassBean	84
9.12.4 Member Function Documentation	84
9.12.4.1 getDefinitions	84
9.12.4.2 getFullId	84
9.12.4.3 getId	84
9.12.4.4 getLabel	84
9.12.4.5 getRelations	84
9.12.4.6 getSynonyms	84
9.12.4.7 getType	84
9.12.5 Member Data Documentation	84
9.12.5.1 definitions	84
9.12.5.2 fullId	84
9.12.5.3 id	85
9.12.5.4 label	85
9.12.5.5 relations	85
9.12.5.6 synonyms	85
9.12.5.7 type	85
9.13 unisys::Complex Class Reference	85
9.13.1 Detailed Description	88
9.13.2 Constructor & Destructor Documentation	88
9.13.2.1 Complex	88
9.13.2.2 Complex	88
9.13.3 Member Function Documentation	88

9.13.3.1 addComplexMember	88
9.13.3.2 addSpecificKineticParameter	88
9.13.3.3 initField	88
9.14 unisys::ConnectionError Class Reference	88
9.14.1 Member Function Documentation	90
9.14.1.1 what	90
9.15 unisys::Control Class Reference	91
9.15.1 Detailed Description	93
9.15.2 Constructor & Destructor Documentation	93
9.15.2.1 Control	93
9.15.2.2 Control	93
9.15.3 Member Function Documentation	93
9.15.3.1 addControlType	93
9.15.3.2 addModificationType	93
9.15.3.3 addPhenotype	93
9.15.3.4 initField	93
9.15.3.5 setControlled	94
9.15.3.6 setController	94
9.16 unisys::Conversion Class Reference	94
9.16.1 Detailed Description	97
9.16.2 Constructor & Destructor Documentation	97
9.16.2.1 Conversion	97
9.16.3 Member Function Documentation	97
9.16.3.1 addKineticLaw	97
9.16.3.2 initField	97
9.16.3.3 setFunctional	97
9.16.3.4 setSpontaneous	97
9.17 unisys::Database Class Reference	97
9.17.1 Detailed Description	99
9.17.2 Constructor & Destructor Documentation	99
9.17.2.1 Database	99
9.17.3 Member Function Documentation	99
9.17.3.1 connect	99
9.17.3.2 fetchDBRef	99
9.17.3.3 getdbname	99
9.17.3.4 setdbname	99
9.17.4 Friends And Related Function Documentation	99
9.17.4.1 Updater	99
9.17.5 Member Data Documentation	99
9.17.5.1 dbname	99

9.18 unisys::DataError Class Reference	100
9.18.1 Constructor & Destructor Documentation	101
9.18.1.1 DataError	101
9.18.2 Member Function Documentation	101
9.18.2.1 what	102
9.19 unisys::DataObj Class Reference	102
9.19.1 Constructor & Destructor Documentation	105
9.19.1.1 DataObj	105
9.19.1.2 DataObj	105
9.19.2 Member Function Documentation	105
9.19.2.1 addFieldSet	105
9.19.2.2 addWithCheck	105
9.19.2.3 addWithOutCheck	105
9.19.2.4 append	105
9.19.2.5 append	105
9.19.2.6 appendArray	105
9.19.2.7 appendArray	105
9.19.2.8 appendArray	105
9.19.2.9 appendAs	105
9.19.2.10 appendAsNumber	105
9.19.2.11 appendCode	105
9.19.2.12 appendDBRef	105
9.19.2.13 appendElements	105
9.19.2.14 appendElementsUnique	105
9.19.2.15 appendNull	105
9.19.2.16 appendRegex	105
9.19.2.17 appendTimeStamp	105
9.19.2.18 appendTimeT	105
9.19.2.19 genOID	106
9.19.2.20 getField	106
9.19.2.21 hasField	106
9.19.2.22 initField	106
9.19.2.23 isOwned	106
9.19.2.24 isValid	106
9.19.2.25 md5	106
9.19.2.26 operator<	106
9.19.2.27 operator<=	106
9.19.2.28 operator==	106
9.19.2.29 operator>	106
9.19.2.30 operator>=	106

9.19.2.31 removeField	106
9.19.2.32 set	106
9.19.2.33 set	106
9.19.2.34 set	106
9.19.2.35 set	106
9.19.2.36 set	106
9.19.2.37 set	106
9.19.2.38 set	106
9.19.2.39 sort	106
9.19.2.40 toBSONObj	106
9.19.2.41 toString	107
9.19.2.42 valid	107
9.19.3 Member Data Documentation	107
9.19.3.1 data	107
9.19.3.2 fieldSet	107
9.19.3.3 requireField	107
9.20 unisys::DNA Class Reference	107
9.20.1 Detailed Description	110
9.20.2 Constructor & Destructor Documentation	110
9.20.2.1 DNA	110
9.20.2.2 DNA	110
9.20.3 Member Function Documentation	110
9.20.3.1 addDNARegion	110
9.20.3.2 initField	110
9.20.3.3 setLength	110
9.20.3.4 setSource	110
9.21 unisys::DNARegion Class Reference	111
9.21.1 Detailed Description	113
9.21.2 Constructor & Destructor Documentation	113
9.21.2.1 DNARegion	113
9.21.2.2 DNARegion	113
9.21.3 Member Function Documentation	113
9.21.3.1 addTranscriptionProduct	113
9.21.3.2 initField	113
9.21.3.3 setDNASource	113
9.21.3.4 setPosition	113
9.22 unisys::Evidence Class Reference	114
9.22.1 Detailed Description	116
9.22.2 Constructor & Destructor Documentation	116
9.22.2.1 Evidence	116

9.22.2.2 Evidence	116
9.22.3 Member Function Documentation	116
9.22.3.1 initField	116
9.22.3.2 insertPublicationSupport	116
9.22.3.3 insertPublicationSupport	116
9.22.3.4 setBiologicalSource	116
9.22.3.5 setConfidence	117
9.22.3.6 setEvidenceSource	117
9.22.3.7 setEvidenceSource	117
9.22.3.8 setExperimentMethod	117
9.22.3.9 setExperimentMethod	117
9.23 unisys::GeneticInteraction Class Reference	117
9.23.1 Detailed Description	120
9.23.2 Constructor & Destructor Documentation	120
9.23.2.1 GeneticInteraction	120
9.23.2.2 GeneticInteraction	120
9.23.3 Member Function Documentation	120
9.23.3.1 initField	120
9.23.3.2 setInteractionType	120
9.23.3.3 setPhenotype	120
9.24 unisys::IdRef Class Reference	121
9.24.1 Detailed Description	123
9.24.2 Constructor & Destructor Documentation	123
9.24.2.1 IdRef	123
9.24.2.2 IdRef	123
9.24.2.3 IdRef	123
9.24.3 Member Function Documentation	123
9.24.3.1 getId	123
9.24.3.2 getNS	123
9.24.3.3 initField	123
9.24.3.4 initMember	124
9.24.3.5 isValid	124
9.25 unisys::Interaction Class Reference	124
9.25.1 Detailed Description	127
9.25.2 Constructor & Destructor Documentation	127
9.25.2.1 Interaction	127
9.25.3 Member Function Documentation	127
9.25.3.1 callInteractionKey	127
9.25.3.2 getParticipant	127
9.25.3.3 initField	127

9.25.3.4	setInteractionKey	127
9.25.3.5	setParticipant	127
9.25.3.6	toBSONObj	127
9.26	unisys::IntIdRef Class Reference	128
9.26.1	Detailed Description	130
9.26.2	Constructor & Destructor Documentation	130
9.26.2.1	IntIdRef	130
9.26.2.2	IntIdRef	130
9.26.2.3	IntIdRef	130
9.26.3	Member Function Documentation	130
9.26.3.1	initMember	130
9.26.3.2	setId	130
9.27	unisys::KineticParameter Class Reference	130
9.27.1	Detailed Description	133
9.27.2	Constructor & Destructor Documentation	133
9.27.2.1	KineticParameter	133
9.27.2.2	KineticParameter	133
9.27.3	Member Function Documentation	133
9.27.3.1	initField	133
9.27.3.2	operator<	133
9.27.3.3	setEvidence	134
9.27.3.4	setTerm	134
9.27.3.5	setTerm	134
9.27.3.6	setUnit	134
9.27.3.7	setValue	134
9.28	unisys::Literal Class Reference	134
9.28.1	Detailed Description	136
9.28.2	Constructor & Destructor Documentation	136
9.28.2.1	Literal	136
9.28.3	Member Function Documentation	136
9.28.3.1	initField	136
9.28.3.2	setComment	136
9.29	unisys::LiteralBSON Class Reference	136
9.29.1	Detailed Description	137
9.29.2	Constructor & Destructor Documentation	137
9.29.2.1	LiteralBSON	137
9.30	unisys::MathML Class Reference	137
9.30.1	Detailed Description	140
9.30.2	Constructor & Destructor Documentation	140
9.30.2.1	MathML	140

9.30.2.2	MathML	140
9.30.3	Member Function Documentation	140
9.30.3.1	initField	140
9.30.3.2	setEvidence	140
9.30.3.3	setMathMLDetail	140
9.31	unisys::Metadata Class Reference	140
9.31.1	Detailed Description	143
9.31.2	Constructor & Destructor Documentation	143
9.31.2.1	Metadata	143
9.31.2.2	Metadata	143
9.31.2.3	Metadata	143
9.31.3	Member Function Documentation	143
9.31.3.1	initField	143
9.31.3.2	setDetail	143
9.32	unisys::Miriam Class Reference	143
9.32.1	Detailed Description	145
9.32.2	Constructor & Destructor Documentation	145
9.32.2.1	Miriam	145
9.32.2.2	Miriam	145
9.32.3	Member Function Documentation	146
9.32.3.1	increaseVer	146
9.32.3.2	isValid	146
9.32.3.3	operator++	146
9.32.3.4	operator<	146
9.32.3.5	operator==	146
9.32.3.6	set	146
9.32.3.7	setUri	146
9.32.3.8	toDBId	146
9.32.3.9	toDBIdWithVer	146
9.32.3.10	told	146
9.32.3.11	toURI	146
9.32.3.12	toURIWithVer	146
9.32.4	Member Data Documentation	147
9.32.4.1	id	147
9.32.4.2	ns	147
9.32.4.3	version	147
9.33	unisys::MolecularInteraction Class Reference	147
9.33.1	Detailed Description	150
9.33.2	Constructor & Destructor Documentation	150
9.33.2.1	MolecularInteraction	150

9.33.2.2	MolecularInteraction	150
9.33.3	Member Function Documentation	150
9.33.3.1	addLeft	150
9.33.3.2	addRight	150
9.33.3.3	initField	150
9.33.3.4	setKineticLaw	150
9.34	unisys::Object Class Reference	151
9.34.1	Detailed Description	153
9.34.2	Constructor & Destructor Documentation	153
9.34.2.1	Object	153
9.34.2.2	Object	153
9.34.3	Member Function Documentation	153
9.34.3.1	addOntoRelationship	153
9.34.3.2	addRelation	153
9.34.3.3	addRelation	153
9.34.3.4	createRelationInsert	153
9.34.3.5	createRelationRemove	153
9.34.3.6	createTrack	153
9.34.3.7	getRelation	153
9.34.3.8	initField	154
9.34.3.9	setId	154
9.34.3.10	setId	154
9.34.4	Member Data Documentation	154
9.34.4.1	relationMap	154
9.35	unisys::OBOXML Class Reference	154
9.35.1	Detailed Description	156
9.35.2	Constructor & Destructor Documentation	156
9.35.2.1	OBOXML	156
9.35.2.2	OBOXML	156
9.35.3	Member Function Documentation	156
9.35.3.1	stanzasBegin	156
9.35.3.2	stanzasBegin	156
9.35.3.3	stanzasEnd	156
9.35.3.4	stanzasEnd	156
9.35.4	Member Data Documentation	156
9.35.4.1	header	156
9.35.4.2	source	156
9.35.4.3	stanzas	156
9.36	unisys::OntoldRef Class Reference	156
9.36.1	Detailed Description	159

9.36.2 Constructor & Destructor Documentation	159
9.36.2.1 OntoldRef	159
9.36.2.2 OntoldRef	159
9.36.2.3 OntoldRef	159
9.36.3 Member Function Documentation	159
9.36.3.1 initMember	159
9.36.3.2 setId	159
9.37 unisys::Ontology Class Reference	159
9.37.1 Detailed Description	162
9.37.2 Constructor & Destructor Documentation	162
9.37.2.1 Ontology	162
9.37.2.2 Ontology	162
9.37.3 Member Function Documentation	162
9.37.3.1 initField	162
9.37.3.2 setDefinition	162
9.37.3.3 setNS	162
9.37.3.4 setTerm	162
9.38 unisys::OntoRelationship Class Reference	162
9.38.1 Detailed Description	165
9.38.2 Constructor & Destructor Documentation	165
9.38.2.1 OntoRelationship	165
9.38.2.2 OntoRelationship	165
9.38.2.3 OntoRelationship	165
9.38.2.4 OntoRelationship	165
9.38.3 Member Function Documentation	165
9.38.3.1 initField	165
9.38.3.2 setEvidence	165
9.38.3.3 setOntoRelationship	165
9.38.3.4 setRelationType	165
9.38.3.5 setRelationWith	166
9.39 unisys::ParsingError Class Reference	166
9.39.1 Constructor & Destructor Documentation	167
9.39.1.1 ParsingError	167
9.39.2 Member Function Documentation	167
9.39.2.1 what	168
9.40 unisys::PEIdRef Class Reference	168
9.40.1 Detailed Description	171
9.40.2 Constructor & Destructor Documentation	171
9.40.2.1 PEIdRef	171
9.40.2.2 PEIdRef	171

9.40.2.3	PEIdRef	171
9.40.3	Member Function Documentation	171
9.40.3.1	initMember	171
9.40.3.2	setId	171
9.41	unisys::PhysicalEntity Class Reference	171
9.41.1	Detailed Description	174
9.41.2	Constructor & Destructor Documentation	174
9.41.2.1	PhysicalEntity	174
9.41.3	Member Function Documentation	174
9.41.3.1	addCellularLocation	174
9.41.3.2	addSubRegion	174
9.41.3.3	initField	174
9.42	unisys::Protein Class Reference	174
9.42.1	Detailed Description	177
9.42.2	Constructor & Destructor Documentation	177
9.42.2.1	Protein	177
9.42.2.2	Protein	177
9.42.3	Member Function Documentation	177
9.42.3.1	addRNASource	177
9.42.3.2	addSpecificKineticParameter	177
9.42.3.3	initField	177
9.43	unisys::PSIMI Class Reference	177
9.43.1	Detailed Description	179
9.43.2	Member Typedef Documentation	180
9.43.2.1	mapOfStringString	180
9.43.2.2	mapOfXMLNode	180
9.43.3	Constructor & Destructor Documentation	180
9.43.3.1	PSIMI	180
9.43.3.2	PSIMI	180
9.43.4	Member Function Documentation	180
9.43.4.1	getNodeAttributeList	180
9.43.4.2	getNodeAvailabilityList	180
9.43.4.3	getNodeExperimentList	181
9.43.4.4	getNodeInteractionList	181
9.43.4.5	getNodeInteractorList	181
9.43.4.6	getNodeSource	181
9.43.4.7	parse	181
9.43.5	Member Data Documentation	181
9.43.5.1	attributeList	181
9.43.5.2	availabilityList	181

9.43.5.3	experimentList	181
9.43.5.4	interactionList	181
9.43.5.5	interactorList	181
9.43.5.6	source	182
9.44	unisys::Query Class Reference	182
9.44.1	Constructor & Destructor Documentation	184
9.44.1.1	Query	184
9.44.1.2	Query	184
9.44.2	Member Function Documentation	184
9.44.2.1	explain	184
9.44.2.2	perform	184
9.44.2.3	performOne	184
9.44.2.4	queryByld	184
9.44.2.5	setDbHandle	184
9.44.2.6	setQuery	184
9.44.2.7	setQuery	184
9.44.2.8	sort	184
9.44.2.9	toString	184
9.44.2.10	where	184
9.44.3	Member Data Documentation	184
9.44.3.1	databaseHandle	184
9.44.3.2	query	184
9.45	unisys::QueryError Class Reference	185
9.45.1	Constructor & Destructor Documentation	186
9.45.1.1	QueryError	186
9.45.2	Member Function Documentation	186
9.45.2.1	what	187
9.46	unisys::Relation Class Reference	187
9.46.1	Detailed Description	190
9.46.2	Constructor & Destructor Documentation	190
9.46.2.1	Relation	190
9.46.2.2	Relation	190
9.46.3	Member Function Documentation	190
9.46.3.1	initField	190
9.46.3.2	isValid	190
9.46.3.3	setCoefficient	190
9.46.3.4	setEvidence	190
9.46.3.5	setRelationWith	190
9.46.3.6	setRelationWith	190
9.46.3.7	setType	190

9.46.3.8	toRXNString	190
9.47	unisys::ClassBean::relationDataStructure Struct Reference	191
9.47.1	Member Function Documentation	192
9.47.1.1	getIdList	192
9.47.2	Member Data Documentation	192
9.47.2.1	count	192
9.47.2.2	list	192
9.47.2.3	type	192
9.48	unisys::REST Class Reference	192
9.48.1	Detailed Description	195
9.48.2	Constructor & Destructor Documentation	196
9.48.2.1	REST	196
9.48.2.2	~REST	196
9.48.3	Member Function Documentation	196
9.48.3.1	callback	196
9.48.3.2	cleanString	196
9.48.3.3	getResult	196
9.48.3.4	getResultInXML	196
9.48.3.5	globalClean	196
9.48.3.6	globallInit	196
9.48.3.7	isResponseError	196
9.48.3.8	perform	196
9.48.3.9	performHTTPGET	196
9.48.3.10	performPOST	196
9.48.3.11	print	196
9.48.4	Member Data Documentation	197
9.48.4.1	buffer	197
9.48.4.2	curlHandle	197
9.48.4.3	curlResult	197
9.48.4.4	result	197
9.48.4.5	xmlResult	197
9.49	unisys::RestBioMart Class Reference	197
9.49.1	Detailed Description	200
9.49.2	Constructor & Destructor Documentation	200
9.49.2.1	RestBioMart	200
9.49.2.2	RestBioMart	200
9.49.3	Member Function Documentation	200
9.49.3.1	bioMartVersion	200
9.49.3.2	getServiceURL	200
9.49.3.3	query	200

9.49.3.4	setServiceURL	200
9.49.4	Member Data Documentation	200
9.49.4.1	serviceURL	200
9.50	unisys::RestBioMartResult Class Reference	200
9.50.1	Detailed Description	202
9.50.2	Constructor & Destructor Documentation	202
9.50.2.1	RestBioMartResult	202
9.50.3	Member Function Documentation	202
9.50.3.1	bioMartVersion	202
9.50.4	Friends And Related Function Documentation	202
9.50.4.1	RestBioMart	202
9.50.5	Member Data Documentation	202
9.50.5.1	header	202
9.50.5.2	responseXML	202
9.51	unisys::RestBioPortal Class Reference	202
9.51.1	Detailed Description	205
9.51.2	Constructor & Destructor Documentation	205
9.51.2.1	RestBioPortal	205
9.51.3	Member Function Documentation	205
9.51.3.1	getSpecificOntology	205
9.51.3.2	listAllLastestOntologies	205
9.51.3.3	search	205
9.51.3.4	term	206
9.51.4	Member Data Documentation	206
9.51.4.1	baseURL	206
9.52	unisys::RestBioPortalResult Class Reference	206
9.52.1	Detailed Description	209
9.52.2	Constructor & Destructor Documentation	209
9.52.2.1	RestBioPortalResult	209
9.52.3	Member Function Documentation	209
9.52.3.1	getAccessDate	209
9.52.3.2	getAccessedResource	209
9.52.3.3	getError	209
9.52.3.4	isError	209
9.52.3.5	parseError	209
9.52.4	Member Data Documentation	210
9.52.4.1	accessDate	210
9.52.4.2	accessedResource	210
9.52.4.3	errorCode	210
9.52.4.4	errorLongMessage	210

9.53	unisys::RestBioPortalSearchResult Class Reference	210
9.53.1	Detailed Description	213
9.53.2	Constructor & Destructor Documentation	213
9.53.2.1	RestBioPortalSearchResult	213
9.53.2.2	RestBioPortalSearchResult	213
9.53.3	Member Function Documentation	213
9.53.3.1	getOntologyHitList	213
9.53.3.2	getSearchResultList	213
9.53.3.3	nSearchResults	213
9.53.4	Member Data Documentation	213
9.53.4.1	ontologyHitList	213
9.53.4.2	searchResultList	214
9.54	unisys::RestBioPortalTermResult Class Reference	214
9.54.1	Detailed Description	216
9.54.2	Constructor & Destructor Documentation	216
9.54.2.1	RestBioPortalTermResult	216
9.54.2.2	RestBioPortalTermResult	216
9.54.3	Member Function Documentation	216
9.54.3.1	getClassBean	216
9.54.4	Member Data Documentation	216
9.54.4.1	classBean	216
9.55	unisys::RestMiriam Class Reference	216
9.55.1	Detailed Description	219
9.55.2	Constructor & Destructor Documentation	219
9.55.2.1	RestMiriam	219
9.55.3	Member Function Documentation	219
9.55.3.1	resolve	219
9.55.4	Member Data Documentation	219
9.55.4.1	baseURL	219
9.56	unisys::RESTServiceError Class Reference	219
9.56.1	Constructor & Destructor Documentation	221
9.56.1.1	RESTServiceError	221
9.56.2	Member Function Documentation	221
9.56.2.1	what	222
9.57	unisys::RNA Class Reference	222
9.57.1	Detailed Description	225
9.57.2	Constructor & Destructor Documentation	225
9.57.2.1	RNA	225
9.57.2.2	RNA	225
9.57.3	Member Function Documentation	225

9.57.3.1	addPosition	225
9.57.3.2	addTranslationProduct	225
9.57.3.3	initField	225
9.57.3.4	setDNARegionSource	225
9.58	unisys::Score Class Reference	226
9.58.1	Detailed Description	228
9.58.2	Constructor & Destructor Documentation	228
9.58.2.1	Score	228
9.58.2.2	Score	228
9.58.3	Member Function Documentation	228
9.58.3.1	initField	228
9.58.3.2	setScoreType	228
9.58.3.3	setUnit	228
9.58.3.4	setValue	228
9.59	unisys::SmallMolecule Class Reference	229
9.59.1	Detailed Description	231
9.59.2	Constructor & Destructor Documentation	231
9.59.2.1	SmallMolecule	231
9.59.2.2	SmallMolecule	231
9.59.3	Member Function Documentation	231
9.59.3.1	addSpecificKineticParameter	231
9.59.3.2	initField	231
9.59.3.3	setFormula	231
9.59.3.4	setInChi	231
9.59.3.5	setInChiKey	232
9.59.3.6	setSMILES	232
9.60	unisys::Stanza Class Reference	232
9.60.1	Detailed Description	234
9.60.2	Constructor & Destructor Documentation	234
9.60.2.1	Stanza	234
9.60.2.2	Stanza	234
9.60.3	Member Function Documentation	234
9.60.3.1	getDefinition	234
9.60.3.2	getId	234
9.60.3.3	getName	234
9.60.3.4	getNS	234
9.60.3.5	getRelationship	234
9.60.3.6	isObsolete	234
9.60.3.7	toXML	234
9.60.4	Member Data Documentation	234

9.60.4.1	xml	234
9.61	unisys::SubRegion Class Reference	234
9.61.1	Detailed Description	237
9.61.2	Constructor & Destructor Documentation	237
9.61.2.1	SubRegion	237
9.61.2.2	SubRegion	237
9.61.3	Member Function Documentation	237
9.61.3.1	addPosition	237
9.61.3.2	initField	237
9.61.3.3	setAnnotation	237
9.61.3.4	setStrand	237
9.62	ToXMLStringTool Struct Reference	238
9.62.1	Detailed Description	238
9.62.2	Constructor & Destructor Documentation	239
9.62.2.1	ToXMLStringTool	239
9.62.2.2	~ToXMLStringTool	239
9.62.3	Member Function Documentation	239
9.62.3.1	freeBuffer	239
9.62.3.2	lengthXMLString	239
9.62.3.3	toXML	239
9.62.3.4	toXMLUnSafe	239
9.62.4	Member Data Documentation	239
9.62.4.1	buf	239
9.62.4.2	buflen	239
9.63	unisys::Tracking Class Reference	239
9.63.1	Detailed Description	242
9.63.2	Constructor & Destructor Documentation	242
9.63.2.1	Tracking	242
9.63.2.2	Tracking	242
9.63.3	Member Function Documentation	242
9.63.3.1	initField	242
9.63.3.2	toString	242
9.64	unisys::Transport Class Reference	242
9.64.1	Detailed Description	245
9.64.2	Constructor & Destructor Documentation	245
9.64.2.1	Transport	245
9.64.2.2	Transport	245
9.64.3	Member Function Documentation	245
9.64.3.1	addExport	245
9.64.3.2	addImport	245

9.64.3.3	initField	245
9.64.3.4	toRXNString	245
9.65	unisys::UniSysError Class Reference	246
9.65.1	Constructor & Destructor Documentation	247
9.65.1.1	~UniSysError	247
9.65.2	Member Data Documentation	247
9.65.2.1	errorMsg	247
9.66	unisys::UpdateError Class Reference	247
9.66.1	Constructor & Destructor Documentation	248
9.66.1.1	UpdateError	248
9.66.2	Member Function Documentation	248
9.66.2.1	what	249
9.67	unisys::Updater Class Reference	249
9.67.1	Constructor & Destructor Documentation	251
9.67.1.1	Updater	251
9.67.1.2	Updater	251
9.67.2	Member Function Documentation	251
9.67.2.1	checkIdPair	251
9.67.2.2	idConverter	252
9.67.2.3	idConverter	252
9.67.2.4	insert	252
9.67.2.5	insert	252
9.67.2.6	insert	252
9.67.2.7	insert	252
9.67.2.8	insert	252
9.67.2.9	remove	252
9.67.2.10	remove	252
9.67.2.11	remove	253
9.67.2.12	setDbHandle	253
9.67.2.13	update	253
9.67.2.14	update	253
9.67.2.15	update	253
9.67.2.16	update	254
9.67.2.17	updateIdPair	254
9.67.2.18	updateRelationAdd	254
9.67.2.19	updateRelationDel	254
9.67.3	Member Data Documentation	254
9.67.3.1	databaseHandle	254
9.68	XMLAttribute Struct Reference	255
9.68.1	Detailed Description	255

9.68.2 Member Data Documentation	255
9.68.2.1 IpszName	255
9.68.2.2 IpszValue	255
9.69 XMLClear Struct Reference	255
9.69.1 Detailed Description	256
9.69.2 Member Data Documentation	256
9.69.2.1 IpszCloseTag	256
9.69.2.2 IpszOpenTag	256
9.69.2.3 IpszValue	256
9.70 XMLNode Struct Reference	256
9.70.1 Detailed Description	262
9.70.2 Member Typedef Documentation	262
9.70.2.1 XMLCharEncoding	262
9.70.2.2 XMLNodeData	263
9.70.3 Member Enumeration Documentation	263
9.70.3.1 XMLCharEncoding	263
9.70.4 Constructor & Destructor Documentation	263
9.70.4.1 XMLNode	263
9.70.4.2 XMLNode	263
9.70.4.3 ~XMLNode	263
9.70.4.4 XMLNode	263
9.70.4.5 XMLNode	263
9.70.5 Member Function Documentation	263
9.70.5.1 addAttribute_priv	263
9.70.5.2 addChild_priv	263
9.70.5.3 addClear_priv	263
9.70.5.4 addText_priv	263
9.70.5.5 addToOrder	263
9.70.5.6 CreateXMLStringR	264
9.70.5.7 detachFromParent	264
9.70.5.8 emptyTheNode	264
9.70.5.9 exactMemory	264
9.70.5.10 findPosition	264
9.70.5.11 getVersion	264
9.70.5.12 indexClear	264
9.70.5.13 indexText	264
9.70.5.14 maybeAddTxT	264
9.70.5.15 operator=	264
9.70.5.16 parseClearTag	264
9.70.5.17 ParseXMLElement	264

9.70.5.18 removeOrderElement	264
9.70.6 Member Data Documentation	264
9.70.6.1 d	264
9.70.6.2 emptyXMLAttribute	264
9.70.6.3 emptyXMLClear	264
9.70.6.4 emptyXMLNode	264
9.71 XMLNodeContents Struct Reference	264
9.71.1 Detailed Description	266
9.71.2 Member Data Documentation	266
9.71.2.1 attrib	266
9.71.2.2 child	266
9.71.2.3 clear	266
9.71.2.4 etype	266
9.71.2.5 text	266
9.72 XMLNode::XMLNodeDataTag Struct Reference	267
9.72.1 Member Data Documentation	268
9.72.1.1 isDeclaration	268
9.72.1.2 lpszName	268
9.72.1.3 nAttribute	268
9.72.1.4 nChild	268
9.72.1.5 nClear	268
9.72.1.6 nText	268
9.72.1.7 pAttribute	268
9.72.1.8 pChild	268
9.72.1.9 pClear	268
9.72.1.10 pOrder	268
9.72.1.11 pParent	268
9.72.1.12 pText	268
9.72.1.13 ref_count	268
9.73 XMLParserBase64Tool Struct Reference	268
9.73.1 Detailed Description	270
9.73.2 Constructor & Destructor Documentation	270
9.73.2.1 XMLParserBase64Tool	270
9.73.2.2 ~XMLParserBase64Tool	270
9.73.3 Member Function Documentation	270
9.73.3.1 alloc	270
9.73.3.2 decode	270
9.73.3.3 decode	270
9.73.3.4 decodeSize	270
9.73.3.5 encode	270

9.73.3.6 encodeLength	271
9.73.3.7 freeBuffer	271
9.73.4 Member Data Documentation	271
9.73.4.1 buf	271
9.73.4.2 buflen	271
9.74 XMLResults Struct Reference	271
9.74.1 Detailed Description	272
9.74.2 Member Data Documentation	272
9.74.2.1 error	272
9.74.2.2 nColumn	272
9.74.2.3 nLine	272
9.75 unisys::Xref Class Reference	272
9.75.1 Detailed Description	275
9.75.2 Constructor & Destructor Documentation	275
9.75.2.1 Xref	275
9.75.2.2 Xref	275
9.75.2.3 Xref	275
9.75.3 Member Function Documentation	275
9.75.3.1 initField	275
9.75.3.2 operator<	276
9.75.3.3 setDetail	276
9.75.3.4 setId	276
9.75.3.5 setId	276
9.75.3.6 setType	276
10 File Documentation	277
10.1 biopax.h File Reference	277
10.1.1 Detailed Description	278
10.1.2 Macro Definition Documentation	278
10.1.2.1 _UNISYSDB_BIOPAX_H	278
10.2 chebiowl.h File Reference	278
10.3 database.h File Reference	279
10.4 DBClass.h File Reference	281
10.4.1 Detailed Description	283
10.5 exception.h File Reference	283
10.6 LitClass.h File Reference	284
10.6.1 Detailed Description	286
10.7 ObjClass.h File Reference	286
10.7.1 Detailed Description	287
10.8 objElement.h File Reference	288

10.8.1 Detailed Description	288
10.9 oboXML.h File Reference	288
10.9.1 Detailed Description	289
10.9.2 Macro Definition Documentation	289
10.9.2.1 <code>_UNISYSDB_OBOXML_H_</code>	289
10.10 parser.h File Reference	289
10.11 psimi.h File Reference	289
10.11.1 Detailed Description	290
10.12 query.h File Reference	290
10.12.1 Macro Definition Documentation	291
10.12.1.1 <code>_UNISYSDB_QUERY_H_</code>	291
10.13 rest.h File Reference	291
10.13.1 Detailed Description	292
10.14 restBioMart.h File Reference	293
10.14.1 Macro Definition Documentation	293
10.14.1.1 <code>BIOMART_VERSION</code>	293
10.15 restBioPortal.h File Reference	293
10.15.1 Detailed Description	294
10.16 restMiriam.h File Reference	294
10.16.1 Detailed Description	295
10.17 unisysdb.h File Reference	295
10.18 updater.h File Reference	295
10.18.1 Detailed Description	296
10.18.2 Macro Definition Documentation	297
10.18.2.1 <code>_UNISYSDB_UPDATER_H_</code>	297
10.19 xmlParser.h File Reference	297
10.19.1 Macro Definition Documentation	299
10.19.1.1 <code>_CXML</code>	299
10.19.1.2 <code>FALSE</code>	299
10.19.1.3 <code>TRUE</code>	299
10.19.1.4 <code>XMLCHAR</code>	299
10.19.1.5 <code>XMLCSTR</code>	299
10.19.1.6 <code>XMLELLENTRY</code>	299
10.19.1.7 <code>XMLELLENTRY</code>	299
10.19.1.8 <code>XMLSTR</code>	299
10.19.2 Typedef Documentation	299
10.19.2.1 <code>XMLAttribute</code>	299
10.19.2.2 <code>XMLClear</code>	299
10.19.2.3 <code>XMLElementPosition</code>	299
10.19.2.4 <code>XMLElementType</code>	300

10.19.2.5 XMLError	300
10.19.2.6 XMLNode	300
10.19.2.7 XMLNodeContents	300
10.19.2.8 XMLResults	300
10.19.3 Enumeration Type Documentation	300
10.19.3.1 XMLElementType	300
10.19.3.2 XMLError	300

Chapter 1

UniSysDB library

1.1 Introduction

This is a basic XML parser written in ANSI C++ for portability. It works by using recursion and a node tree for breaking down the elements of an XML document.

Version

V1.0.0

Author

Natapol Pornputtapong, Kwanjeera Wanichtanarak

Copyright (c) 2010-11, Systems and synthetic biology, Chalmers University of Technology
All rights reserved. See the file [AFPL-license.txt](#) about the licensing terms

1.2 First Tutorial

You can follow a simple [Tutorial](#) to know the basics...

1.3 General usage: How to include the XMLParser library inside your project.

The library is composed of two files: `xmlParser.cpp` and `xmlParser.h`. These are the ONLY 2 files that you need when using the library inside your own projects.

All the functions of the library are documented inside the comments of the file `xmlParser.h`. These comments can be transformed in full-fledged HTML documentation using the DOXYGEN software: simply type: "doxygen doxy.cfg"

By default, the XMLParser library uses `(char*)` for string representation. To use the `(wchar_t*)` version of the library, you need to define the `"_UNICODE"` preprocessor definition variable (this is usually done inside your project definition file) (This is done automatically for you when using Visual Studio).

Chapter 2

Module Index

2.1 Modules

Here is a list of all modules:

The XML parser	15
Parsing XML files/strings to an XMLNode structure and Rendering XMLNode's to files/string.	17
Navigate the XMLNode structure	21
Create or Update the XMLNode structure	25
Creating from scratch a XMLNode structure	26
Updating Nodes	28
Deleting Nodes or Attributes	30
??*_WOSD functions.	32
Position helper functions (use in conjunction with the update&add functions	35
String Allocation/Free functions	37
ato? like functions	38
Helper class to create XML files using "printf", "fprintf", "cout",... functions.	39
Helper class to include binary data inside XML strings using "Base64 encoding".	40

Chapter 3

Namespace Index

3.1 Namespace List

Here is a list of all namespaces with brief descriptions:

unisys	This namespace	41
--------	--------------------------	----

Chapter 4

Class Index

4.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

unisys::BIOPIAX2	63
unisys::BIOPIAX_obj	66
unisys::ChEBIOWL	76
unisys::ChEBIOWLClass	78
unisys::ClassBean	81
unisys::Database	97
unisys::DataObj	102
unisys::IdRef	121
unisys::IntIdRef	128
unisys::OntoldRef	156
unisys::PEIdRef	168
unisys::Literal	134
unisys::Annotation	45
unisys::BioSource	68
unisys::CellularLocation	72
unisys::Evidence	114
unisys::KineticParameter	130
unisys::MathML	137
unisys::Metadata	140
unisys::OntoRelationship	162
unisys::Relation	187
unisys::Score	226
unisys::SubRegion	234
unisys::Xref	272
unisys::Object	151
unisys::BioObject	59
unisys::Interaction	124
unisys::Control	91
unisys::Conversion	94
unisys::BiochemicalReaction	51
unisys::BiochemicalReactionWithTransport	55
unisys::Transport	242
unisys::GeneticInteraction	117
unisys::MolecularInteraction	147
unisys::PhysicalEntity	171
unisys::Complex	85
unisys::DNA	107

unisys::DNARegion	111
unisys::Protein	174
unisys::RNA	222
unisys::SmallMolecule	229
unisys::Ontology	159
unisys::Tracking	239
unisys::LiteralBSON	136
unisys::Miriam	143
unisys::OBOXML	154
unisys::PSIMI	177
unisys::Query	182
unisys::ClassBean::relationDataStructure	191
unisys::REST	192
unisys::RestBioMart	197
unisys::RestBioPortal	202
unisys::RestMiriam	216
unisys::RestBioMartResult	200
unisys::RestBioPortalResult	206
unisys::RestBioPortalSearchResult	210
unisys::RestBioPortalTermResult	214
unisys::Stanza	232
ToXMLStringTool	238
unisys::UniSysError	246
unisys::ConnectionError	88
unisys::DataError	100
unisys::ParsingError	166
unisys::QueryError	185
unisys::RESTServiceError	219
unisys::UpdateError	247
unisys::Updater	249
unisys::BatchInsert	49
XmlAttribute	255
XMLClear	255
XMLNode	256
XMLNodeContents	264
XMLNode::XMLNodeDataTag	267
XMLParserBase64Tool	268
XMLResults	271

Chapter 5

Class Index

5.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

unisys::Annotation	The C++ representative class for annotation data class	45
unisys::BatchInsert	49
unisys::BiochemicalReaction	This class is for miriam cross reference annotation	51
unisys::BiochemicalReactionWithTransport	This class is for miriam cross reference annotation	55
unisys::BioObject	This class is for miriam cross reference annotation	59
unisys::BIOPAX2	This class is used to parse OWL format	63
unisys::BIOPAX_obj	This class is used to parse OWL format	66
unisys::BioSource	The C++ representative class for Biological Source data class. This class is designed to manage the biological data of source organism	68
unisys::CellularLocation	The C++ representative class for cellular location data class	72
unisys::ChEBIOWL	Used to parse ChEBI OWL format	76
unisys::ChEBIOWLClass	To handle owl:class tag of ChEBI OWL format	78
unisys::ClassBean	This class is for taking care of the specific structure of response data call classBean	81
unisys::Complex	This class is for miriam cross reference annotation	85
unisys::ConnectionError	88
unisys::Control	This class is for miriam cross reference annotation	91
unisys::Conversion	This class is for miriam cross reference annotation	94
unisys::Database	This class is a database handle class	97
unisys::DataError	100
unisys::DataObj	102
unisys::DNA	This class is for miriam cross reference annotation	107

unisys::DNARegion	This class is for miriam cross reference annotation	111
unisys::Evidence	The C++ representative class for evidence data class	114
unisys::GeneticInteraction	This class is for miriam cross reference annotation	117
unisys::IdRef	Identifier reference class	121
unisys::Interaction	This class is for miriam cross reference annotation	124
unisys::IntIdRef	Identifier reference class specific to interaction collection namespace	128
unisys::KineticParameter	The C++ representative class for kinetic parameter data class	130
unisys::Literal	Literal class	134
unisys::LiteralBSON	Specific BSON object to manage general Literal data object	136
unisys::MathML	The C++ representative class for MathML data class	137
unisys::Metadata	The C++ representative class for metadata data class	140
unisys::Miriam	This class is for miriam cross reference annotation	143
unisys::MolecularInteraction	This class is for miriam cross reference annotation	147
unisys::Object	Root of all object classes This class does not have any interfaces	151
unisys::OBOXML	This class is used to parse OWL format	154
unisys::OntoldRef	Identifier reference class specific to obo collection namespace	156
unisys::Ontology	Ontology data class	159
unisys::OntoRelationship	The C++ representative class for relationship data class	162
unisys::ParsingError	166
unisys::PEIdRef	Identifier reference class specific to physicalentity collection namespace	168
unisys::PhysicalEntity	This class is for miriam cross reference annotation	171
unisys::Protein	This class is for miriam cross reference annotation	174
unisys::PSIMI	A PSIMI class is used for a data in PSI-MI XML2.5 file format	177
unisys::Query	182
unisys::QueryError	185
unisys::Relation	The C++ representative class for relationship data class	187
unisys::ClassBean::relationDataStructure	191
unisys::REST	Used for managing the request and response of REST service	192
unisys::RestBioMart	General class for taking care the response from BioPortal's REST service	197
unisys::RestBioMartResult	General class for taking care the response from BioPortal's REST service	200
unisys::RestBioPortal	The main class of BioPortal REST service library	202

unisys::RestBioPortalResult	General class for taking care the response from BioPortal's REST service	206
unisys::RestBioPortalSearchResult	General class for taking care the response from BioPortal's search command of REST service	210
unisys::RestBioPortalTermResult	This class is for parsing the response of term command	214
unisys::RestMiriam	General class for taking care the response from BioPortal's REST service	216
unisys::RESTServiceError	219
unisys::RNA	This class is for miriam cross reference annotation	222
unisys::Score	The C++ representative class for Score class in data structure	226
unisys::SmallMolecule	This class is for miriam cross reference annotation	229
unisys::Stanza	This class is used to parse OWL format	232
unisys::SubRegion	The C++ representative class for sub-region data class	234
ToXMLStringTool	Helper class to create XML files using "printf", "fprintf", "cout",... functions	238
unisys::Tracking	Data updating history class	239
unisys::Transport	This class is for miriam cross reference annotation	242
unisys::UniSysError	246
unisys::UpdateError	247
unisys::Updater	249
XmlAttribute	Structure for XML attribute	255
XMLClear	Structure for XML clear (unformatted) node (usually comments)	255
XMLNode	Main Class representing a XML node	256
XMLNodeContents	This structure is given by the function XMLNode::enumContents	264
XMLNode::XMLNodeDataTag	267
XMLParserBase64Tool	Helper class to include binary data inside XML strings using "Base64 encoding"	268
XMLResults	Structure used to obtain error details if the parse fails	271
unisys::Xref	The C++ representative class for Cross reference data class	272

Chapter 6

File Index

6.1 File List

Here is a list of all files with brief descriptions:

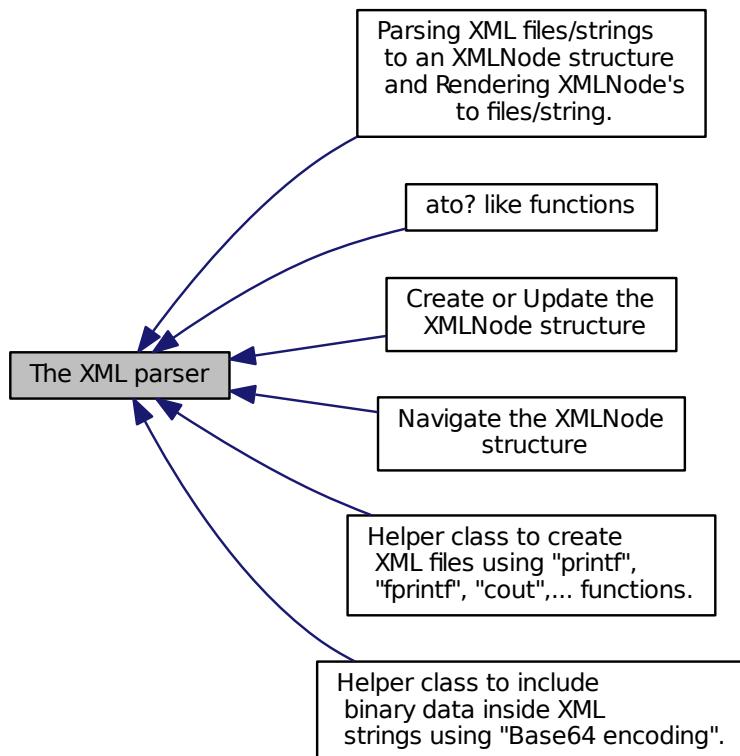
biopax.h	
It contain BIOPAX class using for parsing OWL (Web Ontology Language) format especially BIOPAX	277
chebiowl.h	278
database.h	279
DBClass.h	
It contain Miriam, IdRef, Tracking and LiteralBSON class using for formating data to database structure	281
exception.h	283
LitClass.h	
It contain all literal dirived class	284
ObjClass.h	
It contain all object derived class	286
objElement.h	288
oboXML.h	
It contain OBOXML and Stanza class using for parsing OBO-XML format	288
parser.h	289
psimi.h	
It contains PSIMI class for parsing PSI-MI XML2.5 file	289
query.h	290
rest.h	
It contains the library to handle the REST service request and response	291
restBioMart.h	293
restBioPortal.h	
BioPortal's REST service library	293
restMiriam.h	
Miriam's REST service library	294
unisysdb.h	295
updater.h	
It contain KGMLPlus and its subclass Graphic	295
xmlParser.h	297

Chapter 7

Module Documentation

7.1 The XML parser

Collaboration diagram for The XML parser:



Modules

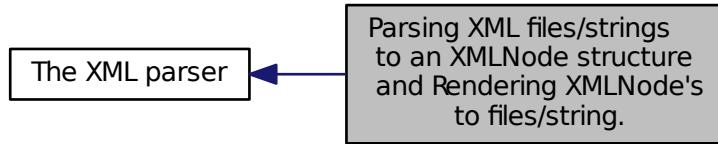
- Parsing XML files/strings to an XMLNode structure and Rendering XMLNode's to files/string.
- Navigate the XMLNode structure
- Create or Update the XMLNode structure

- `ato?` like functions
- Helper class to create XML files using "printf", "fprintf", "cout",... functions.
- Helper class to include binary data inside XML strings using "Base64 encoding".

7.1.1 Detailed Description

7.2 Parsing XML files/strings to an XMLNode structure and Rendering XMLNode's to files/string.

Collaboration diagram for Parsing XML files/strings to an XMLNode structure and Rendering XMLNode's to files/string.:



Functions

- static `XMLNode XMLNode::parseString (XMLCSTR lpXMLString, XMLCSTR tag=NULL, XMLResults *p-Results=NULL)`
Parse an XML string and return the root of a `XMLNode` tree representing the string.
- static `XMLNode XMLNode::parseFile (XMLCSTR filename, XMLCSTR tag=NULL, XMLResults *p-Results=NULL)`
Parse an XML file and return the root of a `XMLNode` tree representing the file.
- static `XMLNode XMLNode::openFileHelper (XMLCSTR filename, XMLCSTR tag=NULL)`
*Parse an XML file and return the root of a `XMLNode` tree representing the file. A very crude error checking is made.
An attempt to guess the Char Encoding used in the file is made.*
- static `XMLCSTR XMLNode::getError (XMLError error)`
this gives you a user-friendly explanation of the parsing error
- `XMLSTR XMLNode::createXMLString (int nFormat=1, int *pnSize=NULL) const`
Create an XML string starting from the current `XMLNode`.
- `XMLError XMLNode::writeToFile (XMLCSTR filename, const char *encoding=NULL, char nFormat=1) const`
Save the content of an `xmlNode` inside a file.
- static char `XMLNode::setGlobalOptions (XMLCharEncoding characterEncoding=XMLNode::char_encoding_UTF8, char guessWideCharChars=1, char dropWhiteSpace=1, char removeCommentsInMiddleOfText=1)`
Sets the global options for the conversions.
- static XMLCharEncoding `XMLNode::guessCharEncoding (void *buffer, int bufLen, char useXMLEncoding-Attribute=1)`
Guess the character encoding of the string (ascii, utf8 or shift-JIS)

7.2.1 Detailed Description

7.2.2 Function Documentation

7.2.2.1 `XMLSTR XMLNode::createXMLString (int nFormat = 1, int * pnSize = NULL) const`

Create an XML string starting from the current `XMLNode`.

The returned string should be free'd using the "freeXMLString" function.

If nFormat==0, no formatting is required otherwise this returns an user friendly XML string from a given XML node with appropriate white spaces and carriage returns. if pnSize is given it returns the size in character of the string.

7.2.2.2 static XMLCSTR XMLNode::getError (XMLError error) [static]

this gives you a user-friendly explanation of the parsing error

7.2.2.3 static XMLCharEncoding XMLNode::guessCharEncoding (void * buffer, int bufLen, char useXMLEncodingAttribute = 1) [static]

Guess the character encoding of the string (ascii, utf8 or shift-JIS)

The "guessCharEncoding" function try to guess the character encoding. You most-probably will never have to use this function. It then returns the appropriate value of the global parameter "characterEncoding" described in the [XMLNode::setGlobalOptions](#). The guess is based on the content of a buffer of length "bufLen" bytes that contains the first bytes (minimum 25 bytes; 200 bytes is a good value) of the file to be parsed. The [XMLNode::openFileHelper](#) function is using this function to automatically compute the value of the "characterEncoding" global parameter. There are several heuristics used to do the guess. One of the heuristic is based on the "encoding" attribute. The original XML specifications forbids to use this attribute to do the guess but you can still use it if you set "useXMLEncodingAttribute" to 1 (this is the default behavior and the behavior of most parsers). If an inconsistency in the encoding is detected, then the return value is "0".

7.2.2.4 static XMLNode XMLNode::openFileHelper (XMLCSTR filename, XMLCSTR tag = NULL) [static]

Parse an XML file and return the root of a [XMLNode](#) tree representing the file. A very crude error checking is made. An attempt to guess the Char Encoding used in the file is made.

The "openFileHelper" function reports to the screen all the warnings and errors that occurred during parsing of the XML file. This function also tries to guess char Encoding (UTF-8, ASCII or SHIT-JIS) based on the first 200 bytes of the file. Since each application has its own way to report and deal with errors, you should rather use the "parseFile" function to parse XML files and program yourself thereafter an "error reporting" tailored for your needs (instead of using the very crude "error reporting" mechanism included inside the "openFileHelper" function).

If the XML document is corrupted, the "openFileHelper" method will:

- display an error message on the console (or inside a messageBox for windows).
- stop execution (exit).

I strongly suggest that you write your own "openFileHelper" method tailored to your needs. If you still want to parse the file, you can use the APPROXIMATE_PARSING option as explained inside the note at the beginning of the "xmlParser.cpp" file.

Parameters

<i>filename</i>	the path of the XML file to parse.
<i>tag</i>	the name of the first tag inside the XML file. If the tag parameter is omitted, this function returns a node that represents the head of the xml document including the declaration term (<? ... ?>).

7.2.2.5 static XMLNode XMLNode::parseFile (XMLCSTR filename, XMLCSTR tag = NULL, XMLResults * pResults = NULL) [static]

Parse an XML file and return the root of a [XMLNode](#) tree representing the file.

The "parseFile" function parse an XML file and return the root of a [XMLNode](#) tree. The "opposite" of this function is the function "writeToFile" that re-creates an XML file from an [XMLNode](#) tree. If the XML document is corrupted, the

"parseFile" method will initialize the "pResults" variable with some information that can be used to trace the error. If you still want to parse the file, you can use the APPROXIMATE_PARSING option as explained inside the note at the beginning of the "xmlParser.cpp" file.

Parameters

<i>filename</i>	the path to the XML file to parse
<i>tag</i>	the name of the first tag inside the XML file. If the tag parameter is omitted, this function returns a node that represents the head of the xml document including the declaration term (<? ... ?>).
<i>pResults</i>	a pointer to a XMLResults variable that will contain some information that can be used to trace the XML parsing error. You can have a user-friendly explanation of the parsing error with the "getError" function.

7.2.2.6 static XMLNode XMLNode::parseString (XMLCSTR *lpXMLString*, XMLCSTR *tag* = NULL, XMLResults * *pResults* = NULL) [static]

Parse an XML string and return the root of a [XMLNode](#) tree representing the string.

The "parseString" function parse an XML string and return the root of a [XMLNode](#) tree. The "opposite" of this function is the function "createXMLString" that re-creates an XML string from an [XMLNode](#) tree. If the XML document is corrupted, the "parseString" method will initialize the "pResults" variable with some information that can be used to trace the error. If you still want to parse the file, you can use the APPROXIMATE_PARSING option as explained inside the note at the beginning of the "xmlParser.cpp" file.

Parameters

<i>lpXMLString</i>	the XML string to parse
<i>tag</i>	the name of the first tag inside the XML file. If the tag parameter is omitted, this function returns a node that represents the head of the xml document including the declaration term (<? ... ?>).
<i>pResults</i>	a pointer to a XMLResults variable that will contain some information that can be used to trace the XML parsing error. You can have a user-friendly explanation of the parsing error with the "getError" function.

7.2.2.7 static char XMLNode::setGlobalOptions (XMLCharEncoding *characterEncoding* = XMLNode::char_encoding_UTF8, char *guessWideCharChars* = 1, char *dropWhiteSpace* = 1, char *removeCommentsInMiddleOfText* = 1) [static]

Sets the global options for the conversions.

The "setGlobalOptions" function allows you to change four global parameters that affect string & file parsing. First of all, you most-probably will never have to change these 3 global parameters.

Parameters

<i>guessWideChar-Chars</i>	If "guessWideCharChars"=1 and if this library is compiled in WideChar mode, then the XM- LNode::parseFile and XMLNode::openFileHelper functions will test if the file contains ASCII characters. If this is the case, then the file will be loaded and converted in memory to Wide-Char before being parsed. If 0, no conversion will be performed.
<i>guessWideChar-Chars</i>	If "guessWideCharChars"=1 and if this library is compiled in ASCII=UTF8/char* mode, then the XMLNode::parseFile and XMLNode::openFileHelper functions will test if the file contains WideChar characters. If this is the case, then the file will be loaded and converted in memory to ASCII=UTF8/char* before being parsed. If 0, no conversion will be performed.
<i>character- Encoding</i>	This parameter is only meaningful when compiling in char* mode (multibyte character mode). In wchar_t* (wide char mode), this parameter is ignored. This parameter should be one of the three currently recognized encodings: XMLNode::encoding_UTF8, XMLNode::encoding_ -ascii, XMLNode::encoding_ShiftJIS.

<code>dropWhiteSpace</code>	In most situations, text fields containing only white spaces (and carriage returns) are useless. Even more, these "empty" text fields are annoying because they increase the complexity of the user's code for parsing. So, 99% of the time, it's better to drop the "empty" text fields. However The XML specification indicates that no white spaces should be lost when parsing the file. So to be perfectly XML-compliant, you should set <code>dropWhiteSpace=0</code> . A note of caution: if you set " <code>dropWhiteSpace=0</code> ", the parser will be slower and your code will be more complex.
<code>removeCommentsInMiddleOfText</code>	<p>To explain this parameter, let's consider this code:</p> <pre>XMLNode x=XMLNode::parseString("<a>foo<!-- hello -->bar<!DOCTYPE world >chu", "a");</pre> <p>If <code>removeCommentsInMiddleOfText=0</code>, then we will have:</p> <pre>x.getText(0) -> "foo" x.getText(1) -> "bar" x.getText(2) -> "chu" x.getClear(0) --> "<!-- hello -->" x.getClear(1) --> "<!DOCTYPE world >"</pre> <p>If <code>removeCommentsInMiddleOfText=1</code>, then we will have:</p> <pre>x.getText(0) -> "foobar" x.getText(1) -> "chu" x.getClear(0) --> "<!DOCTYPE world >"</pre>

Returns

"0" when there are no errors. If you try to set an unrecognized encoding then the return value will be "1" to signal an error.

Note

Sometime, it's useful to set "`guessWideCharChars=0`" to disable any conversion because the test to detect the file-type (ASCII/UTF8/char* or WideChar) may fail (rarely).

7.2.2.8 XMLError XMLNode::writeToFile (**XMLCSTR filename, const char * encoding = NULL, char nFormat = 1**) const

Save the content of an `xmlNode` inside a file.

If `nFormat==0`, no formatting is required otherwise this returns an user friendly XML string from a given element with appropriate white spaces and carriage returns. If the global parameter "characterEncoding==encoding_UTF8", then the "encoding" parameter is ignored and always set to "utf-8". If the global parameter "characterEncoding==encoding_ShiftJIS", then the "encoding" parameter is ignored and always set to "SHIFT-JIS". If "_XMLWIDECHAR=1", then the "encoding" parameter is ignored and always set to "utf-16". If no "encoding" parameter is given the "ISO-8859-1" encoding is used.

7.3 Navigate the XMLNode structure

Collaboration diagram for Navigate the XMLNode structure:



Functions

- `XMLCSTR XMLNode::getName () const`
name of the node
- `XMLCSTR XMLNode::getText (int i=0) const`
return ith text field
- `int XMLNode::nText () const`
nbr of text field
- `XMLNode XMLNode::getParentNode () const`
return the parent node
- `XMLNode XMLNode::getChildNode (int i=0) const`
return ith child node
- `XMLNode XMLNode::getChildNode (XMLCSTR name, int i) const`
return ith child node with specific name (return an empty node if failing). If i==1, this returns the last XMLNode with the given name.
- `XMLNode XMLNode::getChildNode (XMLCSTR name, int *i=NULL) const`
return next child node with specific name (return an empty node if failing)
- `XMLNode XMLNode::getChildNodeWithAttribute (XMLCSTR tagName, XMLCSTR attributeName, XMLCSTR attributeValue=NULL, int *i=NULL) const`
return child node with specific name/attribute (return an empty node if failing)
- `XMLNode XMLNode::getChildNodeByPath (XMLCSTR path, char createNodeIfMissing=0, XMLCHAR sep='/')`
return the first child node with specific path
- `XMLNode XMLNode::getChildNodeByPathNonConst (XMLSTR path, char createNodeIfMissing=0, XMLCHAR sep='/')`
return the first child node with specific path.
- `int XMLNode::nChildNode (XMLCSTR name) const`
return the number of child node with specific name
- `int XMLNode::nChildNode () const`
nbr of child node
- `XMLAttribute XMLNode::getAttribute (int i=0) const`
return ith attribute
- `XMLCSTR XMLNode::getAttributeName (int i=0) const`
return ith attribute name
- `XMLCSTR XMLNode::getAttributeValue (int i=0) const`
return ith attribute value
- `char XMLNode::isAttributeSet (XMLCSTR name) const`

- **XMLCSTR XMLNode::getAttribute (XMLCSTR name, int i) const**
return ith attribute content with specific name (return a NULL if failing)
- **XMLCSTR XMLNode::getAttribute (XMLCSTR name, int *i=NULL) const**
return next attribute content with specific name (return a NULL if failing)
- **int XMLNode::nAttribute () const**
nbr of attribute
- **XMLClear XMLNode::getClear (int i=0) const**
return ith clear field (comments)
- **int XMLNode::nClear () const**
nbr of clear field
- **XMLNodeContents XMLNode::enumContents (XMLElementPosition i) const**
enumerate all the different contents (attribute,child,text, clear) of the current XMLNode. The order is reflecting the order of the original file/string. NOTE: 0 <= i < nElement();
- **int XMLNode::nElement () const**
nbr of different contents for current node
- **char XMLNode::isEmpty () const**
is this node Empty?
- **char XMLNode::isDeclaration () const**
is this node a declaration <? ?>
- **XMLNode XMLNode::deepCopy () const**
deep copy (duplicate/clone) a XMLNode
- **static XMLNode XMLNode::emptyNode ()**
return XMLNode::emptyXMLNode;

7.3.1 Detailed Description

7.3.2 Function Documentation

7.3.2.1 XMLNode XMLNode::deepCopy () const

deep copy (duplicate/clone) a XMLNode

7.3.2.2 static XMLNode XMLNode::emptyNode () [static]

return XMLNode::emptyXMLNode;

7.3.2.3 XMLNodeContents XMLNode::enumContents (XMLElementPosition i) const

enumerate all the different contents (attribute,child,text, clear) of the current XMLNode. The order is reflecting the order of the original file/string. NOTE: 0 <= i < nElement();

7.3.2.4 XMLAttribute XMLNode::getAttribute (int i = 0) const

return ith attribute

7.3.2.5 XMLCSTR XMLNode::getAttribute (XMLCSTR name, int i) const

return ith attribute content with specific name (return a NULL if failing)

7.3.2.6 XMLECSTR XMLNode::getAttribute (XMLECSTR *name*, int * *i* = NULL) const

return next attribute content with specific name (return a NULL if failing)

7.3.2.7 XMLECSTR XMLNode::getAttributeName (int *i* = 0) const

return ith attribute name

7.3.2.8 XMLECSTR XMLNode::getAttributeValue (int *i* = 0) const

return ith attribute value

7.3.2.9 XMLNode XMLNode::getChildNode (int *i* = 0) const

return ith child node

7.3.2.10 XMLNode XMLNode::getChildNode (XMLECSTR *name*, int *i*) const

return ith child node with specific name (return an empty node if failing). If *i* == -1, this returns the last [XMLNode](#) with the given name.

7.3.2.11 XMLNode XMLNode::getChildNode (XMLECSTR *name*, int * *i* = NULL) const

return next child node with specific name (return an empty node if failing)

7.3.2.12 XMLNode XMLNode::getChildNodeByPath (XMLECSTR *path*, char *createNodeIfMissing* = 0, XMLCHAR *sep* = '/')

return the first child node with specific path

7.3.2.13 XMLNode XMLNode::getChildNodeByPathNonConst (XMLSTR *path*, char *createNodeIfMissing* = 0, XMLCHAR *sep* = '/')

return the first child node with specific path.

7.3.2.14 XMLNode XMLNode::getChildNodeWithAttribute (XMLECSTR *tagName*, XMLECSTR *attributeName*, XMLECSTR *attributeValue* = NULL, int * *i* = NULL) const

return child node with specific name/attribute (return an empty node if failing)

7.3.2.15 XMLClear XMLNode::getClear (int *i* = 0) const

return ith clear field (comments)

7.3.2.16 XMLECSTR XMLNode::getName () const

name of the node

7.3.2.17 **XMLNode XMLNode::getParentNode () const**

return the parent node

7.3.2.18 **XMLCSTR XMLNode::getText (int *i* = 0) const**

return *i*th text field

7.3.2.19 **char XMLNode::isAttributeSet (XMLCSTR *name*) const**

test if an attribute with a specific name is given

7.3.2.20 **char XMLNode::isDeclaration () const**

is this node a declaration <? ?>

7.3.2.21 **char XMLNode::isEmpty () const**

is this node Empty?

7.3.2.22 **int XMLNode::nAttribute () const**

nbr of attribute

7.3.2.23 **int XMLNode::nChildNode (XMLCSTR *name*) const**

return the number of child node with specific name

7.3.2.24 **int XMLNode::nChildNode () const**

nbr of child node

7.3.2.25 **int XMLNode::nClear () const**

nbr of clear field

7.3.2.26 **int XMLNode::nElement () const**

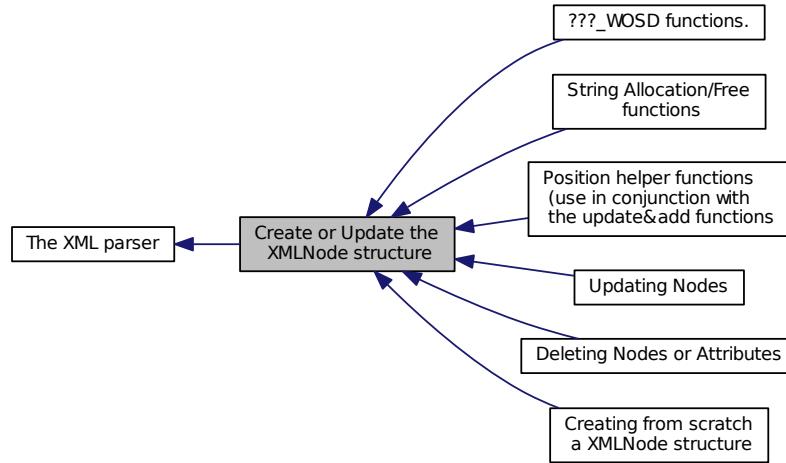
nbr of different contents for current node

7.3.2.27 **int XMLNode::nText () const**

nbr of text field

7.4 Create or Update the XMLNode structure

Collaboration diagram for Create or Update the XMLNode structure:



Modules

- [Creating from scratch a XMLNode structure](#)
- [Updating Nodes](#)
- [Deleting Nodes or Attributes](#)
- [???_WOSD functions.](#)
- [Position helper functions \(use in conjunction with the update&add functions\)](#)
- [String Allocation/Free functions](#)

7.4.1 Detailed Description

The functions in this group allows you to create from scratch (or update) a [XMLNode](#) structure. Start by creating your top node with the "createXMLTopNode" function and then add new nodes with the "addChild" function. The parameter 'pos' gives the position where the childNode, the text or the XMLClearTag will be inserted. The default value (pos=-1) inserts at the end. The value (pos=0) insert at the beginning (Insertion at the beginning is slower than at the end).

REMARK: $0 \leq pos < nChild() + nText() + nClear()$

7.5 Creating from scratch a XMLNode structure

Collaboration diagram for Creating from scratch a XMLNode structure:



Functions

- static `XMLNode XMLNode::createXMLTopNode (XMLCSTR lpszName, char isDeclaration=FALSE)`
Create the top node of an XMLNode structure.
- `XMLNode XMLNode::addChild (XMLCSTR lpszName, char isDeclaration=FALSE, XMLElementPosition pos=-1)`
Add a new child node.
- `XMLNode XMLNode::addChild (XMLNode nodeToAdd, XMLElementPosition pos=-1)`
If the "nodeToAdd" has some parents, it will be detached from its parents before being attached to the current XMLNode.
- `XMLAttribute * XMLNode::addAttribute (XMLCSTR lpszName, XMLCSTR lpszValuev)`
Add a new attribute.
- `XMLCSTR XMLNode::addText (XMLCSTR lpszValue, XMLElementPosition pos=-1)`
Add a new text content.
- `XMLClear * XMLNode::addClear (XMLCSTR lpszValue, XMLCSTR lpszOpen=NULL, XMLCSTR lpszClose=NULL, XMLElementPosition pos=-1)`

7.5.1 Detailed Description

7.5.2 Function Documentation

7.5.2.1 `XMLAttribute* XMLNode::addAttribute (XMLCSTR lpszName, XMLCSTR lpszValuev)`

Add a new attribute.

7.5.2.2 `XMLNode XMLNode::addChild (XMLCSTR lpszName, char isDeclaration = FALSE, XMLElementPosition pos = -1)`

Add a new child node.

7.5.2.3 `XMLNode XMLNode::addChild (XMLNode nodeToAdd, XMLElementPosition pos = -1)`

If the "nodeToAdd" has some parents, it will be detached from its parents before being attached to the current XMLNode.

7.5.2.4 `XMLClear* XMLNode::addClear (XMLECSTR lpszValue, XMLECSTR lpszOpen = NULL, XMLECSTR lpszClose = NULL, XMLElementPosition pos = -1)`

Add a new clear tag

Parameters

<code>lpszOpen</code>	default value "<![CDATA["
<code>lpszClose</code>	default value "]]>"
<code>lpszValue</code>	
<code>pos</code>	

7.5.2.5 `XMLECSTR XMLNode::addText (XMLECSTR lpszValue, XMLElementPosition pos = -1)`

Add a new text content.

7.5.2.6 `static XMLNode XMLNode::createXMLTopNode (XMLECSTR lpszName, char isDeclaration = FALSE) [static]`

Create the top node of an [XMLNode](#) structure.

7.6 Updating Nodes

Collaboration diagram for Updating Nodes:



Functions

- `XMLCSTR XMLNode::updateName (XMLCSTR lpszName)`
change node's name
- `XMLAttribute * XMLNode::updateAttribute (XMLAttribute *newAttribute, XMLAttribute *oldAttribute)`
if the attribute to update is missing, a new one will be added
- `XMLAttribute * XMLNode::updateAttribute (XMLCSTR lpsznewValue, XMLCSTR lpsz newName=NULL, int i=0)`
if the attribute to update is missing, a new one will be added
- `XMLAttribute * XMLNode::updateAttribute (XMLCSTR lpsznewValue, XMLCSTR lpsz newName, XMLCSTR lpszOldName)`
set lpszNewName=NULL if you don't want to change the name of the attribute if the attribute to update is missing, a new one will be added
- `XMLCSTR XMLNode::updateText (XMLCSTR lpsznewValue, int i=0)`
if the text to update is missing, a new one will be added
- `XMLCSTR XMLNode::updateText (XMLCSTR lpsznewValue, XMLCSTR lpszOldValue)`
if the text to update is missing, a new one will be added
- `XMLClear * XMLNode::updateClear (XMLCSTR lpszNewContent, int i=0)`
if the clearTag to update is missing, a new one will be added
- `XMLClear * XMLNode::updateClear (XMLClear *newP, XMLClear *oldP)`
if the clearTag to update is missing, a new one will be added
- `XMLClear * XMLNode::updateClear (XMLCSTR lpsznewValue, XMLCSTR lpszOldValue)`
if the clearTag to update is missing, a new one will be added

7.6.1 Detailed Description

Some update functions:

7.6.2 Function Documentation

7.6.2.1 `XMLAttribute* XMLNode::updateAttribute (XMLAttribute * newAttribute, XMLAttribute * oldAttribute)`

if the attribute to update is missing, a new one will be added

7.6.2.2 `XMLAttribute* XMLNode::updateAttribute (XMLCSTR lpsznewValue, XMLCSTR lpsz newName = NULL, int i = 0)`

if the attribute to update is missing, a new one will be added

7.6.2.3 **XMLAttribute*** XMLNode::updateAttribute (**XMLCSTR** *lpszNewValue*, **XMLCSTR** *lpszNewName*, **XMLCSTR** *lpszOldName*)

set *lpszNewName*=NULL if you don't want to change the name of the attribute if the attribute to update is missing, a new one will be added

7.6.2.4 **XMLClear*** XMLNode::updateClear (**XMLCSTR** *lpszNewContent*, int *i* = 0)

if the clearTag to update is missing, a new one will be added

7.6.2.5 **XMLClear*** XMLNode::updateClear (**XMLClear** * *newP*, **XMLClear** * *oldP*)

if the clearTag to update is missing, a new one will be added

7.6.2.6 **XMLClear*** XMLNode::updateClear (**XMLCSTR** *lpszNewValue*, **XMLCSTR** *lpszOldValue*)

if the clearTag to update is missing, a new one will be added

7.6.2.7 **XMLCSTR** XMLNode::updateName (**XMLCSTR** *lpszName*)

change node's name

7.6.2.8 **XMLCSTR** XMLNode::updateText (**XMLCSTR** *lpsznewValue*, int *i* = 0)

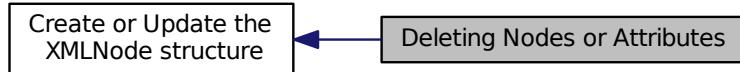
if the text to update is missing, a new one will be added

7.6.2.9 **XMLCSTR** XMLNode::updateText (**XMLCSTR** *lpsznewValue*, **XMLCSTR** *lpszOldValue*)

if the text to update is missing, a new one will be added

7.7 Deleting Nodes or Attributes

Collaboration diagram for Deleting Nodes or Attributes:



Functions

- void [XMLNode::deleteNodeContent \(\)](#)
The "deleteNodeContent" function forces the deletion of the content of this [XMLNode](#) and the subtree.
- void [XMLNode::deleteAttribute \(int i=0\)](#)
Delete the ith attribute of the current [XMLNode](#).
- void [XMLNode::deleteAttribute \(XMLCSTR lpszName\)](#)
Delete the attribute with the given name (the "strcmp" function is used to find the right attribute)
- void [XMLNode::deleteAttribute \(XMLAttribute *anAttribute\)](#)
Delete the attribute with the name "anAttribute->lpszName" (the "strcmp" function is used to find the right attribute)
- void [XMLNode::deleteText \(int i=0\)](#)
Delete the ith text content of the current [XMLNode](#).
- void [XMLNode::deleteText \(XMLCSTR lpszValue\)](#)
Delete the text content "lpszValue" inside the current [XMLNode](#) (direct "pointer-to-pointer" comparison is used to find the right text)
- void [XMLNode::deleteClear \(int i=0\)](#)
Delete the ith clear tag inside the current [XMLNode](#).
- void [XMLNode::deleteClear \(XMLCSTR lpszValue\)](#)
Delete the clear tag "lpszValue" inside the current [XMLNode](#) (direct "pointer-to-pointer" comparison is used to find the clear tag)
- void [XMLNode::deleteClear \(XMLClear *p\)](#)
Delete the clear tag "p" inside the current [XMLNode](#) (direct "pointer-to-pointer" comparison on the lpszName of the clear tag is used to find the clear tag)

7.7.1 Detailed Description

Some deletion functions:

7.7.2 Function Documentation

7.7.2.1 void XMLNode::deleteAttribute (int i = 0)

Delete the ith attribute of the current [XMLNode](#).

7.7.2.2 void XMLNode::deleteAttribute (XMLCSTR lpszName)

Delete the attribute with the given name (the "strcmp" function is used to find the right attribute)

7.7.2.3 void XMLNode::deleteAttribute (*XMLAttribute * anAttribute*)

Delete the attribute with the name "anAttribute->lpszName" (the "strcmp" function is used to find the right attribute)

7.7.2.4 void XMLNode::deleteClear (*int i = 0*)

Delete the *i*th clear tag inside the current [XMLNode](#).

7.7.2.5 void XMLNode::deleteClear (*XMLCSTR lpszValue*)

Delete the clear tag "lpszValue" inside the current [XMLNode](#) (direct "pointer-to-pointer" comparison is used to find the clear tag)

7.7.2.6 void XMLNode::deleteClear (*XMLClear * p*)

Delete the clear tag "p" inside the current [XMLNode](#) (direct "pointer-to-pointer" comparison on the lpszName of the clear tag is used to find the clear tag)

7.7.2.7 void XMLNode::deleteNodeContent ()

The "deleteNodeContent" function forces the deletion of the content of this [XMLNode](#) and the subtree.

Note

The [XMLNode](#) instances that are referring to the part of the subtree that has been deleted CANNOT be used anymore!. Unexpected results will occur if you continue using them.

7.7.2.8 void XMLNode::deleteText (*int i = 0*)

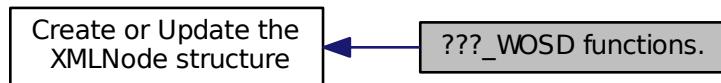
Delete the *i*th text content of the current [XMLNode](#).

7.7.2.9 void XMLNode::deleteText (*XMLCSTR lpszValue*)

Delete the text content "lpszValue" inside the current [XMLNode](#) (direct "pointer-to-pointer" comparison is used to find the right text)

7.8 ???_WOSD functions.

Collaboration diagram for ???_WOSD functions.:



Functions

- static `XMLNode XMLNode::createXMLTopNode_WOSD (XMLSTR lpszName, char isDeclaration=FALSE)`
Create the top node of an XMLNode structure.
- `XMLNode XMLNode::addChild_WOSD (XMLSTR lpszName, char isDeclaration=FALSE, XMLElementPosition pos=-1)`
Add a new child node.
- `XMLAttribute * XMLNode::addAttribute_WOSD (XMLSTR lpszName, XMLSTR lpszValue)`
Add a new attribute.
- `XMLCSTR XMLNode::addText_WOSD (XMLSTR lpszValue, XMLElementPosition pos=-1)`
Add a new text content.
- `XMLClear * XMLNode::addClear_WOSD (XMLSTR lpszValue, XMLCSTR lpszOpen=NULL, XMLCSTR lpszClose=NULL, XMLElementPosition pos=-1)`
Add a new clear Tag.
- `XMLCSTR XMLNode::updateName_WOSD (XMLSTR lpszName)`
change node's name
- `XMLAttribute * XMLNode::updateAttribute_WOSD (XMLAttribute *newAttribute, XMLAttribute *oldAttribute)`
if the attribute to update is missing, a new one will be added
- `XMLAttribute * XMLNode::updateAttribute_WOSD (XMLSTR lpsznewValue, XMLSTR lpszNewName=NULL, int i=0)`
if the attribute to update is missing, a new one will be added
- `XMLAttribute * XMLNode::updateAttribute_WOSD (XMLSTR lpsznewValue, XMLSTR lpszNewName, XMLCSTR lpszOldName)`
set lpszNewName=NULL if you don't want to change the name of the attribute if the attribute to update is missing, a new one will be added
- `XMLCSTR XMLNode::updateText_WOSD (XMLSTR lpsznewValue, int i=0)`
if the text to update is missing, a new one will be added
- `XMLCSTR XMLNode::updateText_WOSD (XMLSTR lpsznewValue, XMLCSTR lpszOldValue)`
if the text to update is missing, a new one will be added
- `XMLClear * XMLNode::updateClear_WOSD (XMLSTR lpszNewContent, int i=0)`
if the clearTag to update is missing, a new one will be added
- `XMLClear * XMLNode::updateClear_WOSD (XMLClear *newP, XMLClear *oldP)`
if the clearTag to update is missing, a new one will be added
- `XMLClear * XMLNode::updateClear_WOSD (XMLSTR lpsznewValue, XMLCSTR lpszOldValue)`
if the clearTag to update is missing, a new one will be added

7.8.1 Detailed Description

The strings given as parameters for the "add" and "update" methods that have a name with the postfix "_WOSD" (that means "WithOut String Duplication") (for example "addText_WOSD") will be free'd by the [XMLNode](#) class. For example, it means that this is incorrect:

```
xNode.addText_WOSD("foo");
xNode.updateAttribute_WOSD("#newcolor",NULL,"color");
```

In opposition, this is correct:

```
xNode.addText("foo");
xNode.addText_WOSD(stringDup("foo"));
xNode.updateAttribute("#newcolor",NULL,"color");
xNode.updateAttribute_WOSD(stringDup("#newcolor"),NULL,"color");
```

Typically, you will never do:

```
char *b=(char*)malloc(...);
xNode.addText(b);
free(b);
```

... but rather:

```
char *b=(char*)malloc(...);
xNode.addText_WOSD(b);
```

('free(b)' is performed by the [XMLNode](#) class)

7.8.2 Function Documentation

7.8.2.1 XMLAttribute* XMLNode::addAttribute_WOSD (XMLSTR lpszName, XMLSTR lpszValue)

Add a new attribute.

7.8.2.2 XMLNode XMLNode::addChild_WOSD (XMLSTR lpszName, char isDeclaration = FALSE, XMLElementPosition pos = -1)

Add a new child node.

7.8.2.3 XMLClear* XMLNode::addClear_WOSD (XMLSTR lpszValue, XMLCSTR lpszOpen = NULL, XMLCSTR lpszClose = NULL, XMLElementPosition pos = -1)

Add a new clear Tag.

7.8.2.4 XMLCSTR XMLNode::addText_WOSD (XMLSTR lpszValue, XMLElementPosition pos = -1)

Add a new text content.

7.8.2.5 static XMLNode XMLNode::createXMLTopNode_WOSD (XMLSTR lpszName, char isDeclaration = FALSE) [static]

Create the top node of an [XMLNode](#) structure.

7.8.2.6 XMLAttribute* XMLNode::updateAttribute_WOSD (*XMLAttribute * newAttribute, XMLAttribute * oldAttribute*)

if the attribute to update is missing, a new one will be added

7.8.2.7 XMLAttribute* XMLNode::updateAttribute_WOSD (*XMLSTR lpsznewValue, XMLSTR lpszNewName = NULL, int i = 0*)

if the attribute to update is missing, a new one will be added

7.8.2.8 XMLAttribute* XMLNode::updateAttribute_WOSD (*XMLSTR lpsznewValue, XMLSTR lpszNewName, XMLCSTR lpszOldName*)

set lpszNewName=NULL if you don't want to change the name of the attribute if the attribute to update is missing, a new one will be added

7.8.2.9 XMLClear* XMLNode::updateClear_WOSD (*XMLSTR lpszNewContent, int i = 0*)

if the clearTag to update is missing, a new one will be added

7.8.2.10 XMLClear* XMLNode::updateClear_WOSD (*XMLClear * newP, XMLClear * oldP*)

if the clearTag to update is missing, a new one will be added

7.8.2.11 XMLClear* XMLNode::updateClear_WOSD (*XMLSTR lpsznewValue, XMLCSTR lpszOldValue*)

if the clearTag to update is missing, a new one will be added

7.8.2.12 XMLCSTR XMLNode::updateName_WOSD (*XMLSTR lpszName*)

change node's name

7.8.2.13 XMLCSTR XMLNode::updateText_WOSD (*XMLSTR lpsznewValue, int i = 0*)

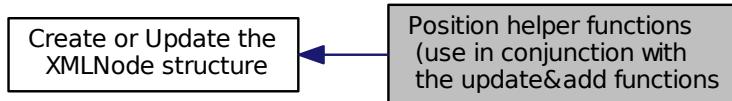
if the text to update is missing, a new one will be added

7.8.2.14 XMLCSTR XMLNode::updateText_WOSD (*XMLSTR lpsznewValue, XMLCSTR lpszOldValue*)

if the text to update is missing, a new one will be added

7.9 Position helper functions (use in conjunction with the update&add functions)

Collaboration diagram for Position helper functions (use in conjunction with the update&add functions):



Functions

- `XMLElementPosition XMLNode::positionOfText (int i=0) const`
 - `XMLElementPosition XMLNode::positionOfText (XMLCSTR lpszValue) const`
 - `XMLElementPosition XMLNode::positionOfClear (int i=0) const`
 - `XMLElementPosition XMLNode::positionOfClear (XMLCSTR lpszValue) const`
 - `XMLElementPosition XMLNode::positionOfClear (XMLClear *a) const`
 - `XMLElementPosition XMLNode::positionOfChildNode (int i=0) const`
 - `XMLElementPosition XMLNode::positionOfChildNode (XMLNode x) const`
 - `XMLElementPosition XMLNode::positionOfChildNode (XMLCSTR name, int i=0) const`
- return the position of the ith childNode with the specified name if (name==NULL) return the position of the ith child-
Node*

7.9.1 Detailed Description

These are some useful functions when you want to insert a childNode, a text or a XMLClearTag in the middle (at a specified position) of a `XMLNode` tree already constructed. The value returned by these methods is to be used as last parameter (parameter 'pos') of addChild, addText or addClear.

7.9.2 Function Documentation

7.9.2.1 `XMLElementPosition XMLNode::positionOfChildNode (int i = 0) const`

7.9.2.2 `XMLElementPosition XMLNode::positionOfChildNode (XMLNode x) const`

7.9.2.3 `XMLElementPosition XMLNode::positionOfChildNode (XMLCSTR name, int i = 0) const`

return the position of the ith childNode with the specified name if (name==NULL) return the position of the ith childNode

7.9.2.4 `XMLElementPosition XMLNode::positionOfClear (int i = 0) const`

7.9.2.5 `XMLElementPosition XMLNode::positionOfClear (XMLCSTR lpszValue) const`

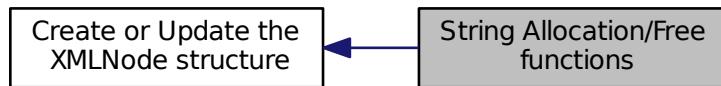
7.9.2.6 `XMLElementPosition XMLNode::positionOfClear (XMLClear * a) const`

7.9.2.7 `XMLElementPosition XMLNode::positionOfText (int i = 0) const`

7.9.2.8 XMLElementPosition XMLNode::positionOfText (*XMLCSTR lpszValue*) const

7.10 String Allocation/Free functions

Collaboration diagram for String Allocation/Free functions:



Functions

- **XMDLLENTRY XMLSTR stringDup (XMLCSTR source, int cbData=-1)**
Duplicate (copy in a new allocated buffer) the source string.
- **XMDLLENTRY void freeXMLString (XMLSTR t)**
to free the string allocated inside the "stringDup" function or the "createXMLString" function.

7.10.1 Detailed Description

7.10.2 Function Documentation

7.10.2.1 XMDLLENTRY void freeXMLString (XMLSTR t)

to free the string allocated inside the "stringDup" function or the "createXMLString" function.

7.10.2.2 XMDLLENTRY XMLSTR stringDup (XMLCSTR source, int cbData = -1)

Duplicate (copy in a new allocated buffer) the source string.

This is a very handy function when used with all the "XMLNode::*_WOSD" functions ([xmlWOSD](#)).

Parameters

<i>cbData</i>	If !=0 then cbData is the number of chars to duplicate. New strings allocated with this function should be free'd using the "freeXMLString" function.
<i>source</i>	

7.11 ato? like functions

Collaboration diagram for ato? like functions:



Functions

- **XMDLLENTRY** char **xmltob** (**XMLCSTR** *xmlString*, char *defaultValue*=0)
- **XMDLLENTRY** int **xmltoi** (**XMLCSTR** *xmlString*, int *defaultValue*=0)
- **XMDLLENTRY** long **xmltol** (**XMLCSTR** *xmlString*, long *defaultValue*=0)
- **XMDLLENTRY** double **xmltof** (**XMLCSTR** *xmlString*, double *defaultValue*=.0)
- **XMDLLENTRY** **XMLCSTR** **xmltoa** (**XMLCSTR** *xmlString*, **XMLCSTR** *defaultValue*=**_CXML("")**)
- **XMDLLENTRY** **XMLCHAR** **xmltoc** (**XMLCSTR** *xmlString*, const **XMLCHAR** *defaultValue*=**_CXML("\0")**)

7.11.1 Detailed Description

The "xmlto?" functions are equivalents to the atoi, atol, atof functions. The only difference is: If the variable "xmlString" is NULL, than the return value is "defaultValue". These 6 functions are only here as "convenience" functions for the user (they are not used inside the XMLparser). If you don't need them, you can delete them without any trouble.

7.11.2 Function Documentation

7.11.2.1 **XMDLLENTRY** **XMLCSTR** **xmltoa** (**XMLCSTR** *xmlString*, **XMLCSTR** *defaultValue*=**_CXML("")**)

7.11.2.2 **XMDLLENTRY** char **xmltob** (**XMLCSTR** *xmlString*, char *defaultValue*=0)

7.11.2.3 **XMDLLENTRY** **XMLCHAR** **xmltoc** (**XMLCSTR** *xmlString*, const **XMLCHAR** *defaultValue*=**_CXML('\0')**)

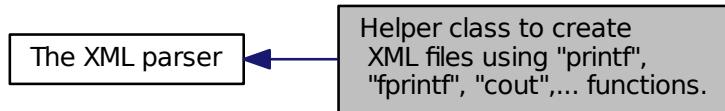
7.11.2.4 **XMDLLENTRY** double **xmltof** (**XMLCSTR** *xmlString*, double *defaultValue*=.0)

7.11.2.5 **XMDLLENTRY** int **xmltoi** (**XMLCSTR** *xmlString*, int *defaultValue*=0)

7.11.2.6 **XMDLLENTRY** long **xmltol** (**XMLCSTR** *xmlString*, long *defaultValue*=0)

7.12 Helper class to create XML files using "printf", "fprintf", "cout",... functions.

Collaboration diagram for Helper class to create XML files using "printf", "fprintf", "cout",... functions.:



Classes

- struct [ToXMLStringTool](#)
Helper class to create XML files using "printf", "fprintf", "cout",... functions.

TypeDefs

- typedef struct [XMLDLLENTRY](#)
[ToXMLStringTool](#) [ToXMLStringTool](#)
Helper class to create XML files using "printf", "fprintf", "cout",... functions.

7.12.1 Detailed Description

7.12.2 Typedef Documentation

7.12.2.1 typedef struct XMLDLLENTRY ToXMLStringTool ToXMLStringTool

Helper class to create XML files using "printf", "fprintf", "cout",... functions.

The [ToXMLStringTool](#) class helps you creating XML files using "printf", "fprintf", "cout",... functions. The "ToXMLStringTool" class is processing strings so that all the characters &,"',<,> are replaced by their XML equivalent:

&, ", ', <, >

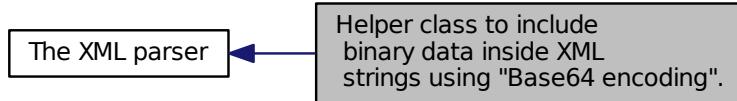
Using the "ToXMLStringTool class" and the "fprintf function" is THE most efficient way to produce VERY large XML documents VERY fast.

Note

If you are creating from scratch an XML file using the provided [XMLNode](#) class you must not use the "ToXMLStringTool" class (because the "XMLNode" class does the processing job for you during rendering).

7.13 Helper class to include binary data inside XML strings using "Base64 encoding".

Collaboration diagram for Helper class to include binary data inside XML strings using "Base64 encoding".:



Classes

- struct [XMLParserBase64Tool](#)

Helper class to include binary data inside XML strings using "Base64 encoding".

Typedefs

- typedef struct [XMLDLLENTRY](#)
[XMLParserBase64Tool](#) [XMLParserBase64Tool](#)

Helper class to include binary data inside XML strings using "Base64 encoding".

7.13.1 Detailed Description

7.13.2 Typedef Documentation

7.13.2.1 typedef struct XMLDLLENTRY XMLParserBase64Tool XMLParserBase64Tool

Helper class to include binary data inside XML strings using "Base64 encoding".

The "XMLParserBase64Tool" class allows you to include any binary data (images, sounds,...) into an XML document using "Base64 encoding". This class is completely separated from the rest of the xmlParser library and can be removed without any problem. To include some binary data into an XML file, you must convert the binary data into standard text (using "encode"). To retrieve the original binary data from the b64-encoded text included inside the XML file, use "decode". Alternatively, these functions can also be used to "encrypt/decrypt" some critical data contained inside the XML (it's not a strong encryption at all, but sometimes it can be useful).

Chapter 8

Namespace Documentation

8.1 unisys Namespace Reference

This namespace.

Classes

- class [Database](#)

This class is a database handle class.

- class [Query](#)

- class [Updater](#)

- class [BatchInsert](#)

- class [DataObj](#)

- class [Miriam](#)

This class is for miriam cross reference annotation.

- class [IdRef](#)

Identifier reference class.

- class [PEIdRef](#)

Identifier reference class specific to physicalentity collection namespace.

- class [IntIdRef](#)

Identifier reference class specific to interaction collection namespace.

- class [OntIdRef](#)

Identifier reference class specific to obo collection namespace.

- class [Tracking](#)

Data updating history class.

- class [LiteralBSON](#)

Specific BSON object to manage general [Literal](#) data object.

- class [Literal](#)

Literal class.

- class [Xref](#)

The C++ representative class for Cross reference data class.

- class [Score](#)

The C++ representative class for [Score](#) class in data structure.

- class [BioSource](#)

The C++ representative class for Biological Source data class. This class is designed to manage the biological data of source organism.

- class [Evidence](#)

- class [Annotation](#)

The C++ representative class for evidence data class.
- class [OntoRelationship](#)

The C++ representative class for annotation data class.
- class [Relation](#)

The C++ representative class for relationship data class.
- class [CellularLocation](#)

The C++ representative class for cellular location data class.
- class [MathML](#)

The C++ representative class for MathML data class.
- class [KineticParameter](#)

The C++ representative class for kinetic parameter data class.
- class [SubRegion](#)

The C++ representative class for sub-region data class.
- class [Metadata](#)

The C++ representative class for metadata data class.
- class [Object](#)

Root of all object classes This class does not have any interfaces.
- class [Ontology](#)

Ontology data class.
- class [BioObject](#)

This class is for miriam cross reference annotation.
- class [PhysicalEntity](#)

This class is for miriam cross reference annotation.
- class [SmallMolecule](#)

This class is for miriam cross reference annotation.
- class [DNA](#)

This class is for miriam cross reference annotation.
- class [DNAResRegion](#)

This class is for miriam cross reference annotation.
- class [RNA](#)

This class is for miriam cross reference annotation.
- class [Protein](#)

This class is for miriam cross reference annotation.
- class [Complex](#)

This class is for miriam cross reference annotation.
- class [Interaction](#)

This class is for miriam cross reference annotation.
- class [Control](#)

This class is for miriam cross reference annotation.
- class [MolecularInteraction](#)

This class is for miriam cross reference annotation.
- class [GeneticInteraction](#)

This class is for miriam cross reference annotation.
- class [Conversion](#)

This class is for miriam cross reference annotation.
- class [BiochemicalReaction](#)

This class is for miriam cross reference annotation.
- class [Transport](#)

This class is for miriam cross reference annotation.

- class [BiochemicalReactionWithTransport](#)

This class is for miriam cross reference annotation.

- class [UniSysError](#)
- class [ConnectionError](#)
- class [UpdateError](#)
- class [QueryError](#)
- class [DataError](#)
- class [ParsingError](#)
- class [RESTServiceError](#)
- class [BIOPAX_obj](#)

This class is used to parse OWL format.

- class [BIOPAX2](#)

This class is used to parse OWL format.

- class [ChEBIOWLClass](#)

To handle owl:class tag of ChEBI OWL format.

- class [ChEBIOWL](#)

Used to parse ChEBI OWL format.

- class [Stanza](#)

This class is used to parse OWL format.

- class [OBOXML](#)

This class is used to parse OWL format.

- class [PSIMI](#)

A PSIMI class is used for a data in PSI-MI XML2.5 file format.

- class [RestBioMartResult](#)

General class for taking care the response from BioPortal's REST service.

- class [RestBioMart](#)

General class for taking care the response from BioPortal's REST service.

- class [RestBioPortalResult](#)

General class for taking care the response from BioPortal's REST service.

- class [RestBioPortalSearchResult](#)

General class for taking care the response from BioPortal's search command of REST service.

- class [ClassBean](#)

This class is for taking care of the specific structure of response data call classBean.

- class [RestBioPortalTermResult](#)

This class is for parsing the response of term command.

- class [RestBioPortal](#)

The main class of BioPortal REST service library.

- class [RestMiriam](#)

General class for taking care the response from BioPortal's REST service.

- class [REST](#)

Used for managing the request and response of REST service.

Enumerations

- enum [dbStatus](#) { `DB_OK`, `DB_CANNOT_CONNECT`, `DB_CANNOT_INSERT` }

Functions

- std::string `setMiriamURN` (std::string db, XMLNode node)
- template<class type >
mongo::BSONArray `toBSONArray` (type const &input)
- template<class type >
std::set< type > `BSONTOSet` (mongo::BSONObj const &bsonObj)
- template<class type >
std::vector< type > `BSONTovector` (mongo::BSONObj const &bsonObj)

8.1.1 Detailed Description

This namespace. some description.

8.1.2 Enumeration Type Documentation

8.1.2.1 enum unisys::dbStatus

Enumerator:

`DB_OK`
`DB_CANNOT_CONNECT`
`DB_CANNOT_INSERT`

8.1.3 Function Documentation

8.1.3.1 template<class type > std::set<type> `unisys::BSONTOSet` (mongo::BSONObj const & `bsonObj`)

8.1.3.2 template<class type > std::vector<type> `unisys::BSONTovector` (mongo::BSONObj const & `bsonObj`)

8.1.3.3 std::string `unisys::setMiriamURN` (std::string `db`, XMLNode `node`)

8.1.3.4 template<class type > mongo::BSONArray `unisys::toBSONArray` (type const & `input`)

Chapter 9

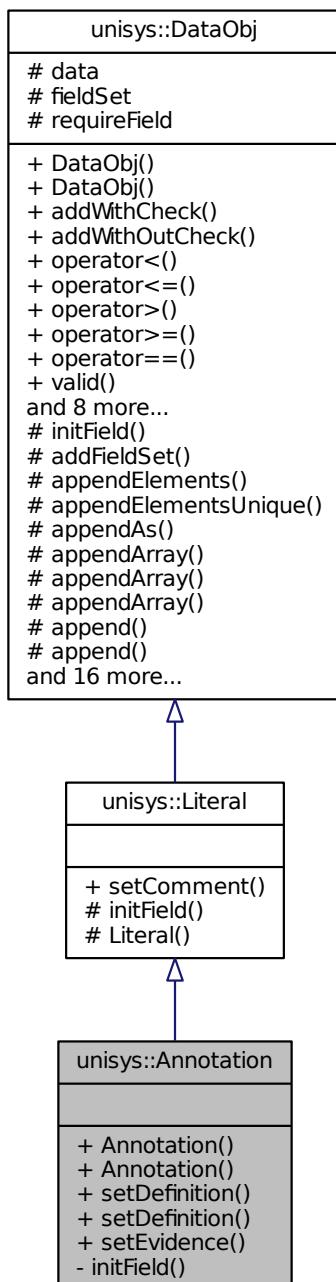
Class Documentation

9.1 unisys::Annotation Class Reference

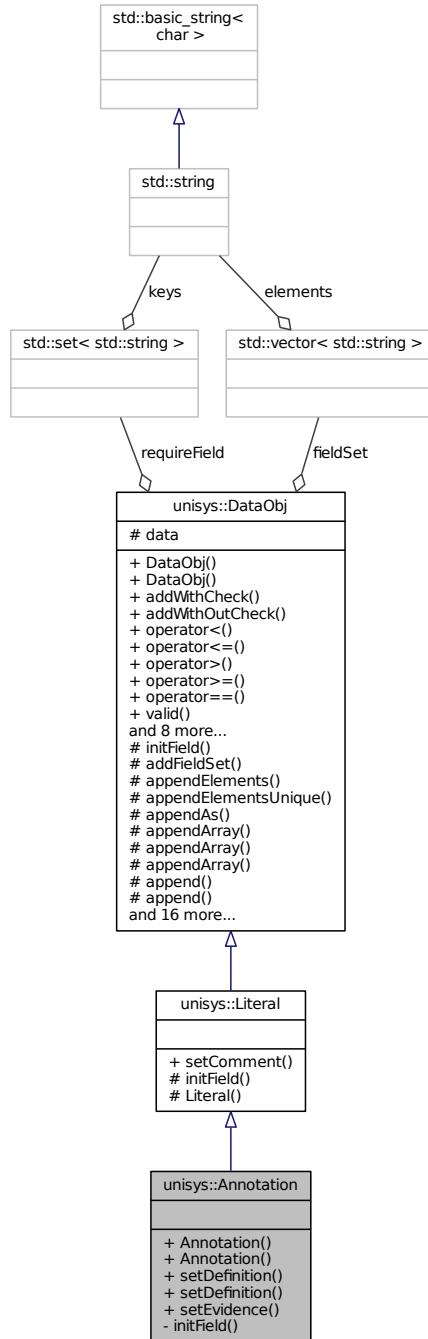
The C++ representative class for annotation data class.

```
#include <LitClass.h>
```

Inheritance diagram for unisys::Annotation:



Collaboration diagram for unisys::Annotation:



Public Member Functions

- [Annotation \(\)](#)
Default constructor.
- [Annotation \(mongo::BSONObj const &bsonObj\)](#)
Overloaded constructor is used when retrieving data in bson object from database and transform to C++ object.
- void [setDefinition \(OntoldRef &ontoldRef\)](#)

- void [setDefinition](#) (std::string const &ontold)
- void [setEvidence](#) (Evidence &evidence)

Private Member Functions

- void [initField](#) ()

function for init field in the object

Additional Inherited Members

9.1.1 Detailed Description

The C++ representative class for annotation data class.

```
 BSON structure:
{
    comment: <string>,
    definition: {$ref: <collname>, $id: <idvalue>}, #format by idref class
    evidence: <EvidenceBSONStructure>
}
```

9.1.2 Constructor & Destructor Documentation

9.1.2.1 [unisys::Annotation::Annotation\(\)](#)

Default constructor.

9.1.2.2 [unisys::Annotation::Annotation\(mongo::BSONObj const & bsonObj \)](#)

Overloaded constructor is used when retrieving data in bson object from database and transform to C++ object.

9.1.3 Member Function Documentation

9.1.3.1 [void unisys::Annotation::initField\(\) \[private\], \[virtual\]](#)

function for init field in the object

Reimplemented from [unisys::Literal](#).

9.1.3.2 [void unisys::Annotation::setDefinition\(OntoldRef & ontoldRef \)](#)

9.1.3.3 [void unisys::Annotation::setDefinition\(std::string const & ontold \)](#)

9.1.3.4 [void unisys::Annotation::setEvidence\(Evidence & evidence \)](#)

The documentation for this class was generated from the following file:

- [LitClass.h](#)

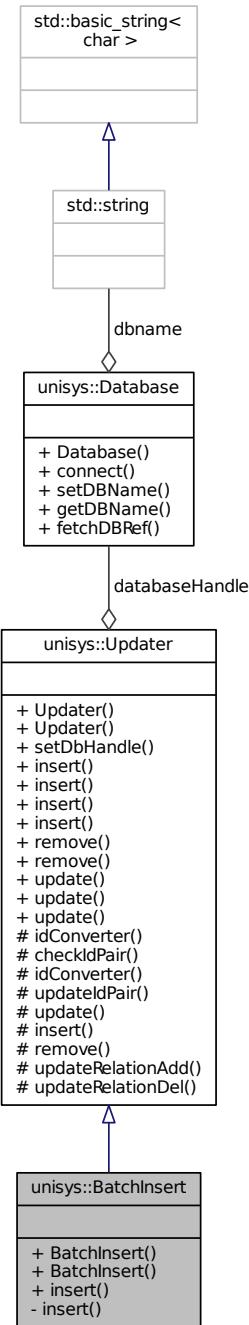
9.2 unisys::BatchInsert Class Reference

```
#include <updater.h>
```

Inheritance diagram for unisys::BatchInsert:



Collaboration diagram for unisys::BatchInsert:



Public Member Functions

- `BatchInsert ()`
- `BatchInsert (Database *databaseHandlePt)`
- template<class myType >
`std::vector< mongo::BSONObj > insert (std::vector< myType > dataList, std::string idNS="", unsigned int startId=0, std::string idPrefix="", unsigned int version=0, bool withVer=true, bool dryrun=false) throw (UpdateError, DataError)`

Private Member Functions

- std::vector< mongo::BSONObj > [insert](#) (std::vector< mongo::BSONObj > dataList, std::string idNS="", int startId=1, bool dryrun=false) throw (UpdateError, DataError)

Additional Inherited Members

9.2.1 Constructor & Destructor Documentation

9.2.1.1 [unisys::BatchInsert::BatchInsert\(\)](#)

9.2.1.2 [unisys::BatchInsert::BatchInsert\(Database * databaseHandlePt \)](#)

9.2.2 Member Function Documentation

9.2.2.1 std::vector<mongo::BSONObj> [unisys::BatchInsert::insert\(std::vector< mongo::BSONObj > dataList, std::string idNS = " ", int startId = 1, bool dryrun = false \) throw \(UpdateError, DataError\) \[private\]](#)

9.2.2.2 template<class myType > std::vector<mongo::BSONObj> [unisys::BatchInsert::insert\(std::vector< myType > dataList, std::string idNS = " ", unsigned int startId = 0, std::string idPrefix = " ", unsigned int version = 0, bool withVer = true, bool dryrun = false \) throw \(UpdateError, DataError\)](#)

The documentation for this class was generated from the following file:

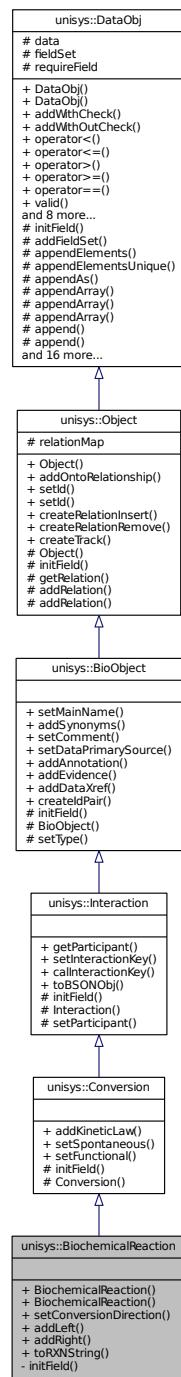
- [updater.h](#)

9.3 unisys::BiochemicalReaction Class Reference

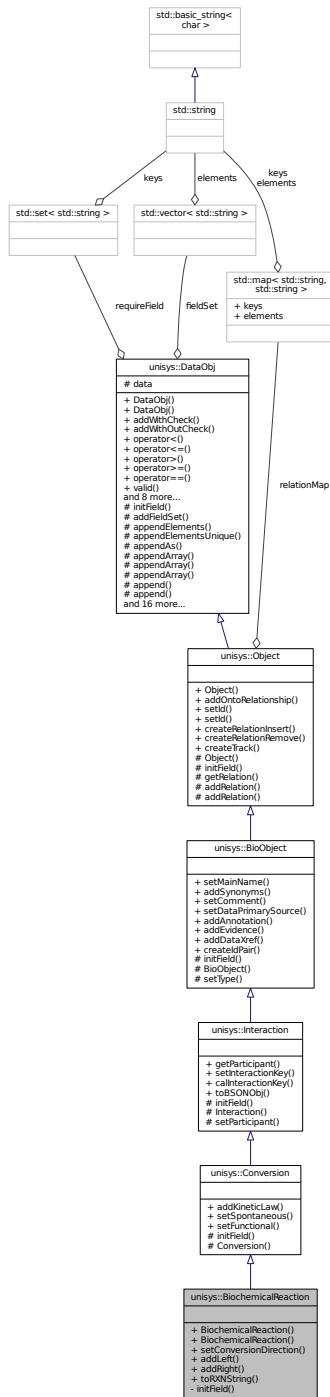
This class is for miriam cross reference annotation.

```
#include <ObjClass.h>
```

Inheritance diagram for unisys::BiochemicalReaction:



Collaboration diagram for unisys::BiochemicalReaction:



Public Member Functions

- [BiochemicalReaction \(\)](#)
Default constructor.
 - [BiochemicalReaction \(mongo::BSONObj const &bsonObj\)](#)
Overloaded constructor is used when retrieving data in bson object from database and transform to C++ object.
 - void [setConversionDirection \(std::string const &direction=""\)](#)

- void `addLeft (PEIdRef &peldRef, double coefficient)`
Add left participants of reaction.
- void `addRight (PEIdRef &peldRef, double coefficient)`
Add right participants of reaction.
- std::string `toRXNString (std::string const &compartment) const`

Private Member Functions

- void `initField ()`

Additional Inherited Members

9.3.1 Detailed Description

This class is for miriam cross reference annotation.

```
BSON structure:
{
    _id: <string>, #mandatory
    type: <string>,
    ontologyRelationship: {<RelationshipBOSON>, <RelationshipBOSON>, ...},
    name: {<string>, <string>, ...}
    comment: <string>,
    dataPrimarySource: <XrefBOSON>,
    functionAnnotation: {<AnnotationBOSON>, <AnnotationBOSON>, ...},
    evidence: {<EvidenceBOSON>, <EvidenceBOSON>, ...},
    dataxref: {<XrefBOSON>, <XrefBOSON>, ...},
    relation: {<StoichiometryBOSON>, <StoichiometryBOSON>, ...},
    interactionKey: <string>,
    conversionDirection: <DbRef>,
    kineticLaw: {<MathMLBOSON>, <MathMLBOSON>, ...}
    spontaneous: <string>,
    functional: <string>
}
```

9.3.2 Constructor & Destructor Documentation

9.3.2.1 unisys::BiochemicalReaction::BiochemicalReaction ()

Default constructor.

9.3.2.2 unisys::BiochemicalReaction::BiochemicalReaction (mongo::BSONObj const & bsonObj)

Overloaded constructor is used when retrieving data in boson object from database and transform to C++ object.

9.3.3 Member Function Documentation

9.3.3.1 void unisys::BiochemicalReaction::addLeft (PEIdRef & peldRef, double coefficient)

Add left participants of reaction.

9.3.3.2 void unisys::BiochemicalReaction::addRight (PEIdRef & peldRef, double coefficient)

Add right participants of reaction.

9.3.3.3 void unisys::BiochemicalReaction::initField() [private], [virtual]

Reimplemented from [unisys::Conversion](#).

9.3.3.4 void unisys::BiochemicalReaction::setConversionDirection(std::string const & *direction* = "=")

9.3.3.5 std::string unisys::BiochemicalReaction::toRXNString(std::string const & *compartment*) const

The documentation for this class was generated from the following file:

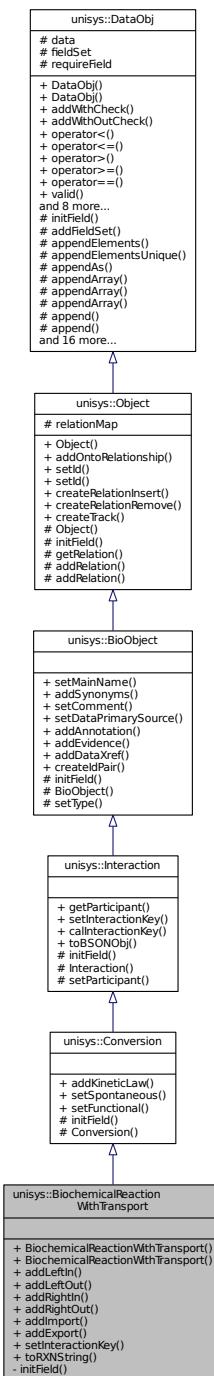
- [ObjClass.h](#)

9.4 unisys::BiochemicalReactionWithTransport Class Reference

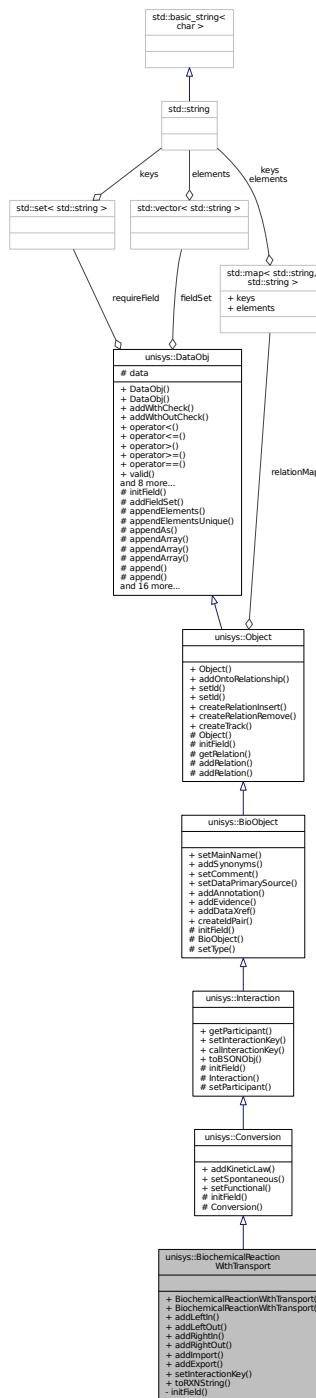
This class is for miriam cross reference annotation.

```
#include <ObjClass.h>
```

Inheritance diagram for unisys::BiochemicalReactionWithTransport:



Collaboration diagram for unisys::BiochemicalReactionWithTransport:



Public Member Functions

- `BiochemicalReactionWithTransport ()`

Default constructor.

- `BiochemicalReactionWithTransport (mongo::BSONObj const &bsonObj)`

Overloaded constructor is used when retrieving data in bson object from database and transform to C++ object.

- `void addLeftIn (PEIdRef &peldRef, double coefficient)`

- void `addLeftOut (PEIdRef &peldRef, double coefficient)`
- void `addRightIn (PEIdRef &peldRef, double coefficient)`
- void `addRightOut (PEIdRef &peldRef, double coefficient)`
- void `addImport (PEIdRef &peldRef, double coefficient)`
- void `addExport (PEIdRef &peldRef, double coefficient)`
- void `setInteractionKey ()`
- std::string `toRXNString (std::string const &outside, std::string const &inside) const`
converse to a string of reaction

Private Member Functions

- void `initField ()`

Additional Inherited Members

9.4.1 Detailed Description

This class is for miriam cross reference annotation.

```
BSON structure:
{
    _id: <string>, #madatory
    type: <string>,
    ontologyRelationship: {<RelationshipBOSON>, <RelationshipBOSON>, ...},
    name: {<string>, <string>, ...}
    comment: <string>,
    dataPrimarySource: <XrefBOSON>,
    functionAnnotation: {<AnnotationBOSON>, <AnnotationBOSON>, ...},
    evidence: {<EvidenceBOSON>, <EvidenceBOSON>, ...},
    dataxref: {<XrefBOSON>, <XrefBOSON>, ...},
    participant: {<StoichiometryBOSON>, <StoichiometryBOSON>, ...},
    interactionKey: <string>,
    conversionDirection: <DbRef>,
    kineticLaw: {<MathMLBOSON>, <MathMLBOSON>, ...}
    spontaneous: <string>,
    functional: <string>
}
```

9.4.2 Constructor & Destructor Documentation

9.4.2.1 unisys::BiochemicalReactionWithTransport::BiochemicalReactionWithTransport ()

Default constructor.

9.4.2.2 unisys::BiochemicalReactionWithTransport::BiochemicalReactionWithTransport (mongo::BSONObj const & bsonObj)

Overloaded constructor is used when retrieving data in boson object from database and transform to C++ object.

9.4.3 Member Function Documentation

9.4.3.1 void unisys::BiochemicalReactionWithTransport::addExport (PEIdRef & peldRef, double coefficient)

9.4.3.2 void unisys::BiochemicalReactionWithTransport::addImport (PEIdRef & peldRef, double coefficient)

9.4.3.3 void unisys::BiochemicalReactionWithTransport::addLeftIn (PEIdRef & peldRef, double coefficient)

9.4.3.4 void unisys::BiochemicalReactionWithTransport::addLeftOut (PEIdRef & *peldRef*, double *coefficient*)

9.4.3.5 void unisys::BiochemicalReactionWithTransport::addRightIn (PEIdRef & *peldRef*, double *coefficient*)

9.4.3.6 void unisys::BiochemicalReactionWithTransport::addRightOut (PEIdRef & *peldRef*, double *coefficient*)

9.4.3.7 void unisys::BiochemicalReactionWithTransport::initField () [private], [virtual]

Reimplemented from [unisys::Conversion](#).

9.4.3.8 void unisys::BiochemicalReactionWithTransport::setInteractionKey ()

Reimplemented from [unisys::Interaction](#).

9.4.3.9 std::string unisys::BiochemicalReactionWithTransport::toRXNString (std::string const & *outside*, std::string const & *inside*) const

converse to a string of reaction

Parameters

<i>outside</i>
<i>inside</i>

The documentation for this class was generated from the following file:

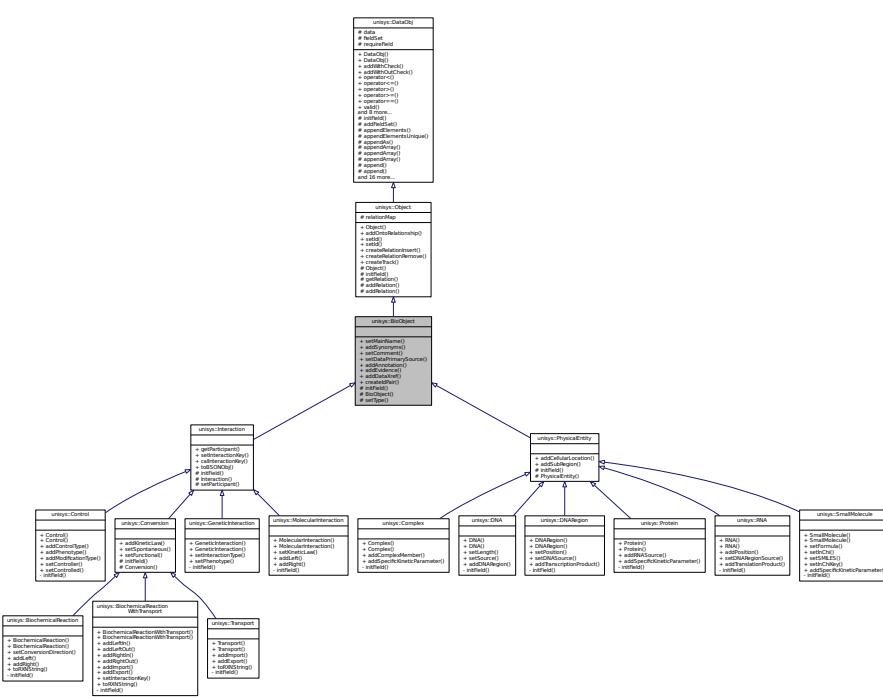
- [ObjClass.h](#)

9.5 unisys::BioObject Class Reference

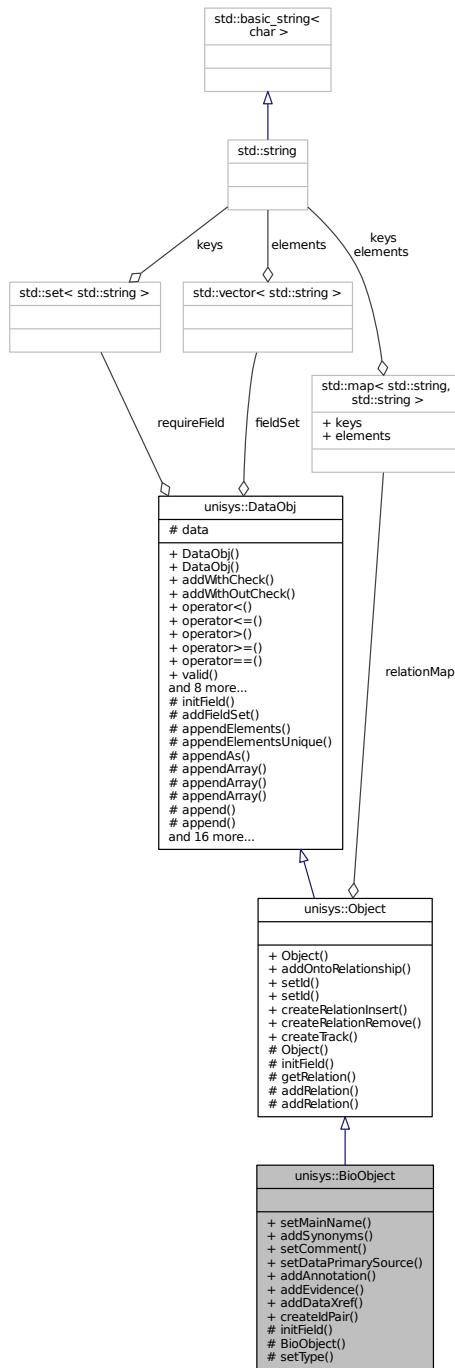
This class is for miriam cross reference annotation.

```
#include <ObjClass.h>
```

Inheritance diagram for unisys::BioObject:



Collaboration diagram for unisys::BioObject:



Public Member Functions

- void [setMainName](#) (std::string const &name)
- void [addSynonyms](#) (std::string const &name, std::string const &sep_char=",")
- void [setComment](#) (std::string const &comment)
- void [setDataPrimarySource](#) (Xref xref)
- void [addAnnotation](#) (Annotation &annotation)

- void [addEvidence \(Evidence &evidence\)](#)
- void [addDataXref \(Xref xref\)](#)
- mongo::BSONObj [createIdPair \(bool strict=true\) const](#)

Protected Member Functions

- void [initField \(\)](#)
- [BioObject \(\)](#)
- void [setType \(std::string const &name\)](#)

Additional Inherited Members

9.5.1 Detailed Description

This class is for miriam cross reference annotation.

```
BSON structure:
{
    _id: <string>, #madatory
    ontologyRelationship: {<RelationshipBOSON>, <RelationshipBOSON>, ...},
    name: {<string>, <string>, ...}
    type: <string>, # pathway, collection, chemicalentity, dna, dnaregion, rna, protien, complex, cont
          genericinteraction, biochemicalreaction, transport and reactiontransport
    comment: <string>,
    dataPrimarySource: <XrefBOSON>,
    functionAnnotation: {<AnnotationBOSON>, <AnnotationBOSON>, ...},
    evidence: {<EvidenceBOSON>, <EvidenceBOSON>, ...},
    dataXref: {<XrefBOSON>, <XrefBOSON>, ...}
}
```

9.5.2 Constructor & Destructor Documentation

9.5.2.1 [unisys::BioObject::BioObject \(\) \[protected\]](#)

9.5.3 Member Function Documentation

9.5.3.1 [void unisys::BioObject::addAnnotation \(Annotation &annotation \)](#)

9.5.3.2 [void unisys::BioObject::addDataXref \(Xref xref \)](#)

9.5.3.3 [void unisys::BioObject::addEvidence \(Evidence &evidence \)](#)

9.5.3.4 [void unisys::BioObject::addSynonyms \(std::string const & name, std::string const & sep_char = " , " \)](#)

9.5.3.5 [mongo::BSONObj unisys::BioObject::createIdPair \(bool strict = true \) const](#)

9.5.3.6 [void unisys::BioObject::initField \(\) \[protected\], \[virtual\]](#)

Reimplemented from [unisys::Object](#).

Reimplemented in [unisys::BiochemicalReactionWithTransport](#), [unisys::Transport](#), [unisys::BiochemicalReaction](#), [unisys::Conversion](#), [unisys::GeneticInteraction](#), [unisys::MolecularInteraction](#), [unisys::Control](#), [unisys::Interaction](#), [unisys::Complex](#), [unisys::Protein](#), [unisys::RNA](#), [unisys::DNARegion](#), [unisys::DNA](#), [unisys::SmallMolecule](#), and [unisys::PhysicalEntity](#).

9.5.3.7 void unisys::BioObject::setComment (std::string const & *comment*)

9.5.3.8 void unisys::BioObject::setDataPrimarySource (Xref *xref*)

9.5.3.9 void unisys::BioObject::setMainName (std::string const & *name*)

9.5.3.10 void unisys::BioObject::setType (std::string const & *name*) [protected]

The documentation for this class was generated from the following file:

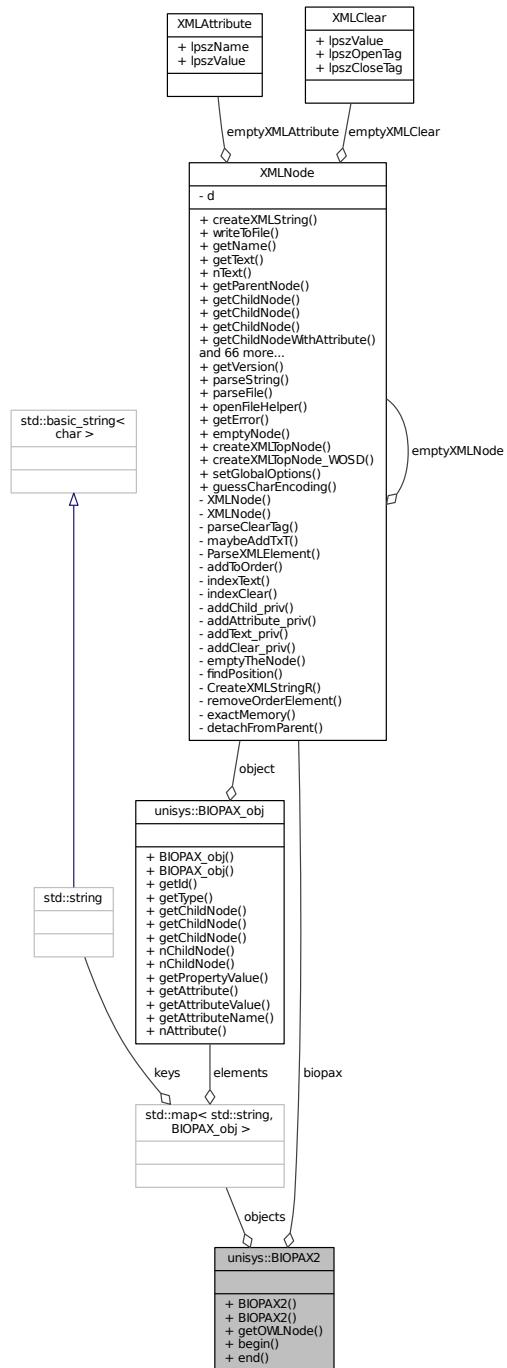
- [ObjClass.h](#)

9.6 unisys::BIOPAX2 Class Reference

This class is used to parse OWL format.

```
#include <biopax.h>
```

Collaboration diagram for unisys::BIOPAX2:



Public Types

- `typedef std::map< std::string, BIOPAX_obj > biopaxObjMap`

Public Member Functions

- **BIOPAX2 ()**
Defualt constructor.
- **BIOPAX2 (std::string fileName)**
Constructor with fileName as parameter.
- **XMLNode getOWLNode () const**
- **biopaxObjMap::const_iterator begin () const**
- **biopaxObjMap::const_iterator end () const**

Private Attributes

- **XMLNode biopax**
- **std::map< std::string, BIOPAX_obj > objects**
All objects map with their ids.

9.6.1 Detailed Description

This class is used to parse OWL format.

Data structure using to store information from Tag-Value pair line except; name and id tag

9.6.2 Member Typedef Documentation

9.6.2.1 **typedef std::map<std::string, BIOPAX_obj> unisys::BIOPAX2::biopaxObjMap**

9.6.3 Constructor & Destructor Documentation

9.6.3.1 **unisys::BIOPAX2::BIOPAX2 ()**

Defualt constructor.

9.6.3.2 **unisys::BIOPAX2::BIOPAX2 (std::string fileName)**

Constructor with fileName as parameter.

9.6.4 Member Function Documentation

9.6.4.1 **biopaxObjMap::const_iterator unisys::BIOPAX2::begin () const**

9.6.4.2 **biopaxObjMap::const_iterator unisys::BIOPAX2::end () const**

9.6.4.3 **XMLNode unisys::BIOPAX2::getOWLNode () const**

9.6.5 Member Data Documentation

9.6.5.1 **XMLNode unisys::BIOPAX2::biopax [private]**

9.6.5.2 **std::map<std::string, BIOPAX_obj> unisys::BIOPAX2::objects [private]**

All objects map with their ids.

The documentation for this class was generated from the following file:

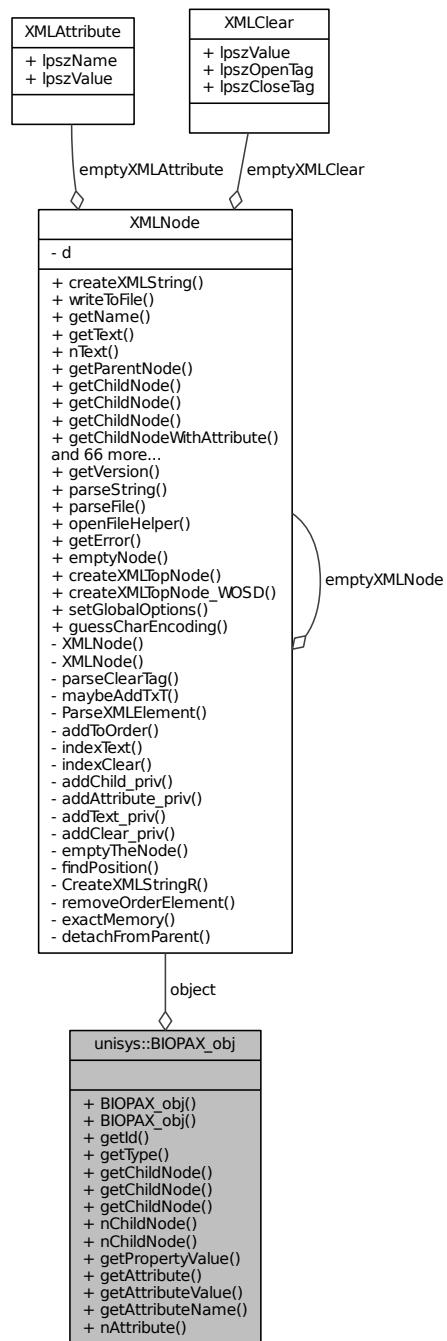
- [biopax.h](#)

9.7 unisys::BIOPAX_obj Class Reference

This class is used to parse OWL format.

```
#include <biopax.h>
```

Collaboration diagram for unisys::BIOPAX_obj:



Public Member Functions

- [BIOPAX_obj \(\)](#)
Defualt constructor.
- [BIOPAX_obj \(XMLNode node\)](#)
- [std::string getId \(\) const](#)
- [std::string getType \(\) const](#)
- [XMLNode getChildNode \(int i=0\) const](#)
- [XMLNode getChildNode \(std::string name, int i\) const](#)
- [XMLNode getChildNode \(std::string name\) const](#)
- [int nChildNode \(\) const](#)
- [int nChildNode \(std::string name\) const](#)
- [std::string getPropertyValue \(std::string name\) const](#)
- [XMLAttribute getAttribute \(int i=0\) const](#)
- [std::string getAttributeValue \(int i=0\) const](#)
- [std::string getAttributeName \(int i=0\) const](#)
- [int nAttribute \(\) const](#)

Private Attributes

- [XMLNode object](#)

9.7.1 Detailed Description

This class is used to parse OWL format.

some description.

9.7.2 Constructor & Destructor Documentation

9.7.2.1 unisys::BIOPAX_obj::BIOPAX_obj ()

Defualt constructor.

9.7.2.2 unisys::BIOPAX_obj::BIOPAX_obj (XMLNode node)

9.7.3 Member Function Documentation

9.7.3.1 XMLAttribute unisys::BIOPAX_obj::getAttribute (int i = 0) const

9.7.3.2 std::string unisys::BIOPAX_obj::getAttributeName (int i = 0) const

9.7.3.3 std::string unisys::BIOPAX_obj::getAttributeValue (int i = 0) const

9.7.3.4 XMLNode unisys::BIOPAX_obj::getChildNode (int i = 0) const

9.7.3.5 XMLNode unisys::BIOPAX_obj::getChildNode (std::string name, int i) const

9.7.3.6 XMLNode unisys::BIOPAX_obj::getChildNode (std::string name) const

9.7.3.7 std::string unisys::BIOPAX_obj::getId () const

9.7.3.8 std::string unisys::BIOPAX_obj::getPropertyValue (std::string name) const

9.7.3.9 `std::string unisys::BIOPAX_obj::getType() const`

9.7.3.10 `int unisys::BIOPAX_obj::nAttribute() const`

9.7.3.11 `int unisys::BIOPAX_obj::nChildNode() const`

9.7.3.12 `int unisys::BIOPAX_obj::nChildNode(std::string name) const`

9.7.4 Member Data Documentation

9.7.4.1 `XMLNode unisys::BIOPAX_obj::object [private]`

The documentation for this class was generated from the following file:

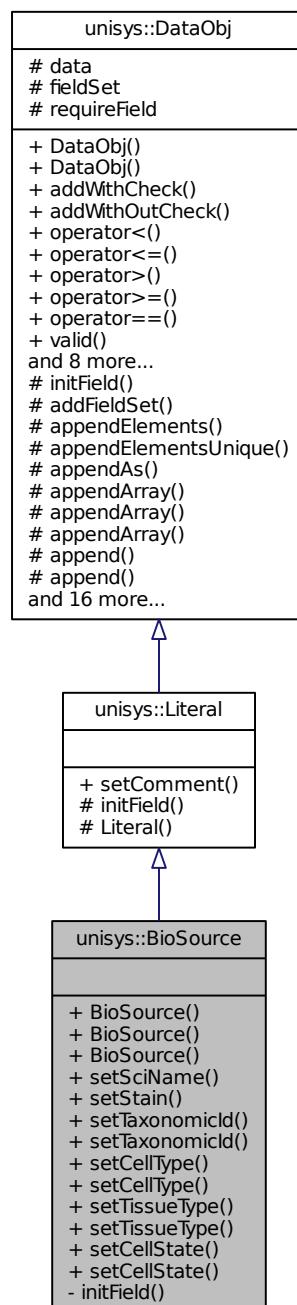
- [biopax.h](#)

9.8 unisys::BioSource Class Reference

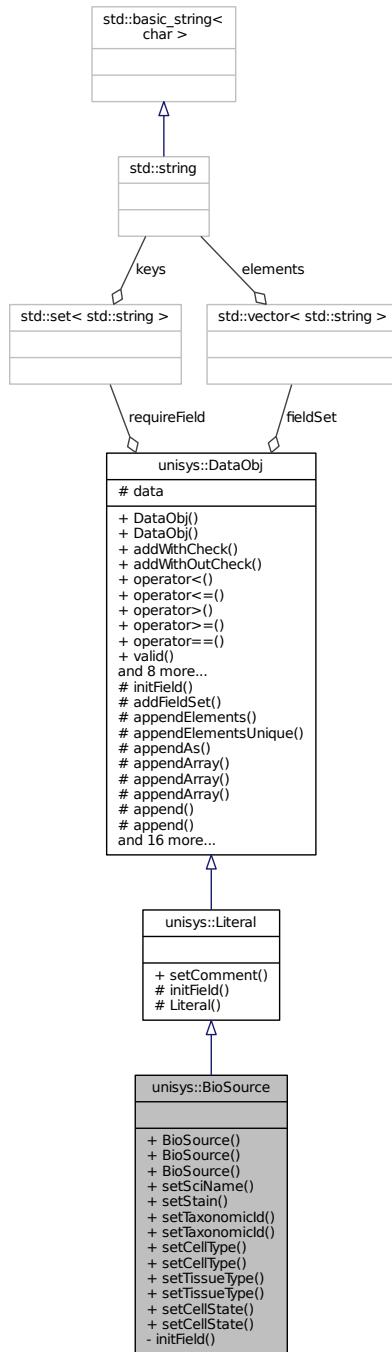
The C++ representative class for Biological Source data class. This class is designed to manage the biological data of source organism.

```
#include <LitClass.h>
```

Inheritance diagram for unisys::BioSource:



Collaboration diagram for unisys::BioSource:



Public Member Functions

- **BioSource ()**
Default constructor.
- **BioSource (mongo::BSONObj const &bsonObj)**
Overloaded constructor is used when retrieving data in bson object from database and transform to C++ object.
- **BioSource (std::string const &genus, std::string const &species=".sp")**

- void `setSciName` (std::string const &genus, std::string const &species=".sp")
- void `setStain` (std::string const &strain)
- void `setTaxonomicId` (`Xref` &taxonomicId)
- void `setTaxonomicId` (std::string const &taxonomicId, std::string const &detail "")
- void `setCellType` (`OntoldRef` &ontoldRef)
- void `setCellType` (std::string const &ontold)
- void `setTissueType` (`OntoldRef` &ontoldRef)
- void `setTissueType` (std::string const &ontold)
- void `setCellState` (`OntoldRef` &ontoldRef)
- void `setCellState` (std::string const &ontold)

Private Member Functions

- void `initField` ()
function for init field in the object

Additional Inherited Members

9.8.1 Detailed Description

The C++ representative class for Biological Source data class. This class is designed to manage the biological data of source organism.

```
 BSON structure:
{
    comment: <string>,
    genus: <string>,
    species: <string>,
    strain: <string>,
    taxonomicId: <XRefBSONStructure>
    cellType: {$ref: <collname>, $id: <idvalue>}, #format by idref class
    tissueType: {$ref: <collname>, $id: <idvalue>}, #format by idref class
    cellState: {$ref: <collname>, $id: <idvalue>}, #format by idref class
}
```

9.8.2 Constructor & Destructor Documentation

9.8.2.1 unisys::BioSource::BioSource()

Default constructor.

9.8.2.2 unisys::BioSource::BioSource(mongo::BSONObj const & bsonObj)

Overloaded constructor is used when retrieving data in boson object from database and tranform to C++ object.

9.8.2.3 unisys::BioSource::BioSource(std::string const & genus, std::string const & species = ".sp")

9.8.3 Member Function Documentation

9.8.3.1 void unisys::BioSource::initField() [private], [virtual]

function for init field in the object

Reimplemented from [unisys::Literal](#).

9.8.3.2 void unisys::BioSource::setCellState (*OntoldRef & ontoldRef*)

9.8.3.3 void unisys::BioSource::setCellState (std::string const & *ontold*)

9.8.3.4 void unisys::BioSource::setCellType (*OntoldRef & ontoldRef*)

9.8.3.5 void unisys::BioSource::setCellType (std::string const & *ontold*)

9.8.3.6 void unisys::BioSource::setSciName (std::string const & *genus*, std::string const & *species* = ".sp")

9.8.3.7 void unisys::BioSource::setStain (std::string const & *strain*)

9.8.3.8 void unisys::BioSource::setTaxonomicId (*Xref & taxonomicId*)

9.8.3.9 void unisys::BioSource::setTaxonomicId (std::string const & *taxonomicId*, std::string const & *detail* = " ")

9.8.3.10 void unisys::BioSource::setTissueType (*OntoldRef & ontoldRef*)

9.8.3.11 void unisys::BioSource::setTissueType (std::string const & *ontold*)

The documentation for this class was generated from the following file:

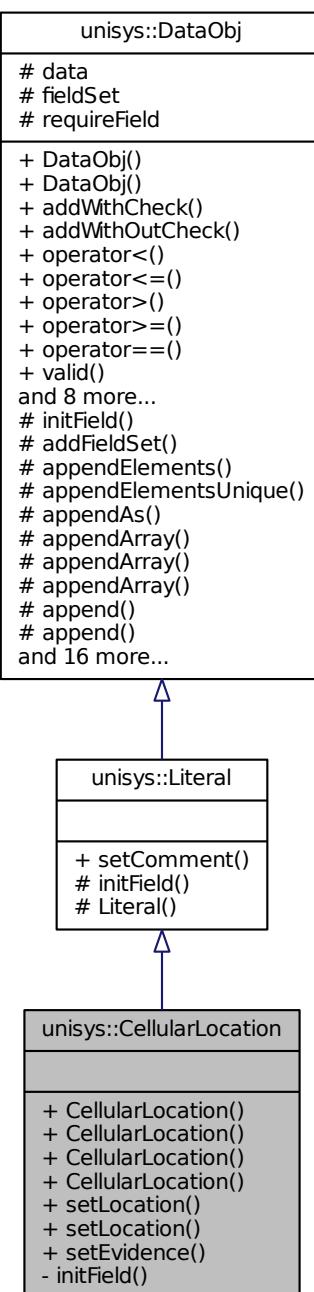
- [LitClass.h](#)

9.9 unisys::CellularLocation Class Reference

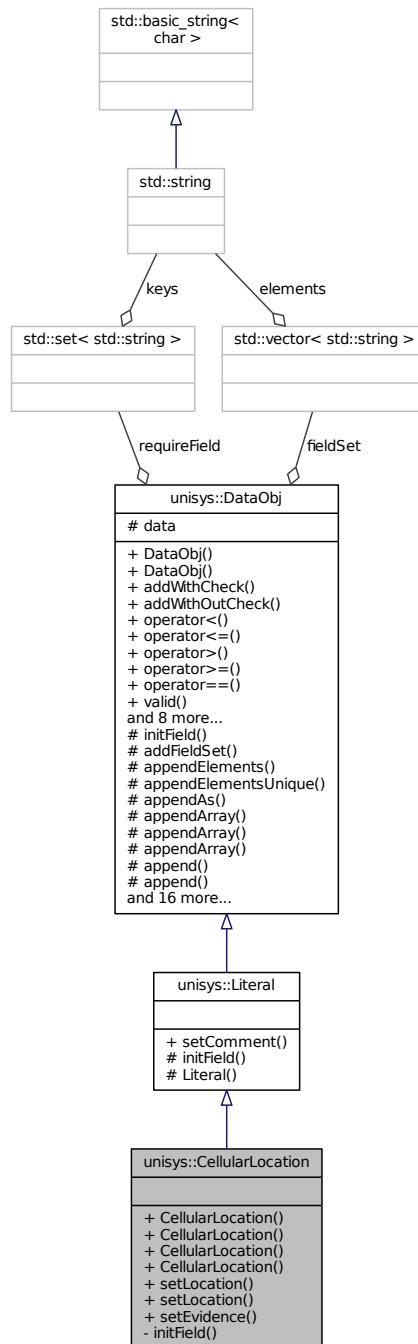
The C++ representative class for cellular location data class.

```
#include <LitClass.h>
```

Inheritance diagram for unisys::CellularLocation:



Collaboration diagram for unisys::CellularLocation:



Public Member Functions

- [CellularLocation \(\)](#)
Default constructor.
- [CellularLocation \(mongo::BSONObj const &bsonObj\)](#)
Overloaded constructor is used when retrieving data in bson object from database and transform to C++ object.
- [CellularLocation \(std::string const &ontold\)](#)

- `CellularLocation (std::string const &ontold, Evidence &evidence)`
- `void setLocation (OntoldRef &ontoldRef)`
- `void setLocation (std::string const &ontold)`
- `void setEvidence (Evidence &evidence)`

Private Member Functions

- `void initField ()`
function for init field in the object

Additional Inherited Members

9.9.1 Detailed Description

The C++ representative class for cellular location data class.

```
 BSON structure:
{
    comment: <string>,
    location: {$ref: <collname>, $id: <idvalue>}, #format by idref class
    evidence: <EvidenceBSONStructure>
}
```

9.9.2 Constructor & Destructor Documentation

9.9.2.1 unisys::CellularLocation::CellularLocation ()

Default constructor.

9.9.2.2 unisys::CellularLocation::CellularLocation (mongo::BSONObj const & bsonObj)

Overloaded constructor is used when retrieving data in boson object from database and tranform to C++ object.

9.9.2.3 unisys::CellularLocation::CellularLocation (std::string const & ontold)

9.9.2.4 unisys::CellularLocation::CellularLocation (std::string const & ontold, Evidence & evidence)

9.9.3 Member Function Documentation

9.9.3.1 void unisys::CellularLocation::initField () [private], [virtual]

function for init field in the object

Reimplemented from [unisys::Literal](#).

9.9.3.2 void unisys::CellularLocation::setEvidence (Evidence & evidence)

9.9.3.3 void unisys::CellularLocation::setLocation (OntoldRef & ontoldRef)

9.9.3.4 void unisys::CellularLocation::setLocation (std::string const & ontold)

The documentation for this class was generated from the following file:

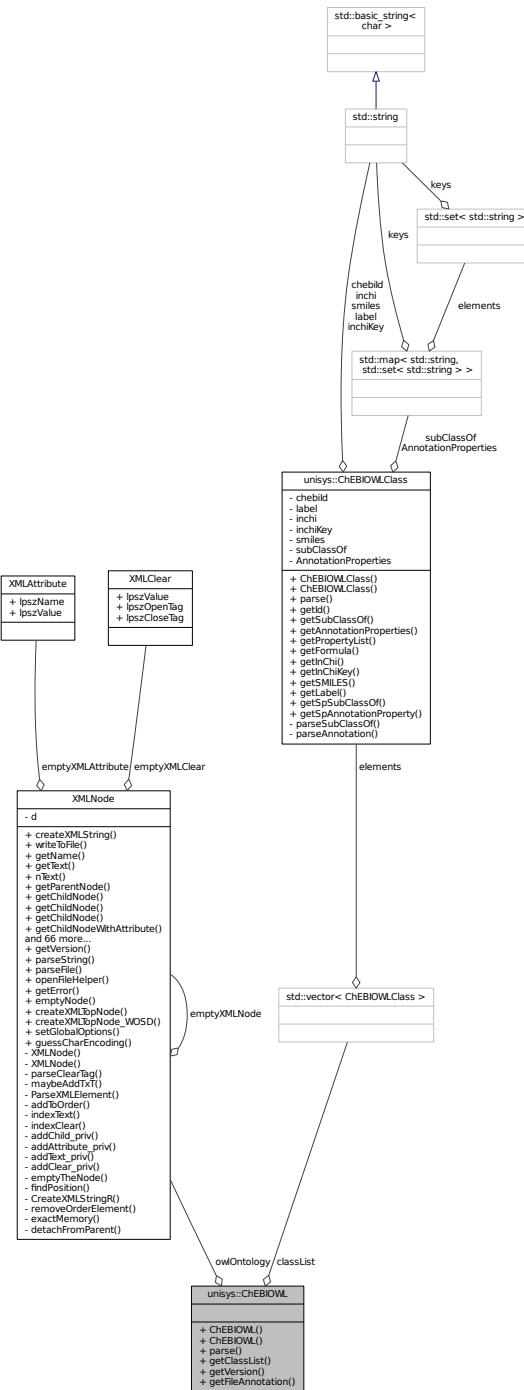
- [LitClass.h](#)

9.10 unisys::ChEBIOWL Class Reference

Used to parse ChEBI OWL format.

```
#include <chebiowl.h>
```

Collaboration diagram for unisys::ChEBIOWL:



Public Member Functions

- [ChEBIOWL \(\)](#)
- [ChEBIOWL \(std::string const &filename\)](#)
- void [parse \(std::string const &filename\)](#)
- std::vector< [ChEBIOWLClass](#) > [getClassList \(\) const](#)
- std::string [getVersion \(\) const](#)
- [XMLNode getFileAnnotation \(\) const](#)

Private Attributes

- [XMLNode owlOntology](#)
Ontology tag with file information.
- std::vector< [ChEBIOWLClass](#) > [classList](#)
vector of ChEBIOWLClass

9.10.1 Detailed Description

Used to parse ChEBI OWL format.

some description.

9.10.2 Constructor & Destructor Documentation

9.10.2.1 [unisys::ChEBIOWL::ChEBIOWL \(\)](#)

9.10.2.2 [unisys::ChEBIOWL::ChEBIOWL \(std::string const & filename \)](#)

9.10.3 Member Function Documentation

9.10.3.1 std::vector<[ChEBIOWLClass](#)> [unisys::ChEBIOWL::getClassList \(\) const](#)

9.10.3.2 [XMLNode unisys::ChEBIOWL::getFileAnnotation \(\) const](#)

9.10.3.3 std::string [unisys::ChEBIOWL::getVersion \(\) const](#)

9.10.3.4 void [unisys::ChEBIOWL::parse \(std::string const & filename \)](#)

9.10.4 Member Data Documentation

9.10.4.1 std::vector<[ChEBIOWLClass](#)> [unisys::ChEBIOWL::classList \[private\]](#)

vector of [ChEBIOWLClass](#)

9.10.4.2 [XMLNode unisys::ChEBIOWL::owlOntology \[private\]](#)

Ontology tag with file information.

The documentation for this class was generated from the following file:

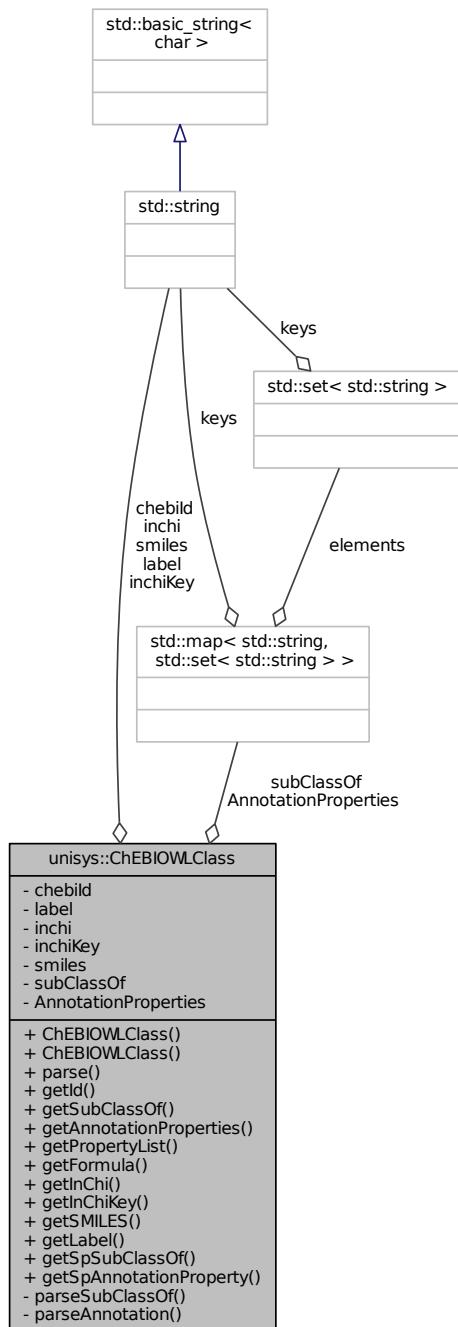
- [chebiowl.h](#)

9.11 unisys::ChEBIOWLClass Class Reference

To handle owl:class tag of ChEBI OWL format.

```
#include <chebiowl.h>
```

Collaboration diagram for unisys::ChEBIOWLClass:



Public Member Functions

- `ChEBIOWLClass ()`
- `ChEBIOWLClass (XMLNode xml)`
- `void parse (XMLNode xml) throw (ParsingError)`
- `std::string getId () const`
- `std::map< std::string, std::set< std::string > > getSubClassOf () const`
- `std::map< std::string, std::set< std::string > > getAnnotationProperties () const`
- `std::set< std::string > getPropertyList () const`
- `std::string getFormula () const`
- `std::string getInChi () const`
- `std::string getInChiKey () const`
- `std::string getSMILES () const`
- `std::string getLabel () const`
- `std::set< std::string > getSpSubClassOf (std::string const &subClassType) const`
- `std::set< std::string > getSpAnnotationProperty (std::string const &propertiesName) const`

Private Member Functions

- `void parseSubClassOf (XMLNode xml) throw (ParsingError)`
Parsing SubClassOf tag.
- `void parseAnnotation (XMLNode xml) throw (ParsingError)`
Parsing Annotation tag.

Private Attributes

- `std::string chebild`
ChEBI ID in /CHEBI:/+ format.
- `std::string label`
Name of ChEBI entity.
- `std::string inchi`
- `std::string inchiKey`
- `std::string smiles`
- `std::map< std::string, std::set< std::string > > subClassOf`
Relationship tag.
- `std::map< std::string, std::set< std::string > > AnnotationProperties`
Annotation tag except InChi, InChiKey and SMILES.

9.11.1 Detailed Description

To handle owl:class tag of ChEBI OWL format.
some description.

9.11.2 Constructor & Destructor Documentation

9.11.2.1 `unisys::ChEBIOWLClass::ChEBIOWLClass()`
 9.11.2.2 `unisys::ChEBIOWLClass::ChEBIOWLClass(XMLNode xml)`

9.11.3 Member Function Documentation

9.11.3.1 `std::map<std::string, std::set<std::string>> unisys::ChEBIOWLClass::getAnnotationProperties() const`
 9.11.3.2 `std::string unisys::ChEBIOWLClass::getFormula() const`
 9.11.3.3 `std::string unisys::ChEBIOWLClass::getId() const`
 9.11.3.4 `std::string unisys::ChEBIOWLClass::getInChi() const`
 9.11.3.5 `std::string unisys::ChEBIOWLClass::getInChiKey() const`
 9.11.3.6 `std::string unisys::ChEBIOWLClass::getLabel() const`
 9.11.3.7 `std::set<std::string> unisys::ChEBIOWLClass::getPropertyList() const`
 9.11.3.8 `std::string unisys::ChEBIOWLClass::getSMILES() const`
 9.11.3.9 `std::set<std::string> unisys::ChEBIOWLClass::getSpAnnotationProperty(std::string const & propertyName) const`
 9.11.3.10 `std::set<std::string> unisys::ChEBIOWLClass::getSpSubClassOf(std::string const & subClassType) const`
 9.11.3.11 `std::map<std::string, std::set<std::string>> unisys::ChEBIOWLClass::getSubClassOf() const`
 9.11.3.12 `void unisys::ChEBIOWLClass::parse(XMLNode xml) throw(ParsingError)`
 9.11.3.13 `void unisys::ChEBIOWLClass::parseAnnotation(XMLNode xml) throw(ParsingError) [private]`

Parsing [Annotation](#) tag.

9.11.3.14 `void unisys::ChEBIOWLClass::parseSubClassOf(XMLNode xml) throw(ParsingError) [private]`

Parsing SubClassOf tag.

9.11.4 Member Data Documentation

9.11.4.1 `std::map<std::string, std::set<std::string>> unisys::ChEBIOWLClass::AnnotationProperties [private]`

[Annotation](#) tag except InChi, InChiKey and SMILES.

9.11.4.2 `std::string unisys::ChEBIOWLClass::chebild [private]`

ChEBI ID in /CHEBI:+/ format.

9.11.4.3 `std::string unisys::ChEBIOWLClass::inchi [private]`

9.11.4.4 std::string unisys::ChEBIOWLClass::inchiKey [private]

9.11.4.5 std::string unisys::ChEBIOWLClass::label [private]

Name of ChEBI entity.

9.11.4.6 std::string unisys::ChEBIOWLClass::smiles [private]

9.11.4.7 std::map<std::string, std::set<std::string> > unisys::ChEBIOWLClass::subClassOf [private]

Relationship tag.

The documentation for this class was generated from the following file:

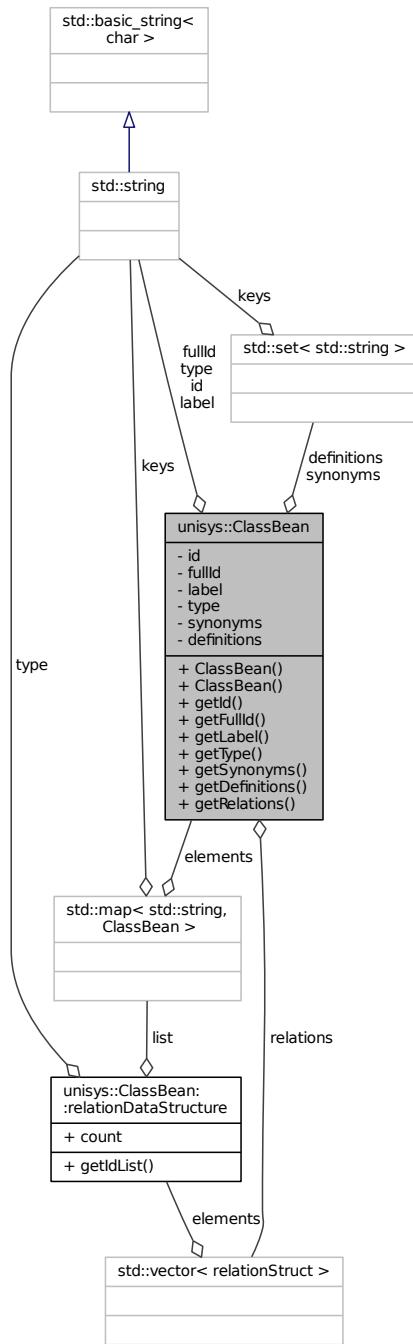
- [chebiowl.h](#)

9.12 unisys::ClassBean Class Reference

This class is for taking care of the specific structure of response data call classBean.

```
#include <restBioPortal.h>
```

Collaboration diagram for unisys::ClassBean:



Classes

- struct [relationDataStructure](#)

Public Types

- `typedef struct
unisys::ClassBean::relationDataStructure relationStruct`
For collecting the data in the relations node.

Public Member Functions

- `ClassBean ()`
Default constructor.
- `ClassBean (XMLNode xml)`
Overload constructor for parsing XML response.
- `std::string getId () const`
Return id.
- `std::string getFullId () const`
Return fullId.
- `std::string getLabel () const`
Return label.
- `std::string getType () const`
Return type.
- `std::set< std::string > getSynonyms () const`
Return the list of synonym.
- `std::set< std::string > getDefinitions () const`
Return the list of definition.
- `std::vector< relationStruct > getRelations () const`
Return the list of relation data structure.

Private Attributes

- `std::string id`
ID.
- `std::string fullId`
Full ID of the class in URI format.
- `std::string label`
- `std::string type`
- `std::set< std::string > synonyms`
- `std::set< std::string > definitions`
- `std::vector< relationStruct > relations`

9.12.1 Detailed Description

This class is for taking care of the specific structure of response data call classBean.

This class is normally used for parsing the response of term command.

9.12.2 Member Typedef Documentation

9.12.2.1 unisys::ClassBean::relationStruct

For collecting the data in the relations node.

9.12.3 Constructor & Destructor Documentation

9.12.3.1 `unisys::ClassBean::ClassBean()`

Default constructor.

9.12.3.2 `unisys::ClassBean::ClassBean(XMLNode xml)`

Overload constructor for parsing XML response.

9.12.4 Member Function Documentation

9.12.4.1 `std::set<std::string> unisys::ClassBean::getDefinitions() const`

Return the list of definition.

9.12.4.2 `std::string unisys::ClassBean::getFullId() const`

Return fulld.

9.12.4.3 `std::string unisys::ClassBean::getId() const`

Return id.

9.12.4.4 `std::string unisys::ClassBean::getLabel() const`

Return label.

9.12.4.5 `std::vector<relationStruct> unisys::ClassBean::getRelations() const`

Return the list of relation data structure.

9.12.4.6 `std::set<std::string> unisys::ClassBean::getSynonyms() const`

Return the list of synonym.

9.12.4.7 `std::string unisys::ClassBean::getType() const`

Returntype.

9.12.5 Member Data Documentation

9.12.5.1 `std::set<std::string> unisys::ClassBean::definitions [private]`

9.12.5.2 `std::string unisys::ClassBean::fullId [private]`

Full ID of the class in URI format.

9.12.5.3 std::string unisys::ClassBean::id [private]

ID.

9.12.5.4 std::string unisys::ClassBean::label [private]

9.12.5.5 std::vector<relationStruct> unisys::ClassBean::relations [private]

9.12.5.6 std::set<std::string> unisys::ClassBean::synonyms [private]

9.12.5.7 std::string unisys::ClassBean::type [private]

The documentation for this class was generated from the following file:

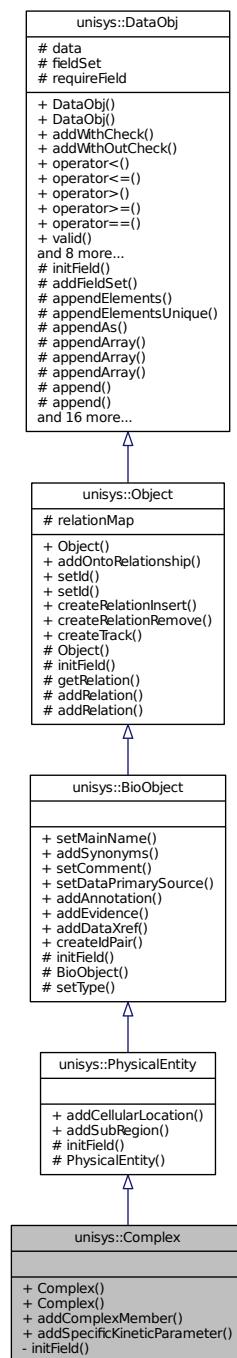
- [restBioPortal.h](#)

9.13 unisys::Complex Class Reference

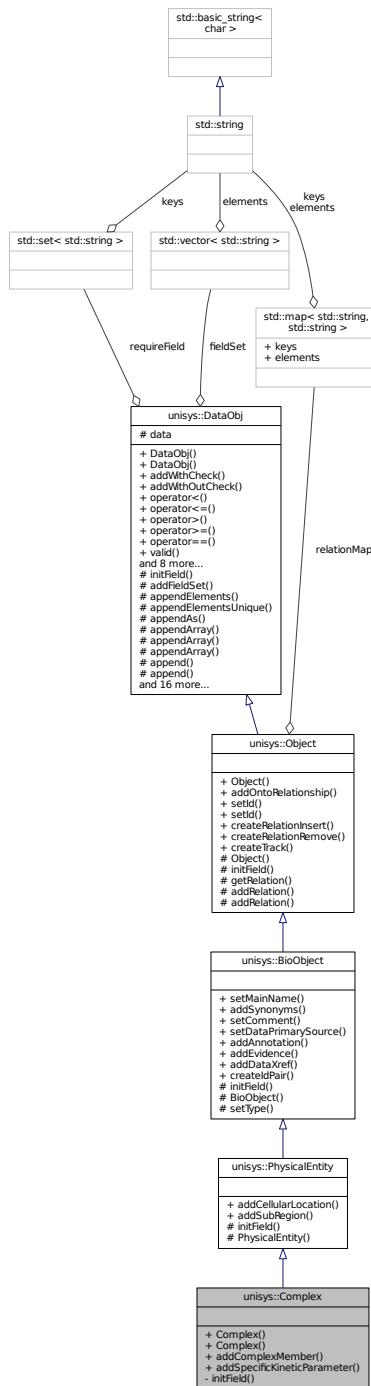
This class is for miriam cross reference annotation.

```
#include <ObjClass.h>
```

Inheritance diagram for unisys::Complex:



Collaboration diagram for unisys::Complex:



Public Member Functions

- [Complex \(\)](#)
Default constructor.
- [Complex \(mongo::BSONObj const &bsonObj\)](#)
Overloaded constructor is used when retrieving data in bson object from database and transform to C++ object.
- void [addComplexMember \(PEIdRef &peldRef\)](#)

- void [addSpecificKineticParameter \(KineticParameter &kineticParameter\)](#)

Private Member Functions

- void [initField \(\)](#)

Additional Inherited Members

9.13.1 Detailed Description

This class is for miriam cross reference annotation.

```
 BSON structure:
{
    _id: <string>, #mandatory
    ontologyRelationship: {<RelationshipBOSON>, <RelationshipBOSON>, ...},
    name: {<string>, <string>, ...}
    comment: <string>,
    dataPrimarySource: <XrefBOSON>,
    functionAnnotation: {<AnnotationBOSON>, <AnnotationBOSON>, ...},
    evidence: {<EvidenceBOSON>, <EvidenceBOSON>, ...},
    dataxref: {<XrefBOSON>, <XrefBOSON>, ...},
    interaction: {<DbRef>, <DbRef>, ...},
    complex: {<DbRef>, <DbRef>, ...},
    subRegion: {<SubRegionBOSON>, <SubRegionBOSON>, ...},
    complexMember: {<DbRef>, <DbRef>, ...},
    specificKineticParameter: {<KineticParameterBOSON>, <KineticParameterBOSON>, ...}
}
```

9.13.2 Constructor & Destructor Documentation

9.13.2.1 unisys::Complex::Complex ()

Default constructor.

9.13.2.2 unisys::Complex::Complex (mongo::BSONObj const & bsonObj)

Overloaded constructor is used when retrieving data in boson object from database and transform to C++ object.

9.13.3 Member Function Documentation

9.13.3.1 void unisys::Complex::addComplexMember (PEIdRef & peldRef)

9.13.3.2 void unisys::Complex::addSpecificKineticParameter (KineticParameter & kineticParameter)

9.13.3.3 void unisys::Complex::initField () [private], [virtual]

Reimplemented from [unisys::PhysicalEntity](#).

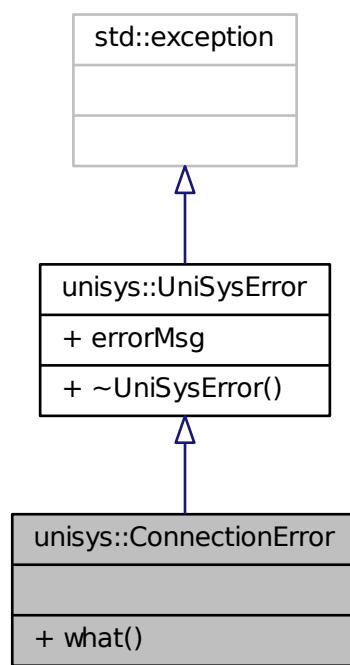
The documentation for this class was generated from the following file:

- [ObjClass.h](#)

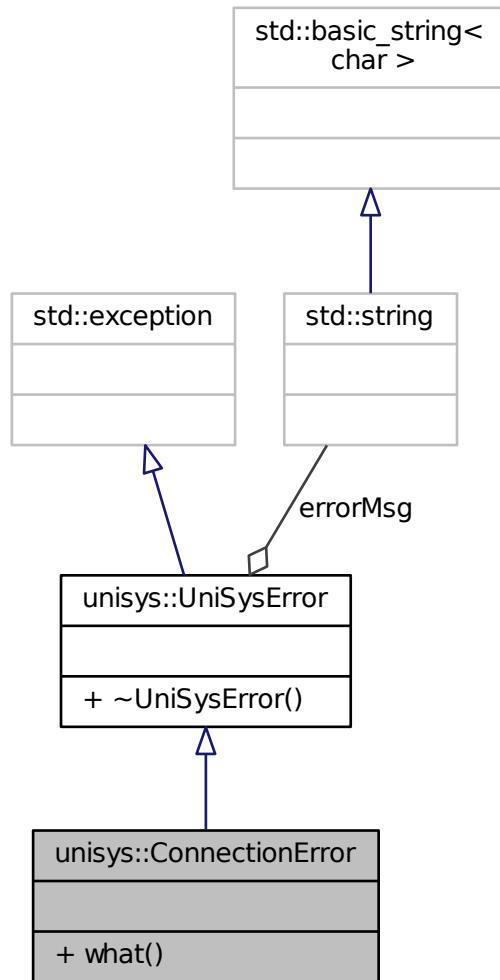
9.14 unisys::ConnectionError Class Reference

```
#include <exception.h>
```

Inheritance diagram for unisys::ConnectionError:



Collaboration diagram for unisys::ConnectionError:



Public Member Functions

- `virtual const char * what () const throw ()`

Additional Inherited Members

9.14.1 Member Function Documentation

9.14.1.1 `virtual const char* unisys::ConnectionError::what () const throw () [inline], [virtual]`

The documentation for this class was generated from the following file:

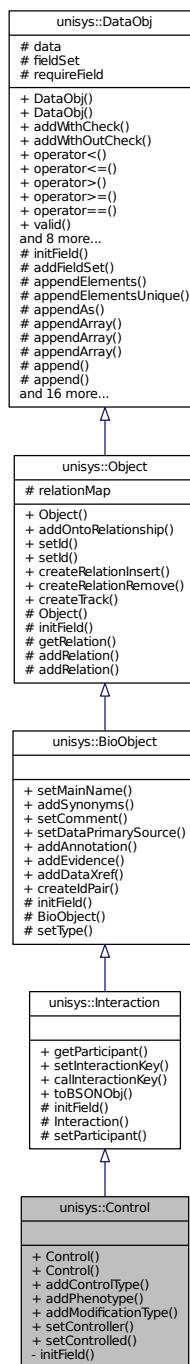
- `exception.h`

9.15 unisys::Control Class Reference

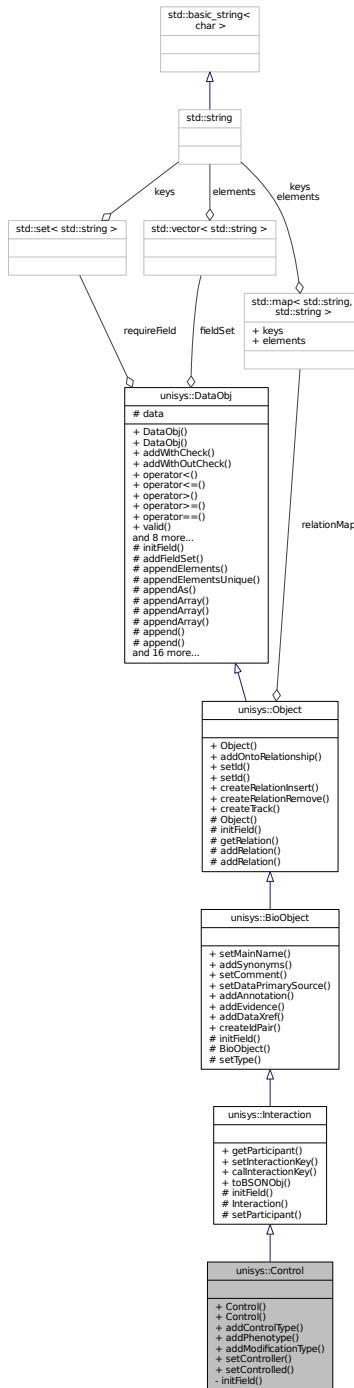
This class is for miriam cross reference annotation.

```
#include <ObjClass.h>
```

Inheritance diagram for unisys::Control:



Collaboration diagram for unisys::Control:



Public Member Functions

- Control ()

Default constructor.

- [Control](#) (`mongo::BSONObj const &bsonObj`)

Overloaded constructor is used when retrieving data in boson object from database and transform to C++ object.

- void addControlType (OntoldRef &ontoldRef)

- void [addPhenotype \(OntoldRef &ontoldRef\)](#)
- void [addModificationType \(OntoldRef &ontoldRef\)](#)
- void [setController \(PEIdRef &peldRef\)](#)
- void [setControlled \(PEIdRef &peldRef\)](#)

Private Member Functions

- void [initField \(\)](#)

Additional Inherited Members

9.15.1 Detailed Description

This class is for miriam cross reference annotation.

```
BSON structure:
{
    _id: <string>, #madatory
    type: <string>,
    ontologyRelationship: {<RelationshipBOSON>, <RelationshipBOSON>, ...},
    name: {<string>, <string>, ...}
    comment: <string>,
    dataPrimarySource: <XrefBOSON>,
    functionAnnotation: {<AnnotationBOSON>, <AnnotationBOSON>, ...},
    evidence: {<EvidenceBOSON>, <EvidenceBOSON>, ...},
    dataxref: {<XrefBOSON>, <XrefBOSON>, ...},
    participant: {<StoichiometryBOSON>, <StoichiometryBOSON>, ...},
    interactionKey: <string>,
    controlType: {<DbRef>, <DbRef>, ...},
    phenotype: {<DbRef>, <DbRef>, ...},
    modificationType: {<DbRef>, <DbRef>, ...}
}
```

9.15.2 Constructor & Destructor Documentation

9.15.2.1 unisys::Control::Control ()

Default constructor.

9.15.2.2 unisys::Control::Control (mongo::BSONObj const & *bsonObj*)

Overloaded constructor is used when retrieving data in boson object from database and transform to C++ object.

9.15.3 Member Function Documentation

9.15.3.1 void unisys::Control::addControlType (OntoldRef & *ontoldRef*)

9.15.3.2 void unisys::Control::addModificationType (OntoldRef & *ontoldRef*)

9.15.3.3 void unisys::Control::addPhenotype (OntoldRef & *ontoldRef*)

9.15.3.4 void unisys::Control::initField () [private], [virtual]

Reimplemented from [unisys::Interaction](#).

9.15.3.5 void unisys::Control::setControlled (**PEIdRef & peldRef**)

9.15.3.6 void unisys::Control::setController (**PEIdRef & peldRef**)

The documentation for this class was generated from the following file:

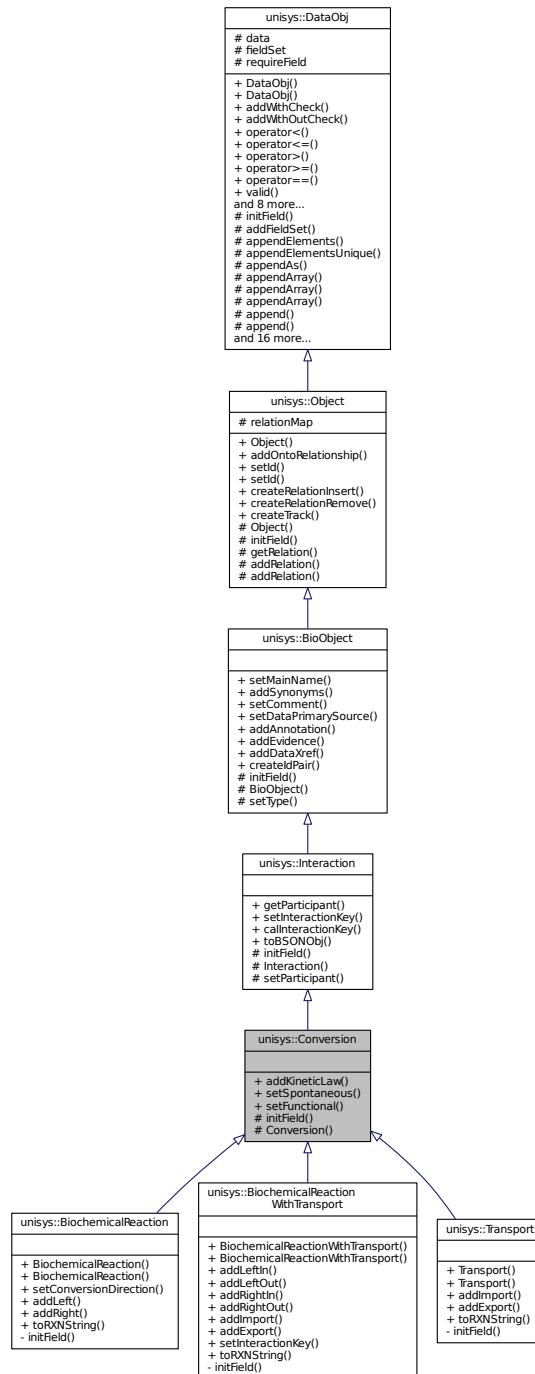
- [ObjClass.h](#)

9.16 unisys::Conversion Class Reference

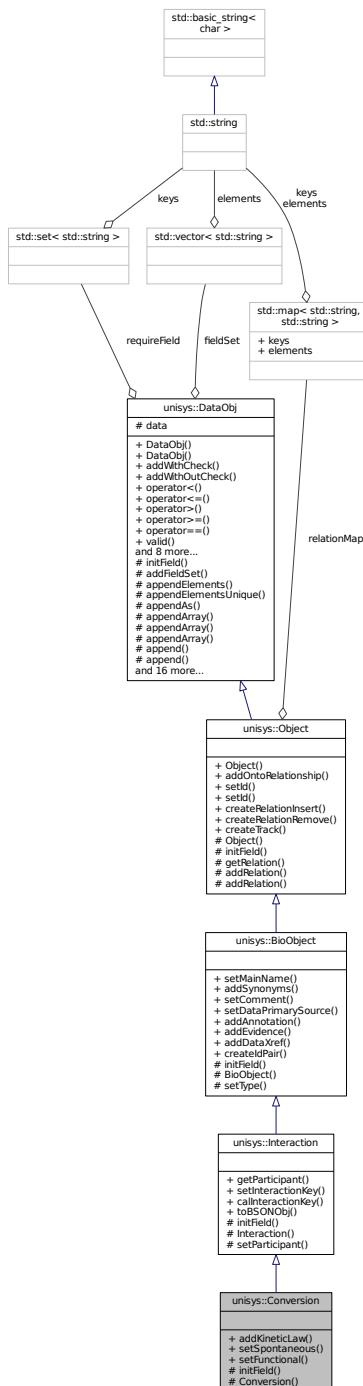
This class is for miriam cross reference annotation.

```
#include <ObjClass.h>
```

Inheritance diagram for unisys::Conversion:



Collaboration diagram for unisys::Conversion:



Public Member Functions

- void `addKineticLaw` (`MathML &kineticLaw`)
- void `setSpontaneous` (bool value=true)
- void `setFunctional` (bool value=true)

Protected Member Functions

- void [initField \(\)](#)
- [Conversion \(\)](#)

9.16.1 Detailed Description

This class is for miriam cross reference annotation.

```
 BSON structure:
{
    _id: <string>, #madatory
    type: <string>,
    ontologyRelationship: {<RelationshipBOSON>, <RelationshipBOSON>, ...},
    name: {<string>,<string>, ...}
    comment: <string>,
    dataPrimarySource: <XrefBOSON>,
    functionAnnotation: {<AnnotationBOSON>, <AnnotationBOSON>, ...},
    evidence: {<EvidenceBOSON>, <EvidenceBOSON>, ...},
    dataxref: {<XrefBOSON>, <XrefBOSON>, ...},
    participant: {<StoichiometryBOSON>, <StoichiometryBOSON>, ...},
    interactionKey: <string>,
    conversionDirection: <DbRef>,
    kineticLaw: {<MathMLBOSON>, <MathMLBOSON>, ...}
    spontaneous: <string>,
    functional: <string>
}
```

9.16.2 Constructor & Destructor Documentation

9.16.2.1 [unisys::Conversion::Conversion \(\)](#) [protected]

9.16.3 Member Function Documentation

9.16.3.1 [void unisys::Conversion::addKineticLaw \(MathML & kineticLaw \)](#)

9.16.3.2 [void unisys::Conversion::initField \(\)](#) [protected], [virtual]

Reimplemented from [unisys::Interaction](#).

Reimplemented in [unisys::BiochemicalReactionWithTransport](#), [unisys::Transport](#), and [unisys::BiochemicalReaction](#).

9.16.3.3 [void unisys::Conversion::setFunctional \(bool value = true \)](#)

9.16.3.4 [void unisys::Conversion::setSpontaneous \(bool value = true \)](#)

The documentation for this class was generated from the following file:

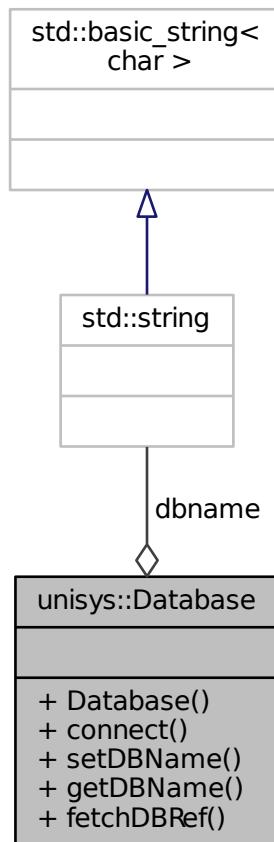
- [ObjClass.h](#)

9.17 unisys::Database Class Reference

This class is a database handle class.

```
#include <database.h>
```

Collaboration diagram for unisys::Database:



Public Member Functions

- `Database (std::string const &dbname="test")`
Overloaded constructor with specific database name.
- `void connect (std::string const &serverHostname)`
Overloaded interface of connection function.
- `void setDBName (std::string const &dbname)`
Set or change database name.
- `std::string getDBName ()`
Return database name.
- `mongo::BSONObj fetchDBRef (mongo::BSONObj const &bsonObj) throw (QueryError)`
Dereference boson database reference (DBRef)

Private Attributes

- `std::string dbname`
Database name, default is test.

Friends

- class [Updater](#)

9.17.1 Detailed Description

This class is a database handle class.

[Database](#) class is derived from mongo::DBClientConnection, used to handle the connection between client and database. In this derived class, there are some functions that specific to UniSysDB database data structure.

9.17.2 Constructor & Destructor Documentation

9.17.2.1 `unisys::Database::Database (std::string const & dbname = "test")`

Overloaded constructor with specific database name.

9.17.3 Member Function Documentation

9.17.3.1 `void unisys::Database::connect (std::string const & serverHostname)`

Overloaded interface of connection function.

9.17.3.2 `mongo::BSONObj unisys::Database::fetchDBRef (mongo::BSONObj const & bsonObj) throw (QueryError)`

Dereference boson database reference (DBRef)

This function retrieves bson database reference, which format is `{"$ref": "collectionNS", "$id": "id"[, "$db": "databaseName"]}`, as an input and searches for the document that has `_id` as described in DBRef from specified collection namespace.

Parameters

<code>bsonObj</code>	boson object of DBRef
----------------------	-----------------------

9.17.3.3 `std::string unisys::Database::getDBName ()`

Return database name.

9.17.3.4 `void unisys::Database::setDBName (std::string const & dbname)`

Set or change database name.

9.17.4 Friends And Related Function Documentation

9.17.4.1 `friend class Updater [friend]`

9.17.5 Member Data Documentation

9.17.5.1 `std::string unisys::Database::dbname [private]`

[Database](#) name, default is test.

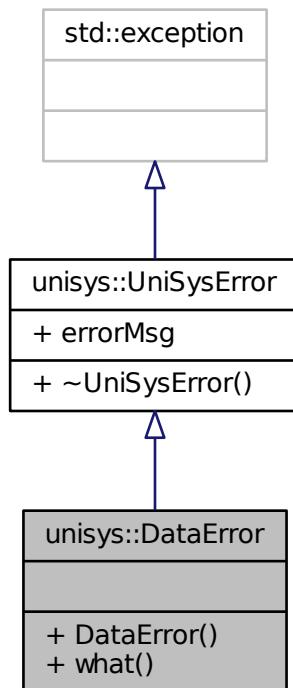
The documentation for this class was generated from the following file:

- [database.h](#)

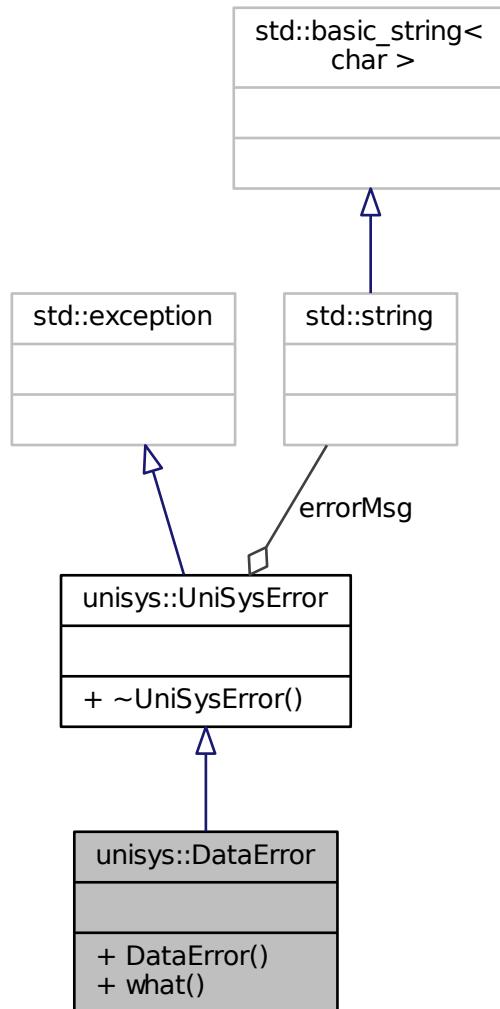
9.18 unisys::DataError Class Reference

```
#include <exception.h>
```

Inheritance diagram for unisys::DataError:



Collaboration diagram for unisys::DataError:



Public Member Functions

- [DataError \(std::string const &str\)](#)
- virtual const char * [what \(\) const throw \(\)](#)

Additional Inherited Members

9.18.1 Constructor & Destructor Documentation

[9.18.1.1 unisys::DataError::DataError \(std::string const & str \) \[inline\]](#)

9.18.2 Member Function Documentation

9.18.2.1 virtual const char* unisys::DataError::what() const throw() [inline], [virtual]

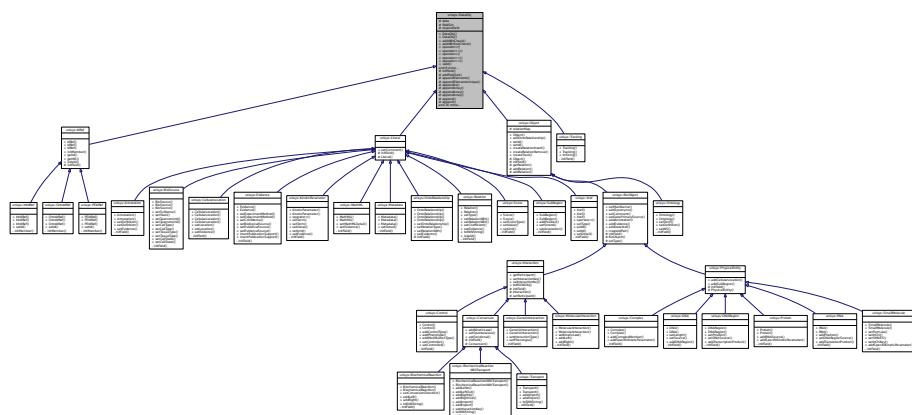
The documentation for this class was generated from the following file:

- [exception.h](#)

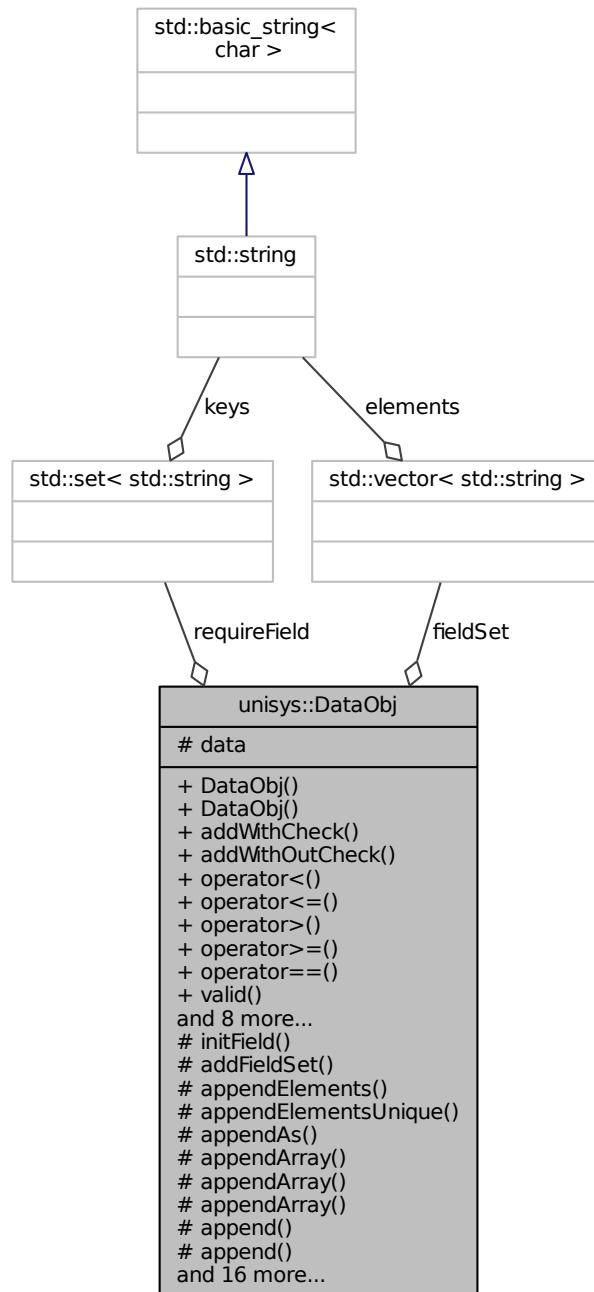
9.19 unisys::DataObj Class Reference

```
#include <DBClass.h>
```

Inheritance diagram for unisys::DataObj:



Collaboration diagram for unisys::DataObj:



Public Member Functions

- `DataObj()`
- `DataObj(mongo::BSONObj const &bsonObj)`
- `void addWithCheck(mongo::BSONObj const &bsonObj)`
- `void addWithOutCheck(mongo::BSONObj const &bsonObj)`
- `bool operator<(DataObj const &other) const`

- bool `operator<=` (`DataObj` const &other) const
- bool `operator>` (`DataObj` const &other) const
- bool `operator>=` (`DataObj` const &other) const
- bool `operator==` (`DataObj` const &other) const
- bool `valid` () const
- std::string `md5` ()
- mongo::BSONObj `toBSONObj` ()
- std::string `toString` (bool isArray=false, bool full=false)
- mongo::BSONElement `getField` (std::string const &fieldName) const
- bool `hasField` (std::string const &fieldName) const
- void `removeField` (std::string const &fieldName)
- bool `isOwned` () const
- bool `isValid` () const

Protected Member Functions

- virtual void `initField` ()=0
- void `addFieldSet` (std::string const &`fieldSet`, bool isRequire=false)
- void `appendElements` (mongo::BSONObj x)
- void `appendElementsUnique` (mongo::BSONObj x)
- void `appendAs` (mongo::BSONElement const &e, std::string const &fieldName, bool substitute=false)
- void `appendArray` (std::string const &fieldName, mongo::BSONObj const &subObj)
- void `appendArray` (std::string const &fieldName, int start, int stop)
- template<class type>
 - void `appendArray` (std::string const &fieldName, type const &subObj)
- void `append` (std::string const &fieldName, const char *str, int sz, bool substitute=false)
- template<class myType>
 - void `append` (std::string const &fieldName, myType value, bool substitute=false)
- void `set` (std::string const &fieldName, std::string const &value)
- template<class myType>
 - void `set` (std::string const &fieldName, myType value)
- void `set` (std::string const &fieldName, int value)
- void `set` (std::string const &fieldName, double value)
- void `set` (std::string const &fieldName, const char *value)
- void `set` (std::string const &fieldName, bool value)
- void `set` (std::string const &fieldName, int start, int stop)
- void `appendTimeT` (std::string const &fieldName, time_t dt, bool substitute=false)
- void `appendRegex` (std::string const &fieldName, std::string const ®ex, std::string const &options="", bool substitute=false)
- void `appendTimeStamp` (std::string const &fieldName, bool substitute=false)
- void `appendAsNumber` (std::string const &fieldName, std::string const &`data`, bool substitute=false)
- void `appendNull` (std::string const &fieldName, bool substitute=false)
- void `appendCode` (std::string const &fieldName, std::string const &code, bool substitute=false)
- void `appendDBRef` (std::string const &fieldName, std::string const &ns, mongo::OID const &oid, bool substitute=false)
- void `genOID` ()
- void `sort` ()

Protected Attributes

- mongo::BSONObj `data`
- std::vector< std::string > `fieldSet`
- std::set< std::string > `requireField`

9.19.1 Constructor & Destructor Documentation

9.19.1.1 `unisys::DataObj::DataObj()`

9.19.1.2 `unisys::DataObj::DataObj(mongo::BSONObj const & bsonObj)`

9.19.2 Member Function Documentation

9.19.2.1 `void unisys::DataObj::addFieldSet(std::string const & fieldSet, bool isRequire = false) [inline], [protected]`

9.19.2.2 `void unisys::DataObj::addWithCheck(mongo::BSONObj const & bsonObj)`

9.19.2.3 `void unisys::DataObj::addWithOutCheck(mongo::BSONObj const & bsonObj)`

9.19.2.4 `void unisys::DataObj::append(std::string const & fieldName, const char * str, int sz, bool substitute = false) [protected]`

9.19.2.5 `template<class myType> void unisys::DataObj::append(std::string const & fieldName, myType value, bool substitute = false) [inline], [protected]`

9.19.2.6 `void unisys::DataObj::appendArray(std::string const & fieldName, mongo::BSONObj const & subObj) [protected]`

9.19.2.7 `void unisys::DataObj::appendArray(std::string const & fieldName, int start, int stop) [protected]`

9.19.2.8 `template<class type> void unisys::DataObj::appendArray(std::string const & fieldName, type const & subObj) [inline], [protected]`

9.19.2.9 `void unisys::DataObj::appendAs(mongo::BSONElement const & e, std::string const & fieldName, bool substitute = false) [protected]`

9.19.2.10 `void unisys::DataObj::appendAsNumber(std::string const & fieldName, std::string const & data, bool substitute = false) [protected]`

9.19.2.11 `void unisys::DataObj::appendCode(std::string const & fieldName, std::string const & code, bool substitute = false) [protected]`

9.19.2.12 `void unisys::DataObj::appendDBRef(std::string const & fieldName, std::string const & ns, mongo::OID const & oid, bool substitute = false) [protected]`

9.19.2.13 `void unisys::DataObj::appendElements(mongo::BSONObj x) [protected]`

9.19.2.14 `void unisys::DataObj::appendElementsUnique(mongo::BSONObj x) [protected]`

9.19.2.15 `void unisys::DataObj::appendNull(std::string const & fieldName, bool substitute = false) [protected]`

9.19.2.16 `void unisys::DataObj::appendRegex(std::string const & fieldName, std::string const & regex, std::string const & options = "", bool substitute = false) [protected]`

9.19.2.17 `void unisys::DataObj::appendTimeStamp(std::string const & fieldName, bool substitute = false) [protected]`

9.19.2.18 `void unisys::DataObj::appendTimeT(std::string const & fieldName, time_t dt, bool substitute = false) [protected]`

```

9.19.2.19 void unisys::DataObj::genOID( ) [protected]

9.19.2.20 mongo::BSONElement unisys::DataObj::getField( std::string const & fieldName ) const

9.19.2.21 bool unisys::DataObj::hasField( std::string const & fieldName ) const

9.19.2.22 virtual void unisys::DataObj::initField( ) [protected], [pure virtual]

```

Implemented in [unisys::BiochemicalReactionWithTransport](#), [unisys::Transport](#), [unisys::BiochemicalReaction](#), [unisys::Conversion](#), [unisys::GeneticInteraction](#), [unisys::MolecularInteraction](#), [unisys::Control](#), [unisys::Interaction](#), [unisys::Metadata](#), [unisys::Complex](#), [unisys::SubRegion](#), [unisys::Protein](#), [unisys::KineticParameter](#), [unisys::Tracking](#), [unisys::MathML](#), [unisys::RNA](#), [unisys::CellularLocation](#), [unisys::DNARegion](#), [unisys::Relation](#), [unisys::IdRef](#), [unisys::DNA](#), [unisys::OntoRelationship](#), [unisys::Annotation](#), [unisys::SmallMolecule](#), [unisys::Evidence](#), [unisys::PhysicalEntity](#), [unisys::BioSource](#), [unisys::BioObject](#), [unisys::Score](#), [unisys::Ontology](#), [unisys::Xref](#), [unisys::Object](#), and [unisys::Literal](#).

```
9.19.2.23 bool unisys::DataObj::isOwned( ) const
```

```
9.19.2.24 bool unisys::DataObj::isValid( ) const
```

Reimplemented in [unisys::IdRef](#).

```
9.19.2.25 std::string unisys::DataObj::md5( )
```

```
9.19.2.26 bool unisys::DataObj::operator<( DataObj const & other ) const
```

```
9.19.2.27 bool unisys::DataObj::operator<=( DataObj const & other ) const
```

```
9.19.2.28 bool unisys::DataObj::operator==( DataObj const & other ) const
```

```
9.19.2.29 bool unisys::DataObj::operator>( DataObj const & other ) const
```

```
9.19.2.30 bool unisys::DataObj::operator>=( DataObj const & other ) const
```

```
9.19.2.31 void unisys::DataObj::removeField( std::string const & fieldName )
```

```
9.19.2.32 void unisys::DataObj::set( std::string const & fieldName, std::string const & value ) [protected]
```

```
9.19.2.33 template<class myType> void unisys::DataObj::set( std::string const & fieldName, myType value ) [inline], [protected]
```

```
9.19.2.34 void unisys::DataObj::set( std::string const & fieldName, int value ) [protected]
```

```
9.19.2.35 void unisys::DataObj::set( std::string const & fieldName, double value ) [protected]
```

```
9.19.2.36 void unisys::DataObj::set( std::string const & fieldName, const char * value ) [protected]
```

```
9.19.2.37 void unisys::DataObj::set( std::string const & fieldName, bool value ) [protected]
```

```
9.19.2.38 void unisys::DataObj::set( std::string const & fieldName, int start, int stop ) [protected]
```

```
9.19.2.39 void unisys::DataObj::sort( ) [protected]
```

```
9.19.2.40 mongo::BSONObj unisys::DataObj::toBSONObj( )
```

Reimplemented in [unisys::Interaction](#).

9.19.2.41 `std::string unisys::DataObj::toString (bool isArray = false, bool full = false)`

9.19.2.42 `bool unisys::DataObj::valid () const`

9.19.3 Member Data Documentation

9.19.3.1 `mongo::BSNOBJ unisys::DataObj::data [protected]`

9.19.3.2 `std::vector<std::string> unisys::DataObj::fieldSet [protected]`

9.19.3.3 `std::set<std::string> unisys::DataObj::requireField [protected]`

The documentation for this class was generated from the following file:

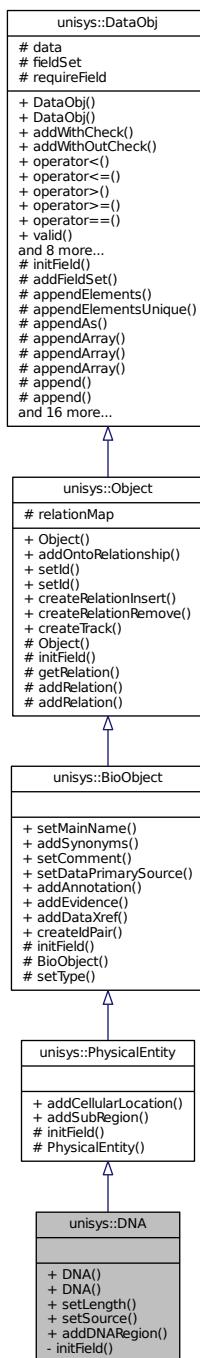
- [DBClass.h](#)

9.20 unisys::DNA Class Reference

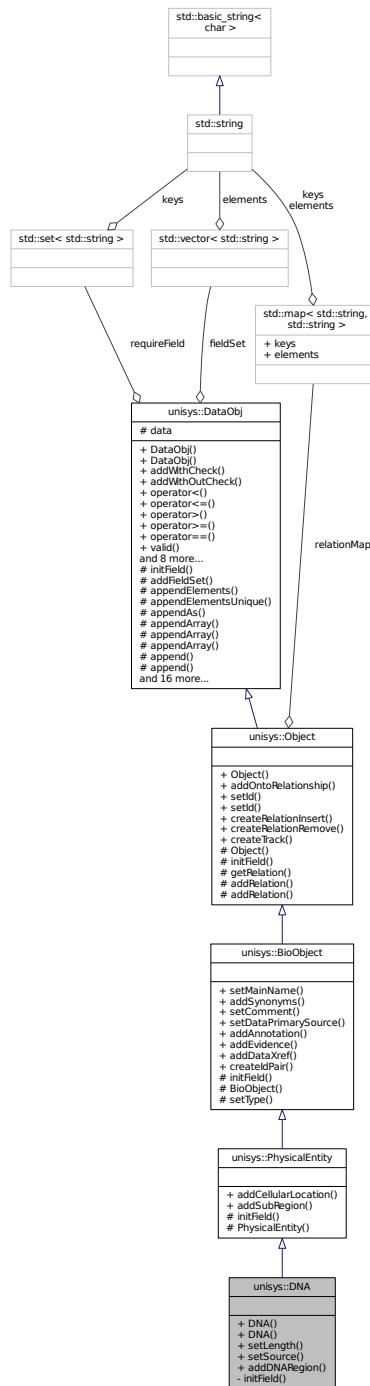
This class is for miriam cross reference annotation.

```
#include <ObjClass.h>
```

Inheritance diagram for unisys::DNA:



Collaboration diagram for unisys::DNA:



Public Member Functions

- [DNA \(\)](#)

Default constructor.

- [DNA \(mongo::BSONObj const &bsonObj\)](#)

Overloaded constructor is used when retrieving data in bson object from database and transform to C++ object.

- void [setLength](#) (int length)

- void [setSource \(BioSource &source\)](#)
- void [addDNARegion \(PEIdRef &dnaRegion\)](#)

Private Member Functions

- void [initField \(\)](#)

Additional Inherited Members

9.20.1 Detailed Description

This class is for miriam cross reference annotation.

```
 BSON structure:
{
    _id: <string>, #madatory
    ontologyRelationship: {<RelationshipBOSON>, <RelationshipBOSON>, ...},
    name: {<string>, <string>, ...}
    comment: <string>,
    dataPrimarySource: <XrefBOSON>,
    functionAnnotation: {<AnnotationBOSON>, <AnnotationBOSON>, ...},
    evidence: {<EvidenceBOSON>, <EvidenceBOSON>, ...},
    dataxref: {<XrefBOSON>, <XrefBOSON>, ...},
    interaction: {<DbRef>, <DbRef>, ...},
    complex: {<DbRef>, <DbRef>, ...},
    subRegion: {<SubRegionBOSON>, <SubRegionBOSON>, ...},
    length: <int>,
    source: <BioSourceBOSON>,
    dnaRegion: {<DbRef>, <DbRef>, ...}
}
```

9.20.2 Constructor & Destructor Documentation

9.20.2.1 unisys::DNA::DNA ()

Default constructor.

9.20.2.2 unisys::DNA::DNA (mongo::BSONObj const & *bsonObj*)

Overloaded constructor is used when retrieving data in boson object from database and tranform to C++ object.

9.20.3 Member Function Documentation

9.20.3.1 void unisys::DNA::addDNARegion (PEIdRef & *dnaRegion*)

9.20.3.2 void unisys::DNA::initField () [private], [virtual]

Reimplemented from [unisys::PhysicalEntity](#).

9.20.3.3 void unisys::DNA::setLength (int *length*)

9.20.3.4 void unisys::DNA::setSource (BioSource & *source*)

The documentation for this class was generated from the following file:

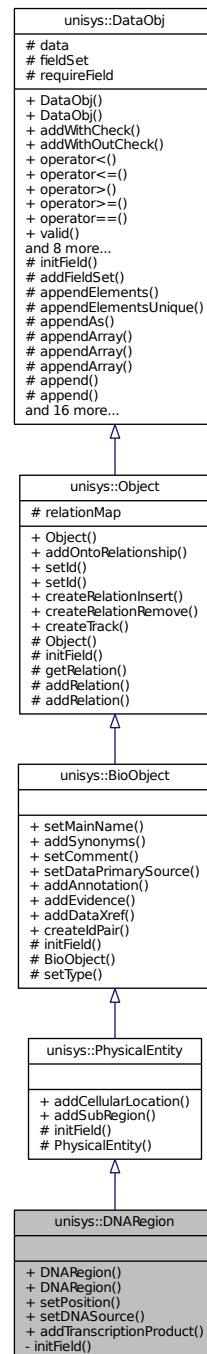
- [ObjClass.h](#)

9.21 unisys::DNARegion Class Reference

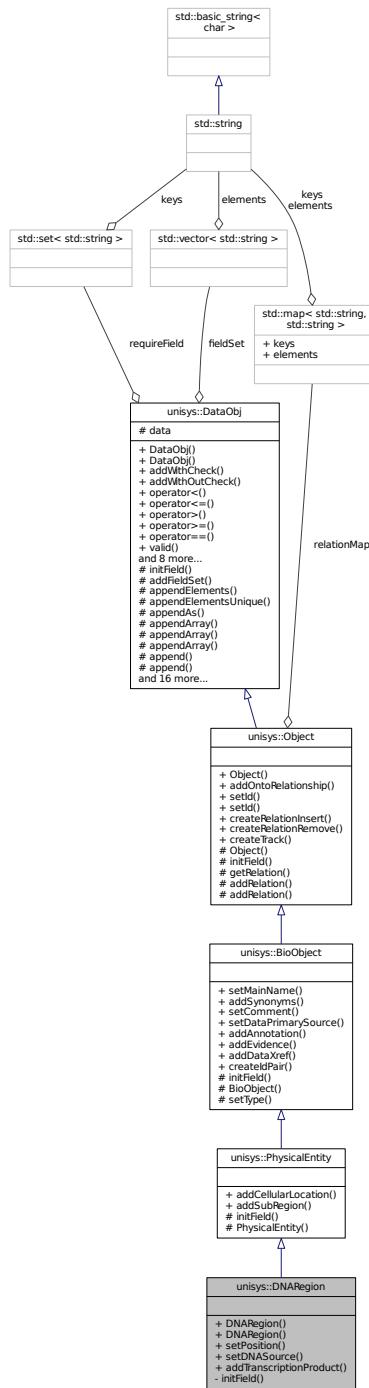
This class is for miriam cross reference annotation.

```
#include <ObjClass.h>
```

Inheritance diagram for unisys::DNARegion:



Collaboration diagram for unisys::DNARegion:



Public Member Functions

- [DNARegion \(\)](#)
Default constructor.
- [DNARegion \(mongo::BSONObj const &bsonObj\)](#)
Overloaded constructor is used when retrieving data in bson object from database and transform to C++ object.
- [void setPosition \(unsigned int start, unsigned int stop\)](#)

- void [setDNASource \(PEIdRef &peldRef \)](#)
- void [addTranscriptionProduct \(PEIdRef &trans \)](#)

Private Member Functions

- void [initField \(\)](#)

Additional Inherited Members

9.21.1 Detailed Description

This class is for miriam cross reference annotation.

```
 BSON structure:
{
    _id: <string>, #madatory
    ontologyRelationship: {<RelationshipBOSON>, <RelationshipBOSON>, ...},
    name: {<string>, <string>, ...}
    comment: <string>,
    dataPrimarySource: <XrefBOSON>,
    functionAnnotation: {<AnnotationBOSON>, <AnnotationBOSON>, ...},
    evidence: {<EvidenceBOSON>, <EvidenceBOSON>, ...},
    dataxref: {<XrefBOSON>, <XrefBOSON>, ...},
    interaction: {<DbRef>, <DbRef>, ...},
    complex: {<DbRef>, <DbRef>, ...},
    subRegion: {<SubRegionBOSON>, <SubRegionBOSON>, ...}
    position: <positionPairBSON>,
    dnaSource: <DbRef>,
    transcriptionProduct: {<DbRef>, <DbRef>, ...}
}
```

9.21.2 Constructor & Destructor Documentation

9.21.2.1 unisys::DNARegion::DNARegion ()

Default constructor.

9.21.2.2 unisys::DNARegion::DNARegion (mongo::BSONObj const & *bsonObj*)

Overloaded constructor is used when retrieving data in boson object from database and transform to C++ object.

9.21.3 Member Function Documentation

9.21.3.1 void unisys::DNARegion::addTranscriptionProduct (PEIdRef & *trans*)

9.21.3.2 void unisys::DNARegion::initField () [private], [virtual]

Reimplemented from [unisys::PhysicalEntity](#).

9.21.3.3 void unisys::DNARegion::setDNASource (PEIdRef & *peldRef*)

9.21.3.4 void unisys::DNARegion::setPosition (unsigned int *start*, unsigned int *stop*)

The documentation for this class was generated from the following file:

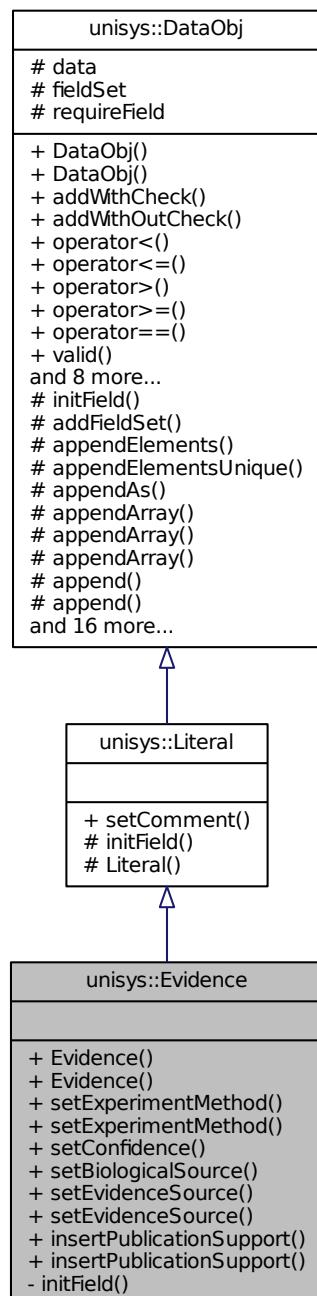
- [ObjClass.h](#)

9.22 unisys::Evidence Class Reference

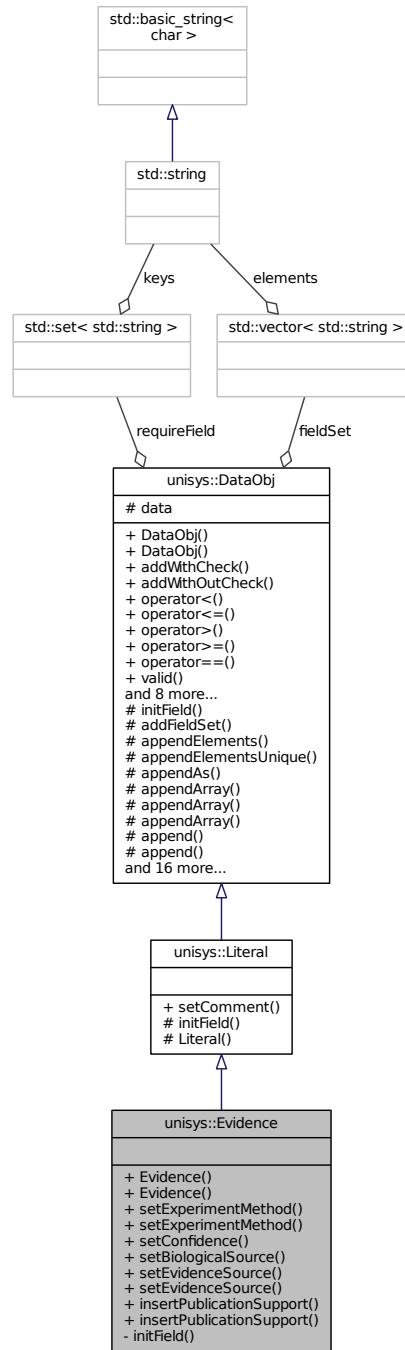
The C++ representative class for evidence data class.

```
#include <LitClass.h>
```

Inheritance diagram for unisys::Evidence:



Collaboration diagram for unisys::Evidence:



Public Member Functions

- [Evidence \(\)](#)
Default constructor.
- [Evidence \(mongo::BSONObj const &bsonObj\)](#)
Overloaded constructor is used when retrieving data in bson object from database and transform to C++ object.
- [void setExperimentMethod \(OntoldRef &ontoldRef\)](#)

- void `setExperimentMethod` (std::string const &ontold)
- void `setConfidence` (Score &score)
- void `setBiologicalSource` (BioSource &biologicalSource)
- void `setEvidenceSource` (Xref &evidenceSource)
- void `setEvidenceSource` (std::string const &evidenceSourceId, std::string const &detail="")
- void `insertPublicationSupport` (Xref &publicationSupport)
- void `insertPublicationSupport` (std::string const &publicationSupportId, std::string const &detail="")

Private Member Functions

- void `initField` ()

function for init field in the object

Additional Inherited Members

9.22.1 Detailed Description

The C++ representative class for evidence data class.

```
BSON structure:
{
    comment: <string>,
    experimentMethod: {$ref: <collname>, $id: <idvalue>}, #format by idref class
    score: <ScoreBSONStructure>,
    biologicalSource: <BioSourceBSONStructure>,
    evidenceSource: <XRefBSONStructure>,
    publicationSupport: {<XRefBSONStructure>, <XRefBSONStructure>, ...}
}
```

9.22.2 Constructor & Destructor Documentation

9.22.2.1 unisys::Evidence::Evidence ()

Default constructor.

9.22.2.2 unisys::Evidence::Evidence (mongo::BSONObj const & bsonObj)

Overloaded constructor is used when retrieving data in boson object from database and tranform to C++ object.

9.22.3 Member Function Documentation

9.22.3.1 void unisys::Evidence::initField () [private], [virtual]

function for init field in the object

Reimplemented from [unisys::Literal](#).

9.22.3.2 void unisys::Evidence::insertPublicationSupport (Xref & publicationSupport)

9.22.3.3 void unisys::Evidence::insertPublicationSupport (std::string const & publicationSupportId, std::string const & detail = " ")

9.22.3.4 void unisys::Evidence::setBiologicalSource (BioSource & biologicalSource)

9.22.3.5 void unisys::Evidence::setConfidence (*Score & score*)

9.22.3.6 void unisys::Evidence::setEvidenceSource (*Xref & evidenceSource*)

9.22.3.7 void unisys::Evidence::setEvidenceSource (std::string const & *evidenceSourceId*, std::string const & *detail* = " ")

9.22.3.8 void unisys::Evidence::setExperimentMethod (*OntoldRef & ontoldRef*)

9.22.3.9 void unisys::Evidence::setExperimentMethod (std::string const & *ontold*)

The documentation for this class was generated from the following file:

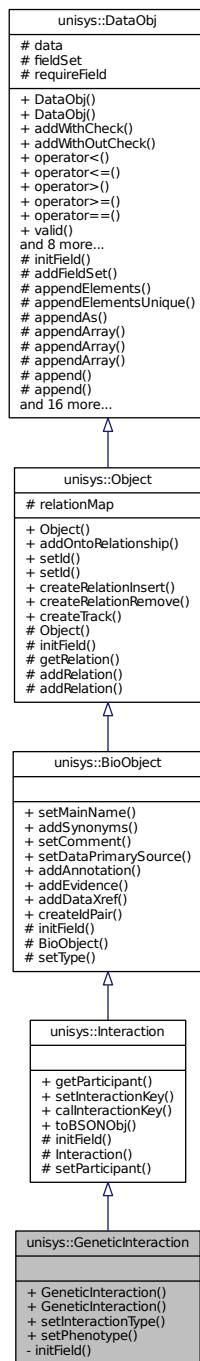
- [LitClass.h](#)

9.23 unisys::GeneticInteraction Class Reference

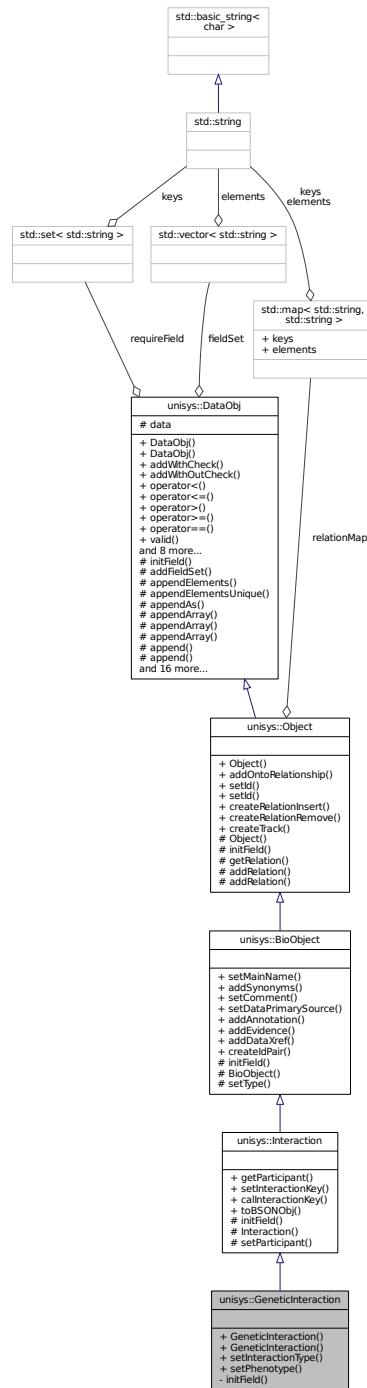
This class is for miriam cross reference annotation.

```
#include <ObjClass.h>
```

Inheritance diagram for unisys::GeneticInteraction:



Collaboration diagram for unisys::GeneticInteraction:



Public Member Functions

- **GeneticInteraction ()**
Default constructor.
- **GeneticInteraction (mongo::BSONObj const &bsonObj)**
Overloaded constructor is used when retrieving data in bson object from database and transform to C++ object.
- void **setInteractionType (OntoldRef &ontoldRef)**

- void [setPhenotype \(OntoldRef &ontoldRef\)](#)

Private Member Functions

- void [initField \(\)](#)

Additional Inherited Members

9.23.1 Detailed Description

This class is for miriam cross reference annotation.

```
 BSON structure:
{
    _id: <string>, #madatory
    type: <string>,
    ontologyRelationship: {<RelationshipBOSON>, <RelationshipBOSON>, ...},
    name: {<string>, <string>, ...}
    comment: <string>,
    dataPrimarySource: <XrefBOSON>,
    functionAnnotation: {<AnnotationBOSON>, <AnnotationBOSON>, ...},
    evidence: {<EvidenceBOSON>, <EvidenceBOSON>, ...},
    dataxref: {<XrefBOSON>, <XrefBOSON>, ...},
    participant: {<StoichiometryBOSON>, <StoichiometryBOSON>, ...},
    interactionKey: <string>,
    interactionType: <DbRef>,
    phynotype: <DbRef>
}
```

9.23.2 Constructor & Destructor Documentation

9.23.2.1 unisys::GeneticInteraction::GeneticInteraction ()

Default constructor.

9.23.2.2 unisys::GeneticInteraction::GeneticInteraction (mongo::BSONObj const & bsonObj)

Overloaded constructor is used when retrieving data in boson object from database and transform to C++ object.

9.23.3 Member Function Documentation

9.23.3.1 void unisys::GeneticInteraction::initField () [private], [virtual]

Reimplemented from [unisys::Interaction](#).

9.23.3.2 void unisys::GeneticInteraction::setInteractionType (OntoldRef & ontoldRef)

9.23.3.3 void unisys::GeneticInteraction::setPhenotype (OntoldRef & ontoldRef)

The documentation for this class was generated from the following file:

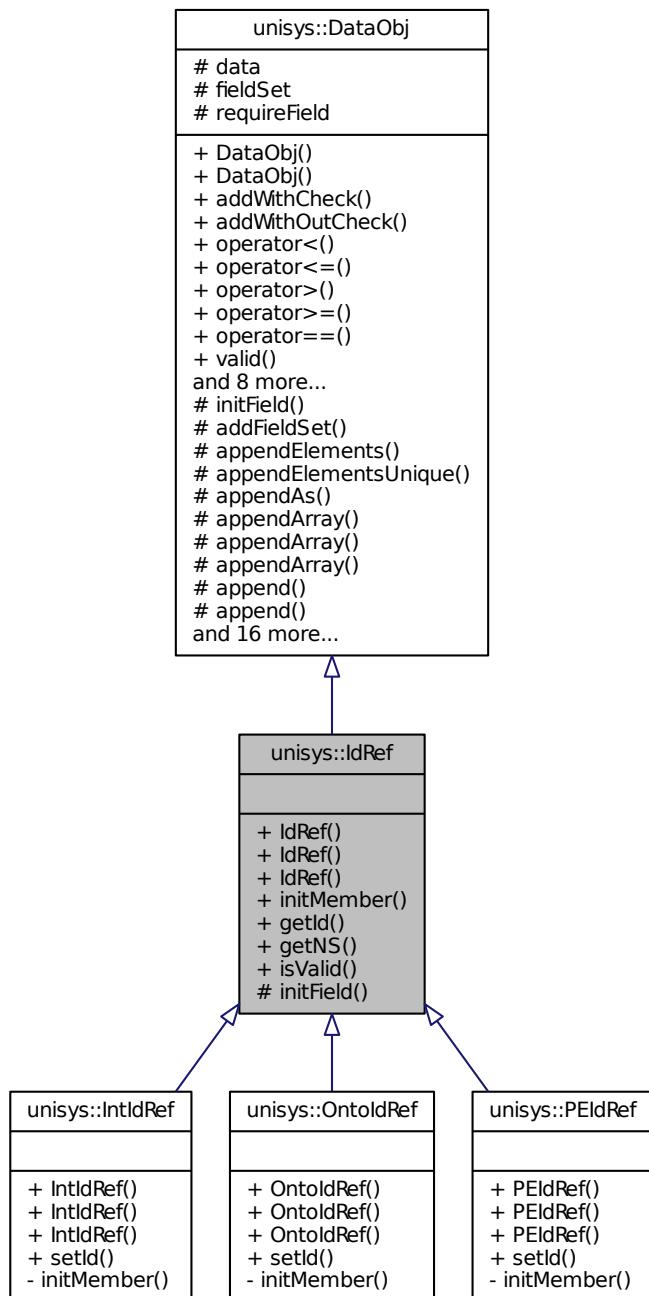
- [ObjClass.h](#)

9.24 unisys::IdRef Class Reference

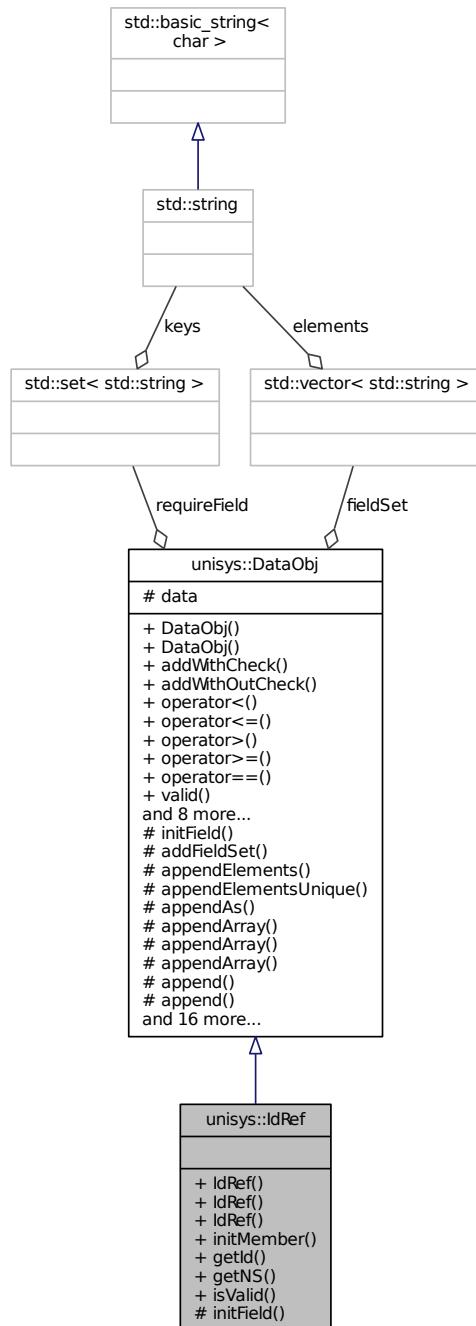
Identifier reference class.

```
#include <DBClass.h>
```

Inheritance diagram for unisys::IdRef:



Collaboration diagram for unisys::IdRef:



Public Member Functions

- [IdRef \(\)](#)
Default constructor.
- [IdRef \(mongo::BSONObj const &bsonObj\)](#)
Overloaded constructor get a BSONObj as parameters.
- [IdRef \(std::string const &DBId, std::string const &collectionNS\)](#)

- Overloaded constructor get a id string and collectionNS as parameters.
- void **initMember** (std::string const &DBId, std::string const &collectionNS)
- std::string **getId** () const
 - Return Id string.
- std::string **getNS** () const
- bool **isValid** () const
 - Check validity of class.

Protected Member Functions

- void **initField** ()
 - set the field names

Additional Inherited Members

9.24.1 Detailed Description

Identifier reference class.

This class is used for storing database reference (DBRef) under MongoDb DBRef format, which is {"\$ref": "collectionNS", "\$id": "id"}, but it support only reference between objects in the same database. This class provides the member functions to manage, converse to another class and to dereference the data from specific database handler.

BSON structure: { \$ref: <collname>, \$id: <idvalue>, }

9.24.2 Constructor & Destructor Documentation

9.24.2.1 unisys::IdRef::IdRef()

Default constructor.

9.24.2.2 unisys::IdRef::IdRef(mongo::BSONObj const & bsonObj)

Overloaded constructor get a BSONObj as parameters.

9.24.2.3 unisys::IdRef::IdRef(std::string const & DBId, std::string const & collectionNS)

Overloaded constructor get a id string and collectionNS as parameters.

9.24.3 Member Function Documentation

9.24.3.1 std::string unisys::IdRef::getId() const

Return Id string.

9.24.3.2 std::string unisys::IdRef::getNS() const

9.24.3.3 void unisys::IdRef::initField() [protected], [virtual]

set the field names

Implements [unisys::DataObj](#).

9.24.3.4 void unisys::IdRef::initMember (std::string const & DBId, std::string const & collectionNS)

9.24.3.5 bool unisys::IdRef::isValid () const

Check validity of class.

Reimplemented from [unisys::DataObj](#).

The documentation for this class was generated from the following file:

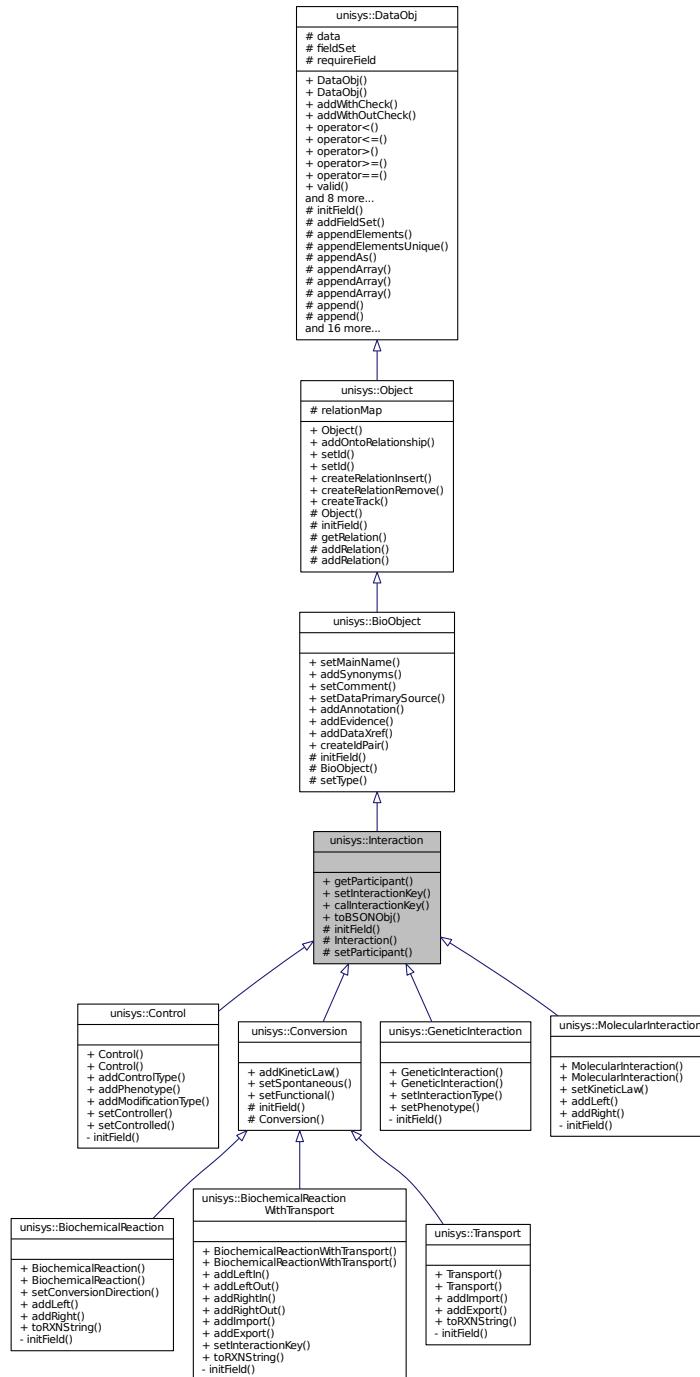
- [DBClass.h](#)

9.25 unisys::Interaction Class Reference

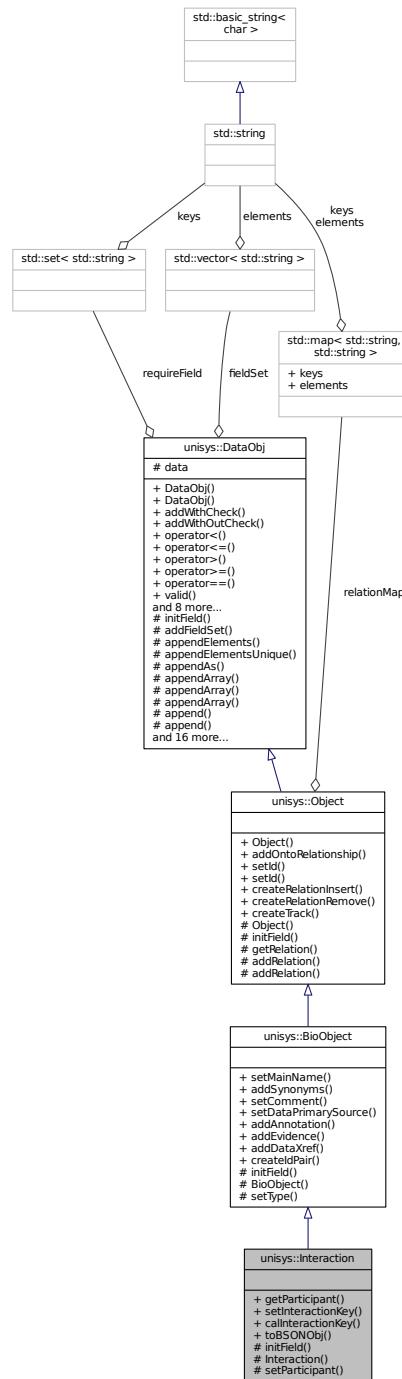
This class is for miriam cross reference annotation.

```
#include <ObjClass.h>
```

Inheritance diagram for unisys::Interaction:



Collaboration diagram for unisys::Interaction:



Public Member Functions

- `std::set< PEIdRef > getParticipant () const`
- `void setInteractionKey ()`
- `std::string callInteractionKey ()`
- `mongo::BSONObj toBSONObj ()`

Protected Member Functions

- void [initField \(\)](#)
- [Interaction \(\)](#)
- void [setParticipant \(mongo::BSNOBJ const &bsonObj\)](#)

9.25.1 Detailed Description

This class is for miriam cross reference annotation.

```
BSON structure:
{
    _id: <string>, #madatory
    ontologyRelationship: {<RelationshipBOSON>, <RelationshipBOSON>, ...},
    name: {<string>, <string>, ...}
    comment: <string>,
    dataPrimarySource: <XrefBOSON>,
    functionAnnotation: {<AnnotationBOSON>, <AnnotationBOSON>, ...},
    evidence: {<EvidenceBOSON>, <EvidenceBOSON>, ...},
    dataxref: {<XrefBOSON>, <XrefBOSON>, ...},
    participant: {<StoichiometryBOSON>, <StoichiometryBOSON>, ...},
    interactionKey: <string>
}
```

9.25.2 Constructor & Destructor Documentation

9.25.2.1 [unisys::Interaction::Interaction \(\) \[protected\]](#)

9.25.3 Member Function Documentation

9.25.3.1 [std::string unisys::Interaction::callInteractionKey \(\)](#)

9.25.3.2 [std::set<PEIdRef> unisys::Interaction::getParticipant \(\) const](#)

9.25.3.3 [void unisys::Interaction::initField \(\) \[protected\], \[virtual\]](#)

Reimplemented from [unisys::BioObject](#).

Reimplemented in [unisys::BiochemicalReactionWithTransport](#), [unisys::Transport](#), [unisys::BiochemicalReaction](#), [unisys::Conversion](#), [unisys::GeneticInteraction](#), [unisys::MolecularInteraction](#), and [unisys::Control](#).

9.25.3.4 [void unisys::Interaction::setInteractionKey \(\)](#)

Reimplemented in [unisys::BiochemicalReactionWithTransport](#).

9.25.3.5 [void unisys::Interaction::setParticipant \(mongo::BSNOBJ const & bsonObj \) \[protected\]](#)

9.25.3.6 [mongo::BSNOBJ unisys::Interaction::toBSNOBJ \(\)](#)

Reimplemented from [unisys::DataObj](#).

The documentation for this class was generated from the following file:

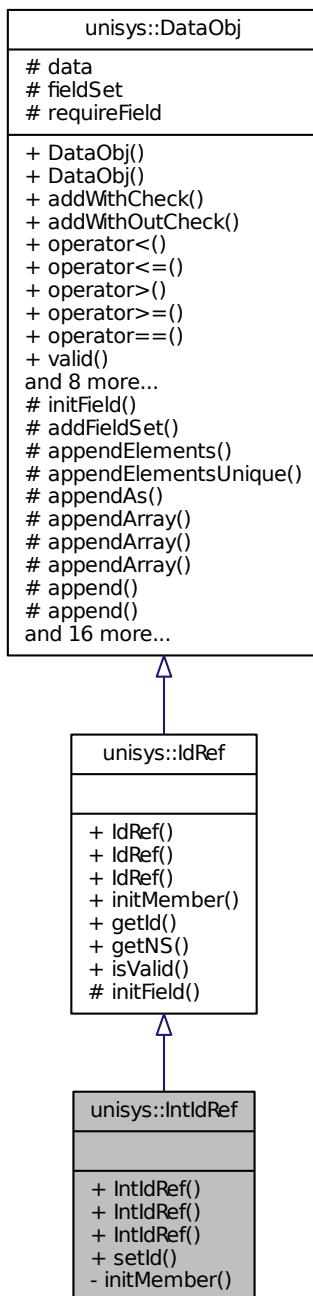
- [ObjClass.h](#)

9.26 unisys::IntIdRef Class Reference

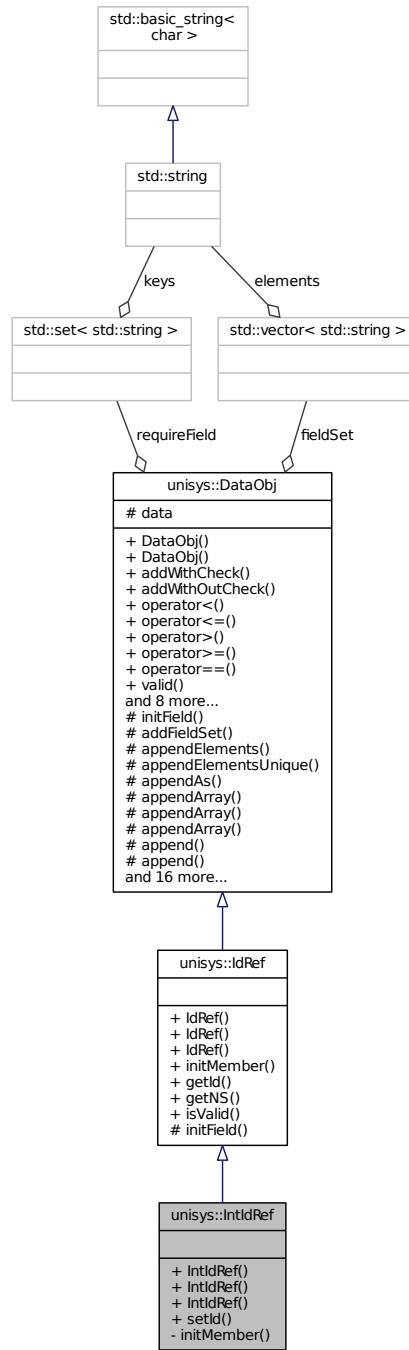
Identifier reference class specific to linteraction collection namespace.

```
#include <DBClass.h>
```

Inheritance diagram for unisys::IntIdRef:



Collaboration diagram for unisys::IntIdRef:



Public Member Functions

- **IntIdRef ()**
Default constructor.
- **IntIdRef (mongo::BSONObj const &bsonObj)**
- **IntIdRef (std::string const &DBId)**
Overloaded constructor get a object id string as parameter.

- void [setId](#) (std::string const &DBId)

Private Member Functions

- void [initMember](#) ()

Additional Inherited Members

9.26.1 Detailed Description

Identifier reference class specific to interaction collection namespace.

This class is the same as [IdRef](#), but has specific collection namespace to interaction collection namespace.

9.26.2 Constructor & Destructor Documentation

9.26.2.1 [unisys::IntIdRef::IntIdRef\(\)](#) [inline]

Default constructor.

9.26.2.2 [unisys::IntIdRef::IntIdRef\(mongo::BSONObj const & bsonObj \)](#) [inline]

9.26.2.3 [unisys::IntIdRef::IntIdRef\(std::string const & DBId \)](#) [inline]

Overloaded constructor get a object id string as parameter.

9.26.3 Member Function Documentation

9.26.3.1 [void unisys::IntIdRef::initMember\(\)](#) [inline], [private]

9.26.3.2 [void unisys::IntIdRef::setId\(std::string const & DBId \)](#) [inline]

The documentation for this class was generated from the following file:

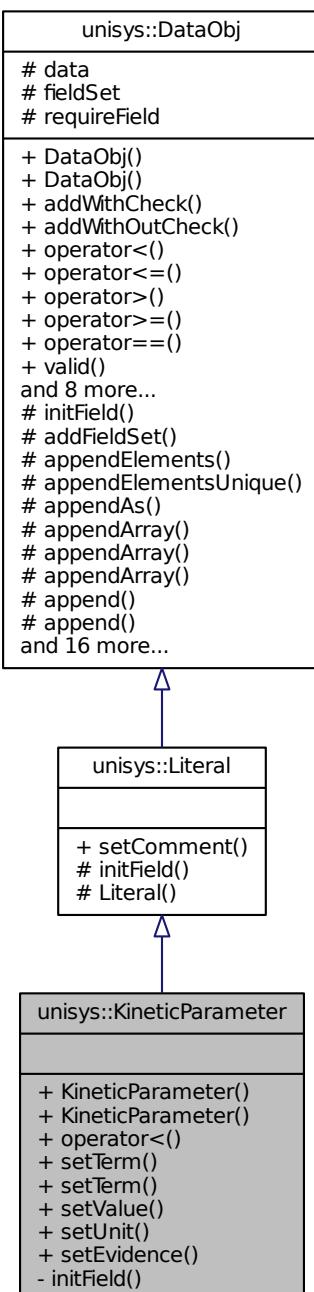
- [DBClass.h](#)

9.27 [unisys::KineticParameter Class Reference](#)

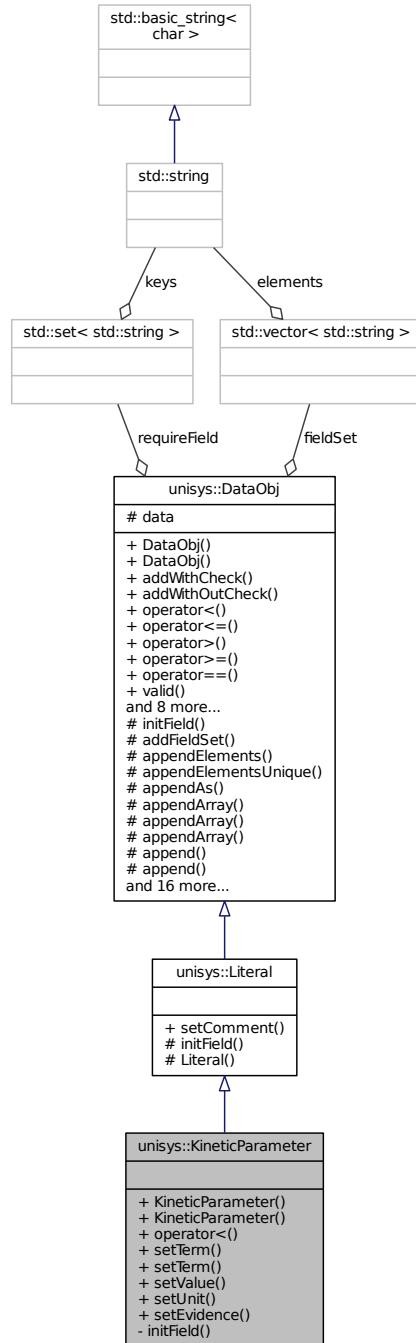
The C++ representative class for kinetic parameter data class.

```
#include <LitClass.h>
```

Inheritance diagram for unisys::KineticParameter:



Collaboration diagram for unisys::KineticParameter:



Public Member Functions

- [KineticParameter \(\)](#)
Default constructor.
- [KineticParameter \(mongo::BSONObj const &bsonObj\)](#)
Overloaded constructor is used when retrieving data in bson object from database and transform to C++ object.
- [bool operator< \(KineticParameter const &other\) const](#)

Less than operator overload.

- void [setTerm](#) ([OntoldRef](#) &ontoldRef)
- void [setTerm](#) (std::string const &ontold)
- void [setValue](#) (double value)
- void [setUnit](#) ([OntoldRef](#) &ontoldRef)
- void [setEvidence](#) ([Evidence](#) &evidence)

Private Member Functions

- void [initField](#) ()

function for init field in the object

Additional Inherited Members

9.27.1 Detailed Description

The C++ representative class for kinetic parameter data class.

```
 BSON structure:
{
    comment: <string>,
    term: {$ref: <collname>, $id: <idvalue>}, #format by idref class
    value: <number>,
    unit: {$ref: <collname>, $id: <idvalue>}, #format by idref class
    evidence: <EvidenceBSONStructure>
}
```

9.27.2 Constructor & Destructor Documentation

9.27.2.1 unisys::KineticParameter::KineticParameter()

Default constructor.

9.27.2.2 unisys::KineticParameter::KineticParameter(mongo::BSONObj const & bsonObj)

Overloaded constructor is used when retrieving data in boson object from database and transform to C++ object.

9.27.3 Member Function Documentation

9.27.3.1 void unisys::KineticParameter::initField() [private], [virtual]

function for init field in the object

Reimplemented from [unisys::Literal](#).

9.27.3.2 bool unisys::KineticParameter::operator< (KineticParameter const & other) const

Less than operator overload.

9.27.3.3 void unisys::KineticParameter::setEvidence (Evidence & evidence)

9.27.3.4 void unisys::KineticParameter::setTerm (OntoldRef & ontoldRef)

9.27.3.5 void unisys::KineticParameter::setTerm (std::string const & *ontold*)

9.27.3.6 void unisys::KineticParameter::setUnit (OntoldRef & *ontoldRef*)

9.27.3.7 void unisys::KineticParameter::setValue (double value)

The documentation for this class was generated from the following file:

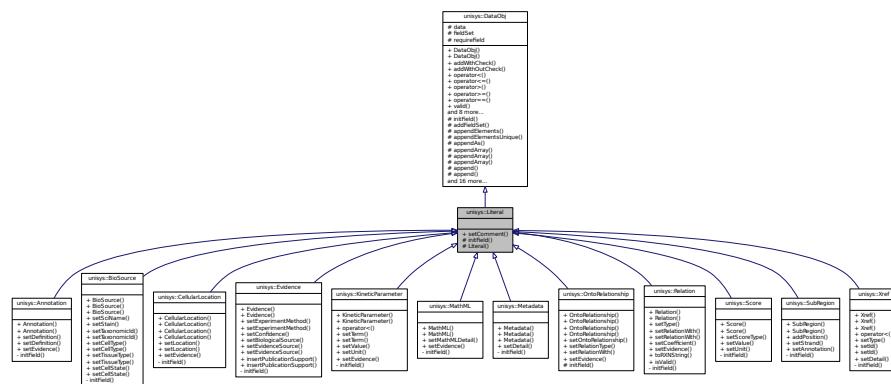
- LitClass.h

9.28 unisys::Literal Class Reference

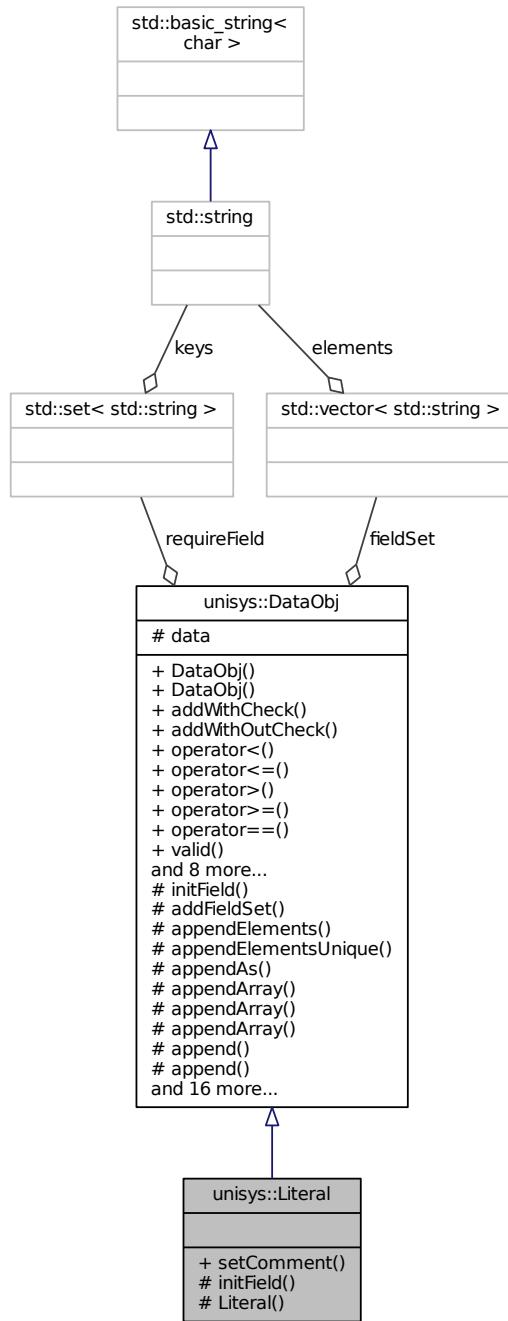
Literal class.

```
#include <LitClass.h>
```

Inheritance diagram for unisys::Literal:



Collaboration diagram for unisys::Literal:



Public Member Functions

- void `setComment` (std::string const &comment)

Protected Member Functions

- void `initField` ()

function for init field in the object

- [Literal \(\)](#)

Default constructor.

Additional Inherited Members

9.28.1 Detailed Description

[Literal](#) class.

This is the root of all literal classes.

9.28.2 Constructor & Destructor Documentation

9.28.2.1 `unisys::Literal::Literal() [protected]`

Default constructor.

9.28.3 Member Function Documentation

9.28.3.1 `void unisys::Literal::initField() [protected], [virtual]`

function for init field in the object

Implements [unisys::DataObj](#).

Reimplemented in [unisys::Metadata](#), [unisys::SubRegion](#), [unisys::KineticParameter](#), [unisys::MathML](#), [unisys::CellularLocation](#), [unisys::Relation](#), [unisys::OntoRelationship](#), [unisys::Annotation](#), [unisys::Evidence](#), [unisys::Bio-Source](#), [unisys::Score](#), and [unisys::Xref](#).

9.28.3.2 `void unisys::Literal::setComment(std::string const & comment)`

The documentation for this class was generated from the following file:

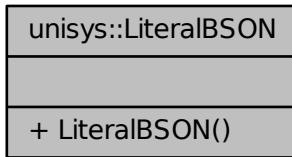
- [LitClass.h](#)

9.29 `unisys::LiteralBSON` Class Reference

Specific BSON object to manage general [Literal](#) data object.

```
#include <DBClass.h>
```

Collaboration diagram for unisys::LiteralBSON:



Public Member Functions

- [LiteralBSON \(\)](#)

Default constructor.

9.29.1 Detailed Description

Specific BSON object to manage general [Literal](#) data object.

9.29.2 Constructor & Destructor Documentation

9.29.2.1 unisys::LiteralBSON::LiteralBSON()

Default constructor.

The documentation for this class was generated from the following file:

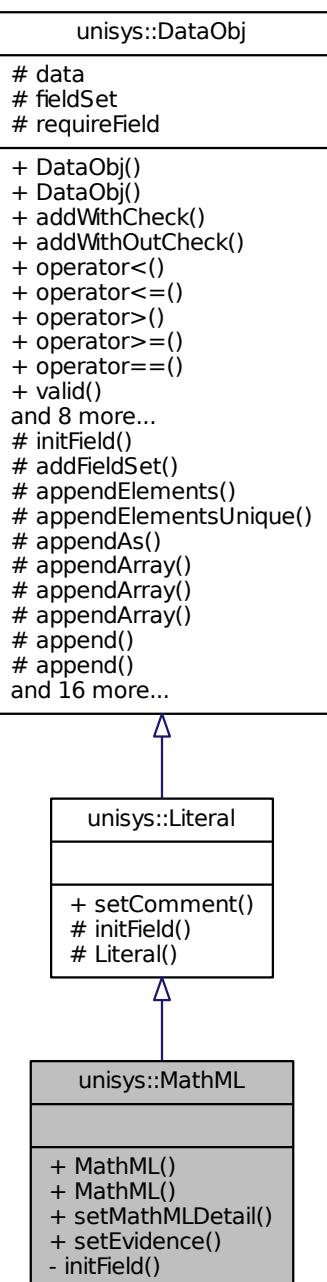
- [DBClass.h](#)

9.30 unisys::MathML Class Reference

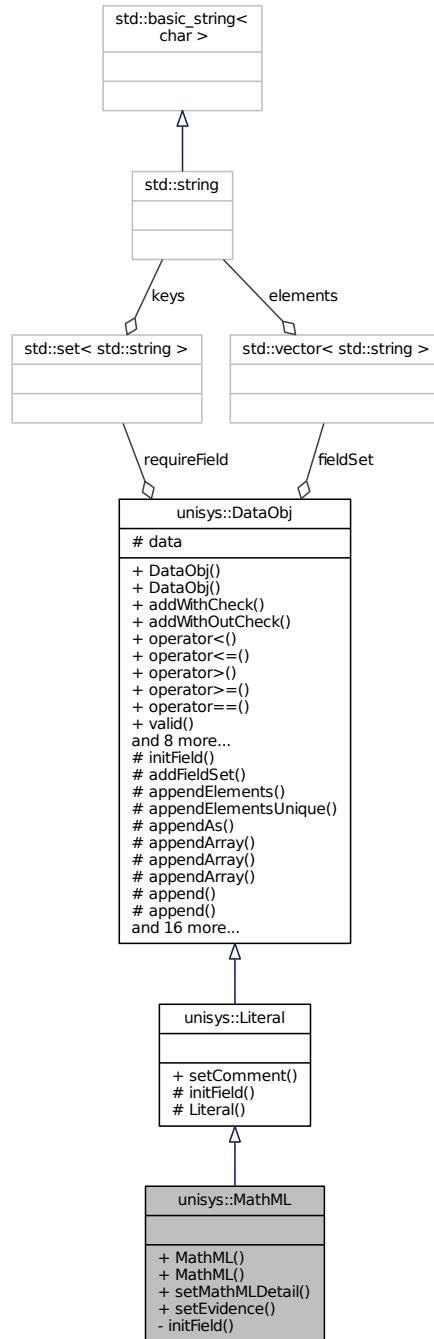
The C++ representative class for [MathML](#) data class.

```
#include <LitClass.h>
```

Inheritance diagram for unisys::MathML:



Collaboration diagram for unisys::MathML:



Public Member Functions

- [MathML \(\)](#)
Default constructor.
- [MathML \(mongo::BSONObj const &bsonObj\)](#)
Overloaded constructor is used when retrieving data in bson object from database and transform to C++ object.
- void [setMathMLDetail \(std::string const &mathMLDetail\)](#)

- void [setEvidence \(Evidence &evidence\)](#)

Private Member Functions

- void [initField \(\)](#)
function for init field in the object

Additional Inherited Members

9.30.1 Detailed Description

The C++ representative class for [MathML](#) data class.

```
 BSON structure:
{
    comment: <string>,
    mathMLDetail: <string>
}
```

9.30.2 Constructor & Destructor Documentation

9.30.2.1 [unisys::MathML::MathML \(\)](#)

Default constructor.

9.30.2.2 [unisys::MathML::MathML \(mongo::BSONObj const & bsonObj \)](#)

Overloaded constructor is used when retrieving data in boson object from database and transform to C++ object.

9.30.3 Member Function Documentation

9.30.3.1 [void unisys::MathML::initField \(\) \[private\], \[virtual\]](#)

function for init field in the object

Reimplemented from [unisys::Literal](#).

9.30.3.2 [void unisys::MathML::setEvidence \(Evidence & evidence \)](#)

9.30.3.3 [void unisys::MathML::setMathMLDetail \(std::string const & mathMLDetail \)](#)

The documentation for this class was generated from the following file:

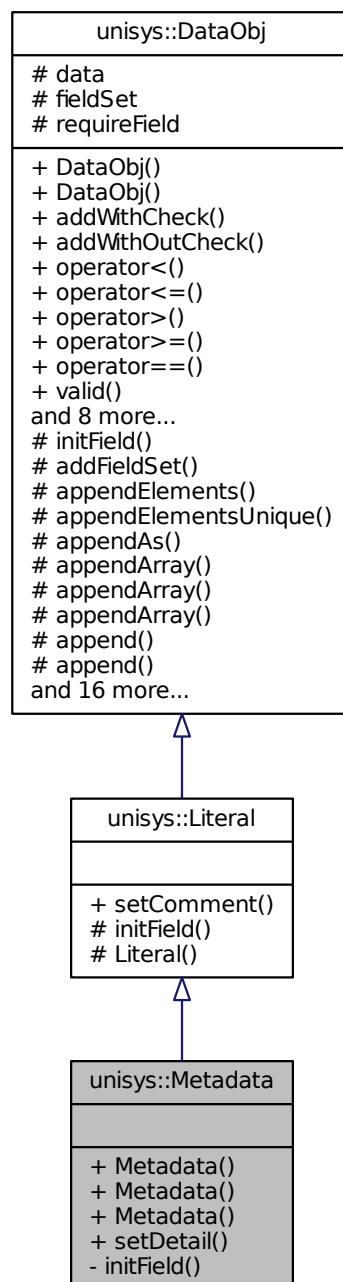
- [LitClass.h](#)

9.31 [unisys::Metadata Class Reference](#)

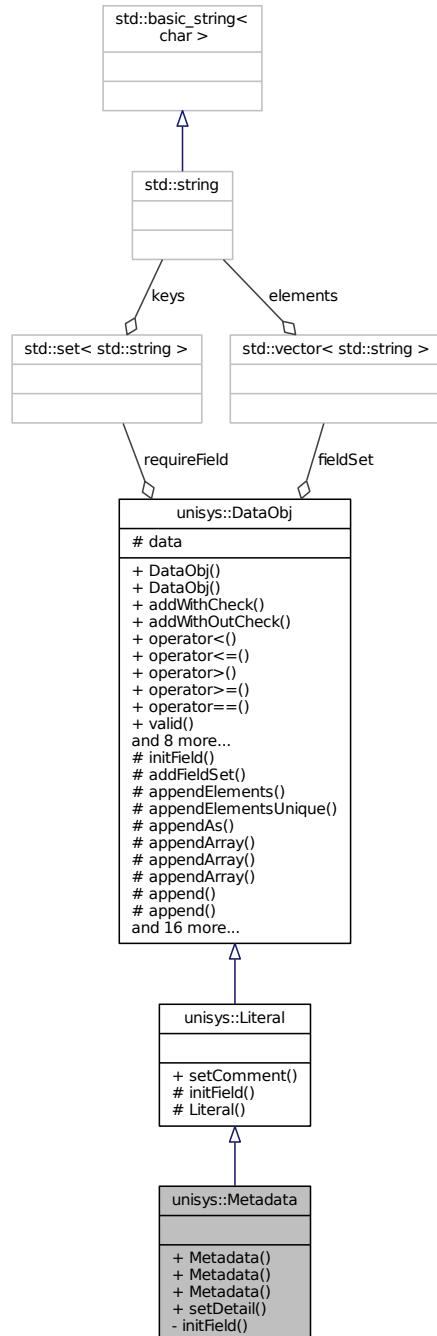
The C++ representative class for metadata data class.

```
#include <LitClass.h>
```

Inheritance diagram for unisys::Metadata:



Collaboration diagram for unisys::Metadata:



Public Member Functions

- [Metadata \(\)](#)
Default constructor.
- [Metadata \(mongo::BSONObj const &bsonObj\)](#)
Overloaded constructor is used when retrieving data in bson object from database and transform to C++ object.
- [Metadata \(std::string const &detail\)](#)

- void [setDetail](#) (std::string const &detail)

Private Member Functions

- void [initField](#) ()
function for init field in the object

Additional Inherited Members

9.31.1 Detailed Description

The C++ representative class for metadata data class.

```
BSON structure:  
{  
    comment: <string>,  
    detail: <string>  
}
```

9.31.2 Constructor & Destructor Documentation

9.31.2.1 [unisys::Metadata::Metadata\(\)](#)

Default constructor.

9.31.2.2 [unisys::Metadata::Metadata\(mongo::BSONObj const & bsonObj \)](#)

Overloaded constructor is used when retrieving data in boson object from database and transform to C++ object.

9.31.2.3 [unisys::Metadata::Metadata\(std::string const & detail \)](#)

9.31.3 Member Function Documentation

9.31.3.1 [void unisys::Metadata::initField\(\) \[private\], \[virtual\]](#)

function for init field in the object

Reimplemented from [unisys::Literal](#).

9.31.3.2 [void unisys::Metadata::setDetail\(std::string const & detail \)](#)

The documentation for this class was generated from the following file:

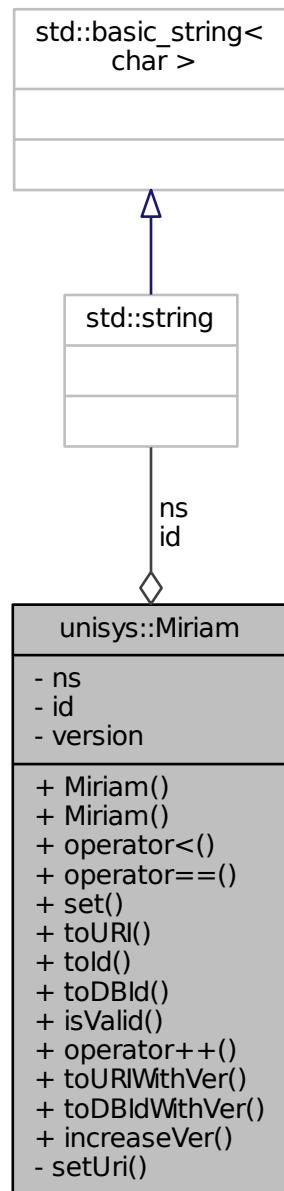
- [LitClass.h](#)

9.32 unisys::Miriam Class Reference

This class is for miriam cross reference annotation.

```
#include <DBClass.h>
```

Collaboration diagram for unisys::Miriam:



Public Member Functions

- [Miriam \(\)](#)
Default constructor.
- [Miriam \(std::string const &**id**, std::string const &**ns**= "", unsigned int **version**=0\)](#)
Overloaded constructor.
- [bool operator< \(Miriam const &other\) const](#)
Less than operator overload.
- [bool operator== \(Miriam const &other\) const](#)

- Equal operator overload.
- void `set` (std::string const &`id`, std::string const &`ns`= "", unsigned int `version`=0)
- std::string `toURI` () const
 - return URI string*
- std::string `told` () const
 - return Id string*
- std::string `toDBId` () const
 - return Database system ID string (domain.sub-domain namespace:Identifier)*
- bool `isValid` () const
 - Check validity of data.*
- void `operator++` (int number=1)
 - Increase the number of version with 1.*
- std::string `toURIWithVer` () const
 - return URI string appended with version number(super-domain:domain.sub-domain namespace:Identifier.Version)*
- std::string `toDBIdWithVer` () const
 - return Database system ID string appended with version number(domain.sub-domain namespace:Identifier.Version)*
- void `increaseVer` (int number=1)
 - Increase the number of version with 1.*

Private Member Functions

- void `setUri` (std::string const &uri)

Private Attributes

- std::string `ns`
 - domain.sub-domain namespace*
- std::string `id`
 - Identifier.*
- unsigned int `version`
 - Version.*

9.32.1 Detailed Description

This class is for miriam cross reference annotation.

This Mirium object is a part of standard guidelines in the controlled annotation of model components, based on Uniform Resource Identifiers (URI). The URI format comprise of two parts URN:identifier. URN or Uniform Resource Name is in format "urn:miriam:domain[.sub-domain]"; such as urn:miriam:kegg.compound. In this class structure, only domain and sub-domain are stored in class data member. Identifier is the id of object reference to its own URN.

9.32.2 Constructor & Destructor Documentation

9.32.2.1 unisys::Miriam::Miriam ()

Default constructor.

9.32.2.2 unisys::Miriam::Miriam (std::string const & `id`, std::string const & `ns` = " ", unsigned int `version` = 0)

Overloaded constructor.

9.32.3 Member Function Documentation

9.32.3.1 `void unisys::Miriam::increaseVer(int number = 1)`

Increase the number of version with 1.

9.32.3.2 `bool unisys::Miriam::isValid() const`

Check validity of data.

9.32.3.3 `void unisys::Miriam::operator++(int number = 1)`

Increase the number of version with 1.

9.32.3.4 `bool unisys::Miriam::operator<(Miriam const & other) const`

Less than operator overload.

9.32.3.5 `bool unisys::Miriam::operator==(Miriam const & other) const`

Equal operator overload.

9.32.3.6 `void unisys::Miriam::set(std::string const & id, std::string const & ns = "", unsigned int version = 0)`

9.32.3.7 `void unisys::Miriam::setUri(std::string const & uri) [private]`

9.32.3.8 `std::string unisys::Miriam::toDBId() const`

return [Database](#) system ID string (domain.sub-domain namespace:Identifier)

9.32.3.9 `std::string unisys::Miriam::toDBIdWithVer() const`

return [Database](#) system ID string appended with version number(domain.sub-domain namespace:Identifier.Version)

9.32.3.10 `std::string unisys::Miriam::told() const`

return Id string

9.32.3.11 `std::string unisys::Miriam::toURI() const`

return URI string

9.32.3.12 `std::string unisys::Miriam::toURIWithVer() const`

return URI string appended with version number(super-domain:domain.sub-domain namespace:Identifier.Version)

9.32.4 Member Data Documentation

9.32.4.1 `std::string unisys::Miriam::id` [private]

Identifier.

9.32.4.2 `std::string unisys::Miriam::ns` [private]

domain.sub-domain namespace

9.32.4.3 `unsigned int unisys::Miriam::version` [private]

Version.

The documentation for this class was generated from the following file:

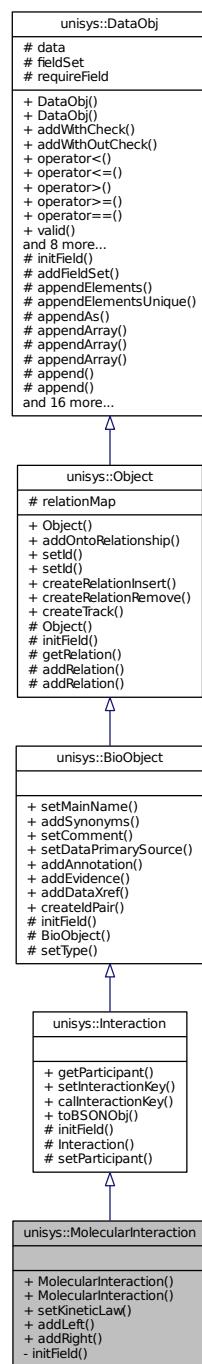
- [DBClass.h](#)

9.33 unisys::MolecularInteraction Class Reference

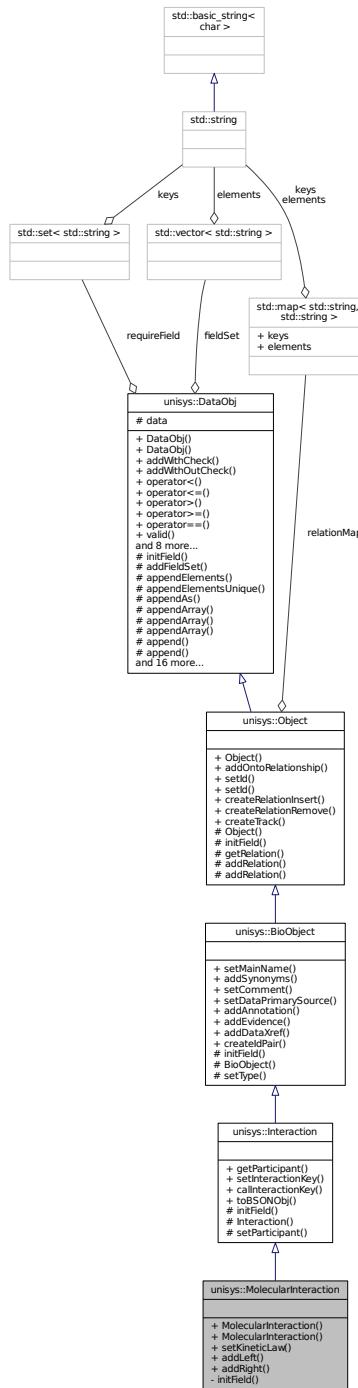
This class is for miriam cross reference annotation.

```
#include <ObjClass.h>
```

Inheritance diagram for unisys::MolecularInteraction:



Collaboration diagram for unisys::MolecularInteraction:



Public Member Functions

- [MolecularInteraction \(\)](#)
Default constructor.
- [MolecularInteraction \(mongo::BSONObj const &bsonObj\)](#)
Overloaded constructor is used when retrieving data in bson object from database and transform to C++ object.
- [void setKineticLaw \(MathML &kineticLaw\)](#)

- void [addLeft \(PEIdRef &peldRef, double coefficient\)](#)
- void [addRight \(PEIdRef &peldRef, double coefficient\)](#)

Private Member Functions

- void [initField \(\)](#)

Additional Inherited Members

9.33.1 Detailed Description

This class is for miriam cross reference annotation.

```
BSON structure:
{
    _id: <string>, #madatory
    type: <string>,
    ontologyRelationship: {<RelationshipBOSON>, <RelationshipBOSON>, ...},
    name: {<string>, <string>, ...}
    comment: <string>,
    dataPrimarySource: <XrefBOSON>,
    functionAnnotation: {<AnnotationBOSON>, <AnnotationBOSON>, ...},
    evidence: {<EvidenceBOSON>, <EvidenceBOSON>, ...},
    dataxref: {<XrefBOSON>, <XrefBOSON>, ...},
    participant: {<StoichiometryBOSON>, <StoichiometryBOSON>, ...},
    interactionKey: <string>,
    kineticLaw: <MathMLBOSON>,
}
```

9.33.2 Constructor & Destructor Documentation

9.33.2.1 unisys::MolecularInteraction::MolecularInteraction ()

Default constructor.

9.33.2.2 unisys::MolecularInteraction::MolecularInteraction (mongo::BSONObj const & bsonObj)

Overloaded constructor is used when retrieving data in boson object from database and transform to C++ object.

9.33.3 Member Function Documentation

9.33.3.1 void unisys::MolecularInteraction::addLeft (PEIdRef & peldRef, double coefficient)

9.33.3.2 void unisys::MolecularInteraction::addRight (PEIdRef & peldRef, double coefficient)

9.33.3.3 void unisys::MolecularInteraction::initField () [private], [virtual]

Reimplemented from [unisys::Interaction](#).

9.33.3.4 void unisys::MolecularInteraction::setKineticLaw (MathML & kineticLaw)

The documentation for this class was generated from the following file:

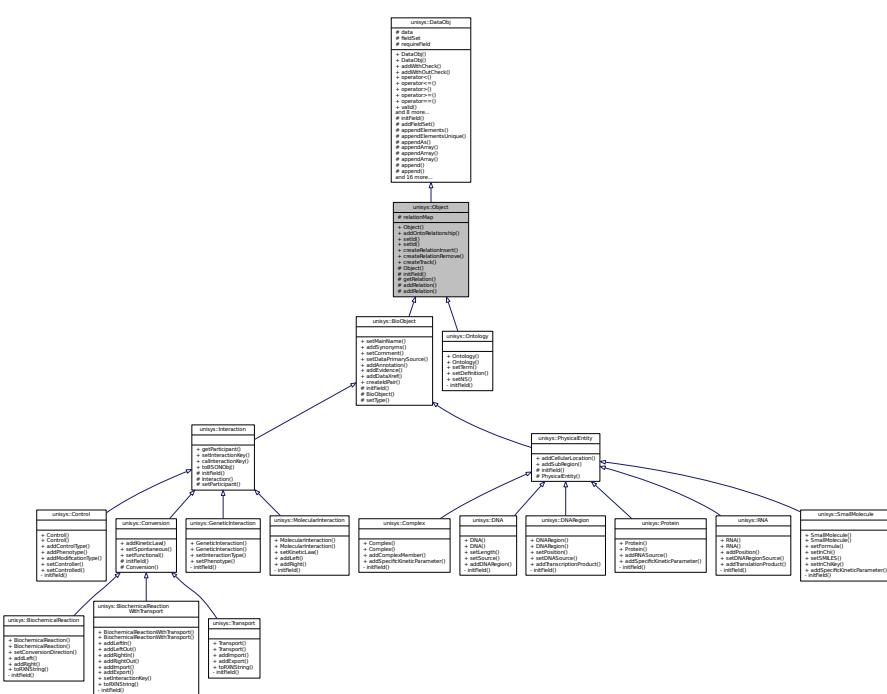
- [ObjClass.h](#)

9.34 unisys::Object Class Reference

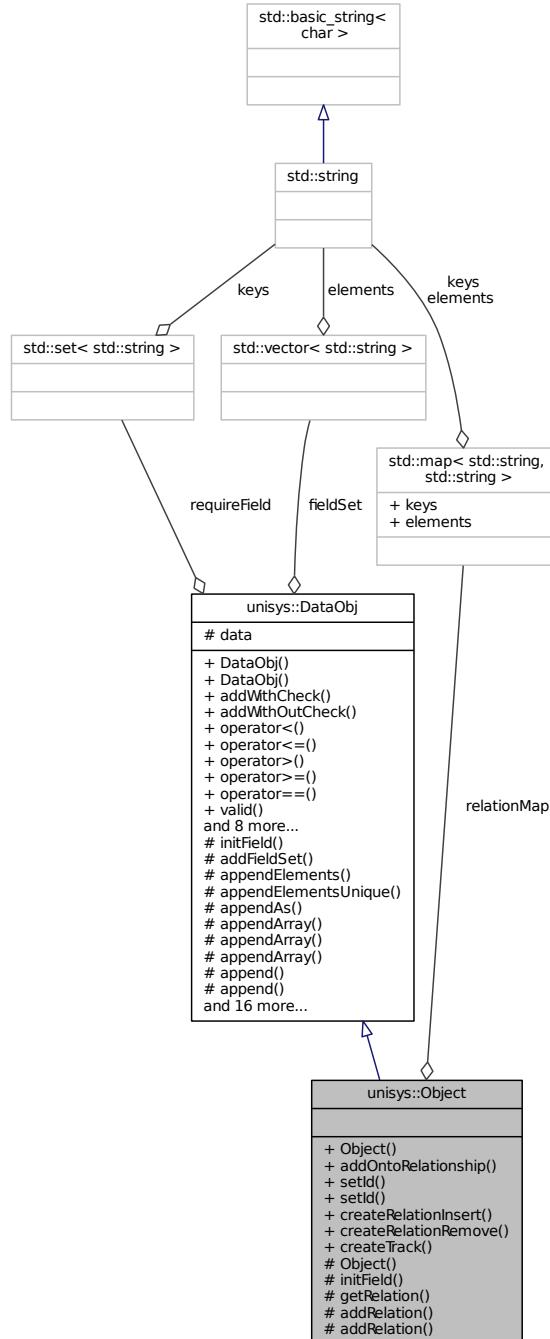
Root of all object classes This class does not have any interfaces.

```
#include <ObjClass.h>
```

Inheritance diagram for unisys::Object:



Collaboration diagram for unisys::Object:



Public Member Functions

- `Object (mongo::BSONObj const &bsonObj)`

Overloaded constructor is used when retrieving data in bson object from database and transform to C++ object.

- `void addOntoRelationship (OntoRelationship &ontorelationship)`
- `void setId (Miriam const &miriam, bool withVer=true)`
- `void setId (std::string const &id, std::string const &ns="", unsigned int version=0, bool withVer=true)`

- mongo::BSONObj [createRelationInsert \(\) const](#)
- mongo::BSONObj [createRelationRemove \(\) const](#)
- Tracking [createTrack \(std::string activity=""\) const](#)

Protected Member Functions

- [Object \(\)](#)
- void [initField \(\)](#)
- std::set< [Relation > getRelation \(std::string const &type\) const](#)
- void [addRelation \(Relation &relation\)](#)
- void [addRelation \(std::string const &type, IdRef &idRef, double coefficient=0, bool canDuplicate=false\)](#)

Protected Attributes

- std::map< std::string, std::string > [relationMap](#)

9.34.1 Detailed Description

Root of all object classes This class does not have any interfaces.

```
BSON structure:
{
    _id: <string>, #mandatory
    ontologyRelationship: {<RelationshipBOSON>, <RelationshipBOSON>, ...},
    relation: {<RelationBOSON>, <RelationBOSON>, ...}
}
```

9.34.2 Constructor & Destructor Documentation

9.34.2.1 [unisys::Object::Object\(\) \[protected\]](#)

9.34.2.2 [unisys::Object::Object \(mongo::BSONObj const & bsonObj \)](#)

Overloaded constructor is used when retrieving data in boson object from database and transform to C++ object.

9.34.3 Member Function Documentation

9.34.3.1 [void unisys::Object::addOntoRelationship \(OntoRelationship & ontorelationship \)](#)

9.34.3.2 [void unisys::Object::addRelation \(Relation & relation \) \[protected\]](#)

9.34.3.3 [void unisys::Object::addRelation \(std::string const & type, IdRef & idRef, double coefficient = 0, bool canDuplicate = false \) \[protected\]](#)

9.34.3.4 [mongo::BSONObj unisys::Object::createRelationInsert \(\) const](#)

9.34.3.5 [mongo::BSONObj unisys::Object::createRelationRemove \(\) const](#)

9.34.3.6 [Tracking unisys::Object::createTrack \(std::string activity = " " \) const](#)

9.34.3.7 [std::set<Relation> unisys::Object::getRelation \(std::string const & type \) const \[protected\]](#)

9.34.3.8 void unisys::Object::initField() [protected], [virtual]

Implements [unisys::DataObj](#).

Reimplemented in [unisys::BiochemicalReactionWithTransport](#), [unisys::Transport](#), [unisys::BiochemicalReaction](#), [unisys::Conversion](#), [unisys::GeneticInteraction](#), [unisys::MolecularInteraction](#), [unisys::Control](#), [unisys::Interaction](#), [unisys::Complex](#), [unisys::Protein](#), [unisys::RNA](#), [unisys::DNARegion](#), [unisys::DNA](#), [unisys::SmallMolecule](#), [unisys::PhysicalEntity](#), [unisys::BioObject](#), and [unisys::Ontology](#).

9.34.3.9 void unisys::Object::setId(*Miriam const & miriam*, *bool withVer = true*)

9.34.3.10 void unisys::Object::setId(*std::string const & id*, *std::string const & ns = " "*, *unsigned int version = 0*, *bool withVer = true*)

9.34.4 Member Data Documentation

9.34.4.1 std::map<std::string, std::string> unisys::Object::relationMap [protected]

The documentation for this class was generated from the following file:

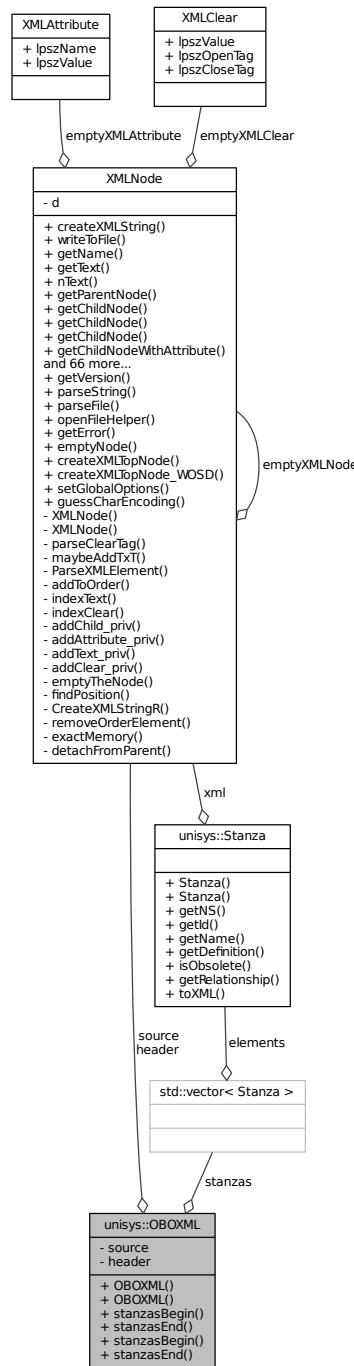
- [ObjClass.h](#)

9.35 unisys::OBOXML Class Reference

This class is used to parse OWL format.

```
#include <oboXML.h>
```

Collaboration diagram for unisys::OBOXML:



Public Member Functions

- `OBOXML ()`
- `OBOXML (std::string const &filename)`
- `std::vector< Stanza >::iterator stanzasBegin ()`
- `std::vector< Stanza >::iterator stanzasEnd ()`
- `std::vector< Stanza >::const_iterator stanzasBegin () const`

- std::vector< Stanza >
 ::const_iterator stanzasEnd () const

Private Attributes

- XMLNode source
- XMLNode header
- std::vector< Stanza > stanzas

9.35.1 Detailed Description

This class is used to parse OWL format.

some description.

9.35.2 Constructor & Destructor Documentation

9.35.2.1 unisys::OBOXML::OBOXML ()

9.35.2.2 unisys::OBOXML::OBOXML (std::string const & *filename*)

9.35.3 Member Function Documentation

9.35.3.1 std::vector<Stanza>::iterator unisys::OBOXML::stanzasBegin ()

9.35.3.2 std::vector<Stanza>::const_iterator unisys::OBOXML::stanzasBegin () const

9.35.3.3 std::vector<Stanza>::iterator unisys::OBOXML::stanzasEnd ()

9.35.3.4 std::vector<Stanza>::const_iterator unisys::OBOXML::stanzasEnd () const

9.35.4 Member Data Documentation

9.35.4.1 XMLNode unisys::OBOXML::header [private]

9.35.4.2 XMLNode unisys::OBOXML::source [private]

9.35.4.3 std::vector<Stanza> unisys::OBOXML::stanzas [private]

The documentation for this class was generated from the following file:

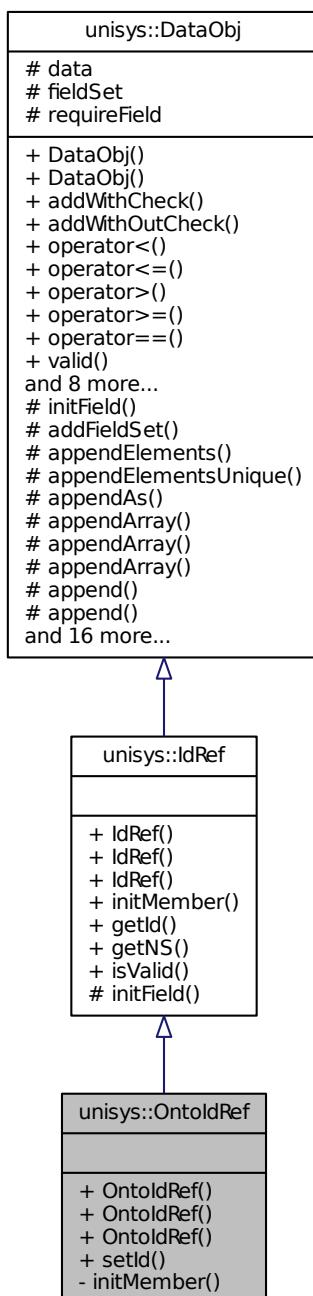
- oboXML.h

9.36 unisys::OntoldRef Class Reference

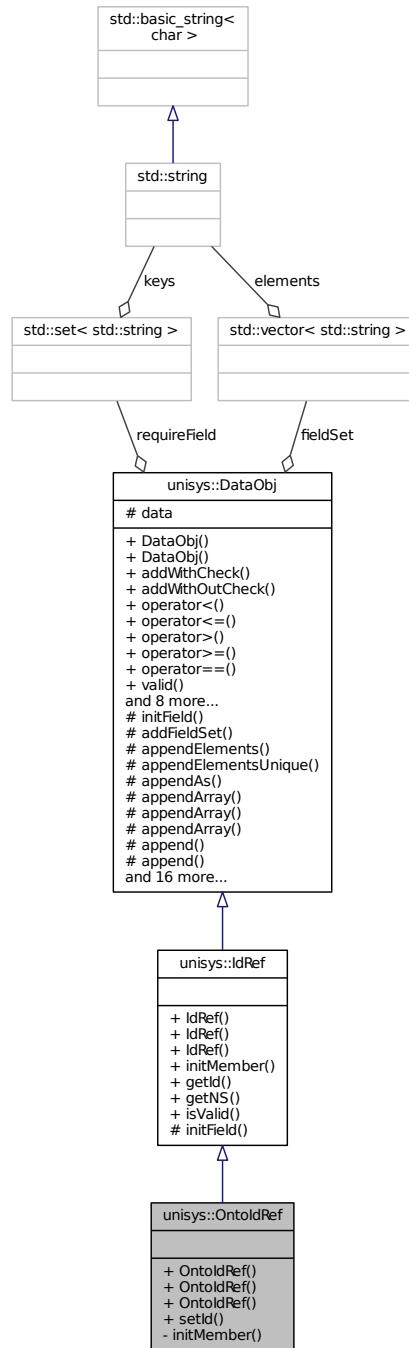
Identifier reference class specific to obo collection namespace.

```
#include <DBClass.h>
```

Inheritance diagram for unisys::OntoldRef:



Collaboration diagram for unisys::OntoldRef:



Public Member Functions

- **`OntoldRef ()`**
Default constructor.
- **`OntoldRef (mongo::BSONObj const &bsonObj)`**
- **`OntoldRef (std::string const &DBId)`**
Overloaded constructor get a object id string as parameter.

- void `setId` (std::string const &DBId)

Private Member Functions

- void `initMember` ()

Additional Inherited Members

9.36.1 Detailed Description

Identifier reference class specific to obo collection namespace.

This class is the same as `IdRef`, but has specific collection namespace to obo collection namespace.

9.36.2 Constructor & Destructor Documentation

9.36.2.1 `unisys::OntoldRef::OntoldRef() [inline]`

Default constructor.

9.36.2.2 `unisys::OntoldRef::OntoldRef(mongo::BSONObj const & bsonObj) [inline]`

9.36.2.3 `unisys::OntoldRef::OntoldRef(std::string const & DBId) [inline]`

Overloaded constructor get a object id string as parameter.

9.36.3 Member Function Documentation

9.36.3.1 `void unisys::OntoldRef::initMember() [inline], [private]`

9.36.3.2 `void unisys::OntoldRef::setId(std::string const & DBId) [inline]`

The documentation for this class was generated from the following file:

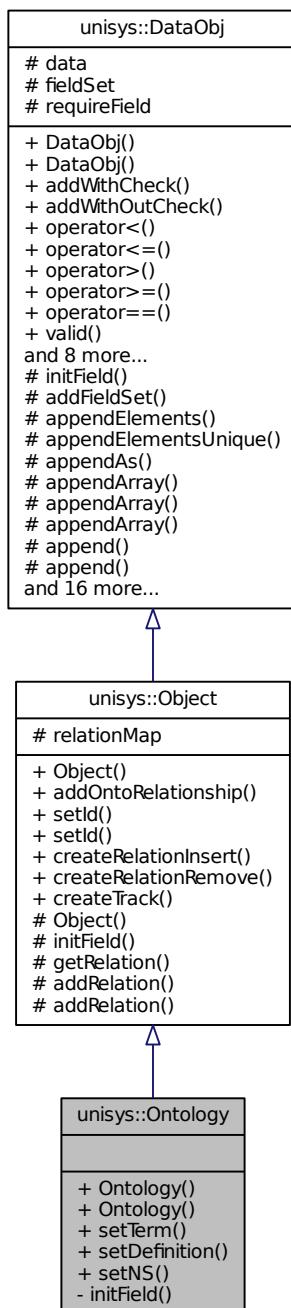
- `DBClass.h`

9.37 unisys::Ontology Class Reference

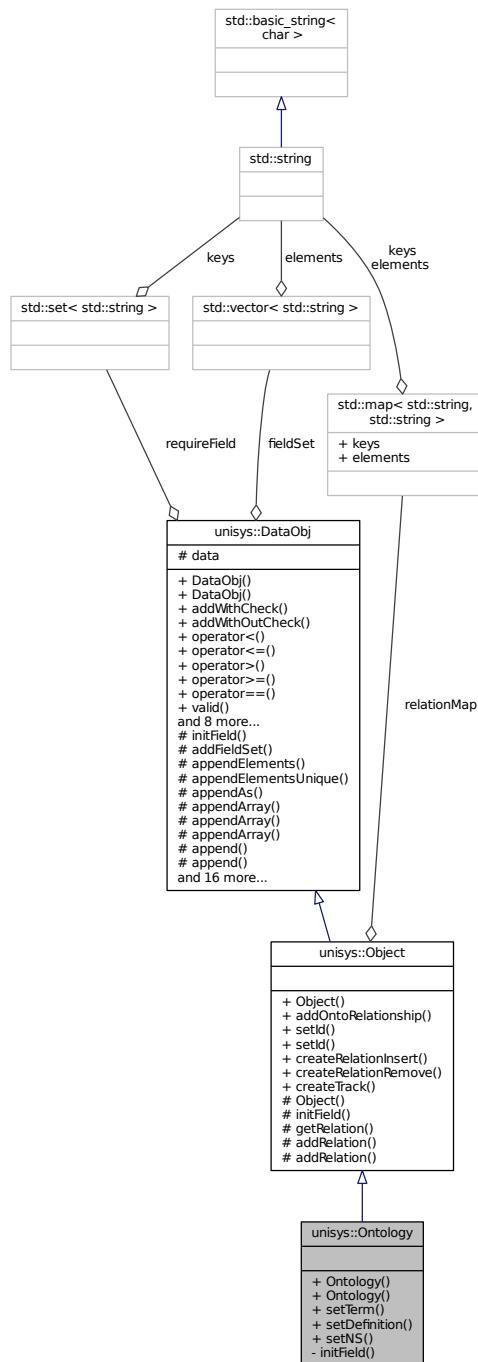
[Ontology](#) data class.

```
#include <ObjClass.h>
```

Inheritance diagram for unisys::Ontology:



Collaboration diagram for unisys::Ontology:



Public Member Functions

- [Ontology \(\)](#)
Default constructor.
- [Ontology \(mongo::BSONObj const &bsonObj\)](#)
Overloaded constructor is used when retrieving data in bson object from database and transform to C++ object.
- [void setTerm \(std::string const &term\)](#)

- void `setDefinition` (std::string const &definition)
- void `setNS` (std::string const &ns)

Private Member Functions

- void `initField` ()

Additional Inherited Members

9.37.1 Detailed Description

[Ontology](#) data class.

```
 BSON structure:
{
    _id: <string>, #mandatory
    ontologyRelationship: {<RelationshipBOSON>, <RelationshipBOSON>, ...}
    term: <string>,
    definition: <string>
}
```

9.37.2 Constructor & Destructor Documentation

9.37.2.1 `unisys::Ontology::Ontology()`

Default constructor.

9.37.2.2 `unisys::Ontology::Ontology(mongo::BSONObj const & bsonObj)`

Overloaded constructor is used when retrieving data in boson object from database and transform to C++ object.

9.37.3 Member Function Documentation

9.37.3.1 `void unisys::Ontology::initField() [private], [virtual]`

Reimplemented from [unisys::Object](#).

9.37.3.2 `void unisys::Ontology::setDefinition(std::string const & definition)`

9.37.3.3 `void unisys::Ontology::setNS(std::string const & ns)`

9.37.3.4 `void unisys::Ontology::setTerm(std::string const & term)`

The documentation for this class was generated from the following file:

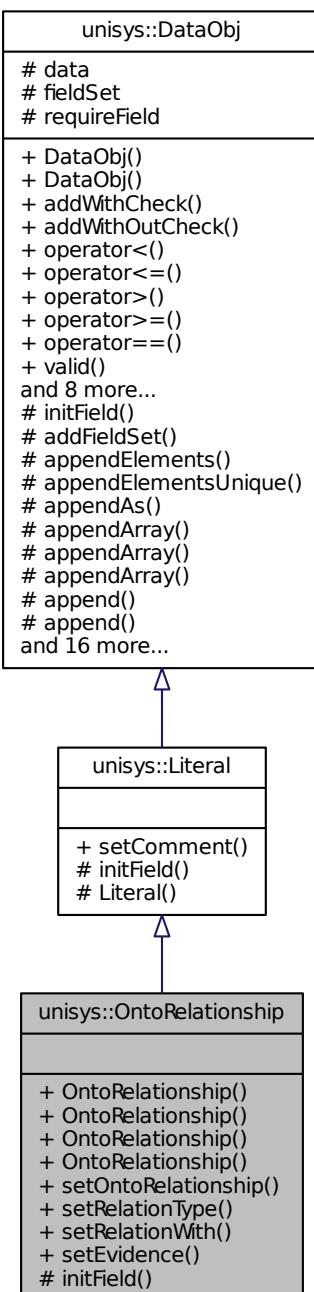
- `ObjClass.h`

9.38 `unisys::OntoRelationship` Class Reference

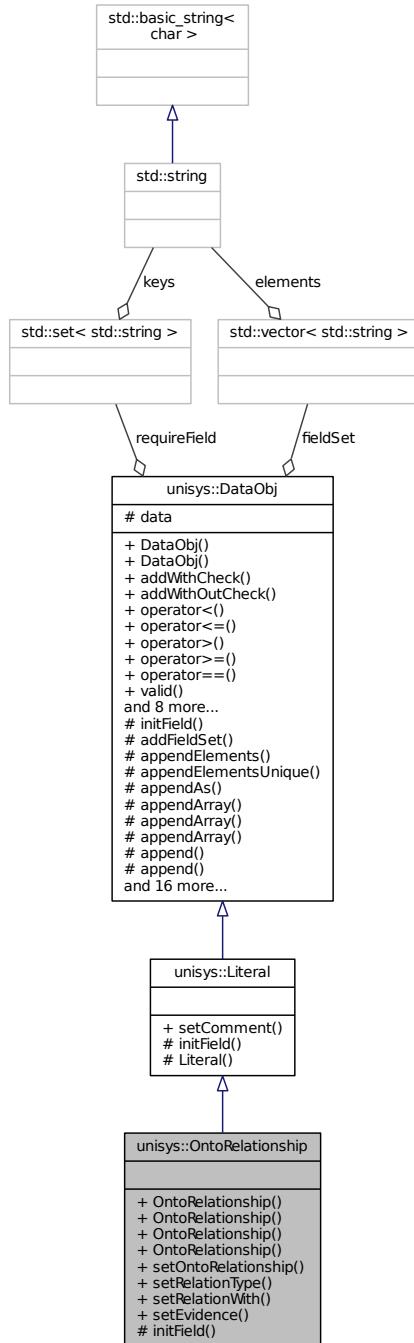
The C++ representative class for relationship data class.

```
#include <LitClass.h>
```

Inheritance diagram for unisys::OntoRelationship:



Collaboration diagram for unisys::OntoRelationship:



Public Member Functions

- [OntoRelationship \(\)](#)
Default constructor.
- [OntoRelationship \(mongo::BSONObj const &bsonObj\)](#)
Overloaded constructor is used when retrieving data in bson object from database and transform to C++ object.
- [OntoRelationship \(std::string const &relationType, std::string const &relationWith\)](#)

- [OntoRelationship](#) (std::string const &relationType, std::string const &relationWith, [Evidence](#) evidence)
- void [setOntoRelationship](#) (std::string const &relationTypeOntold, std::string const &relationWithId, std::string const &targetObjType="physicalentity")
- void [setRelationType](#) ([OntoldRef](#) ontoldRef)
- void [setRelationWith](#) ([IdRef](#) IdRef)
- void [setEvidence](#) ([Evidence](#) &evidence)

Protected Member Functions

- void [initField](#) ()

function for init field in the object

9.38.1 Detailed Description

The C++ representative class for relationship data class.

```
BSON structure:
{
    comment: <string>,
    relationType: {$ref: <collname>, $id: <idvalue>}, #format by idref class
    relationWith: {$ref: <collname>, $id: <idvalue>}, #format by idref class
    evidence: <EvidenceBSONStructure>
}
```

9.38.2 Constructor & Destructor Documentation

9.38.2.1 [unisys::OntoRelationship::OntoRelationship\(\)](#)

Default constructor.

9.38.2.2 [unisys::OntoRelationship::OntoRelationship\(mongo::BSONObj const & bsonObj \)](#)

Overloaded constructor is used when retrieving data in boson object from database and transform to C++ object.

9.38.2.3 [unisys::OntoRelationship::OntoRelationship\(std::string const & relationType, std::string const & relationWith \)](#)

9.38.2.4 [unisys::OntoRelationship::OntoRelationship\(std::string const & relationType, std::string const & relationWith, Evidence evidence \)](#)

9.38.3 Member Function Documentation

9.38.3.1 [void unisys::OntoRelationship::initField\(\) \[protected\], \[virtual\]](#)

function for init field in the object

Reimplemented from [unisys::Literal](#).

9.38.3.2 [void unisys::OntoRelationship::setEvidence\(Evidence & evidence \)](#)

9.38.3.3 [void unisys::OntoRelationship::setOntoRelationship\(std::string const & relationTypeOntold, std::string const & relationWithId, std::string const & targetObjType = "physicalentity" \)](#)

9.38.3.4 [void unisys::OntoRelationship::setRelationType\(OntoldRef ontoldRef \)](#)

9.38.3.5 void unisys::OntoRelationship::setRelationWith (IdRef IdRef)

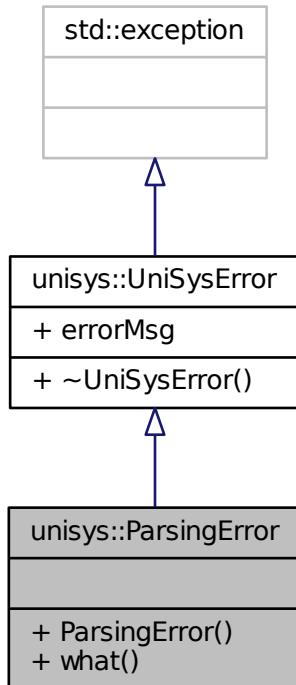
The documentation for this class was generated from the following file:

- [LitClass.h](#)

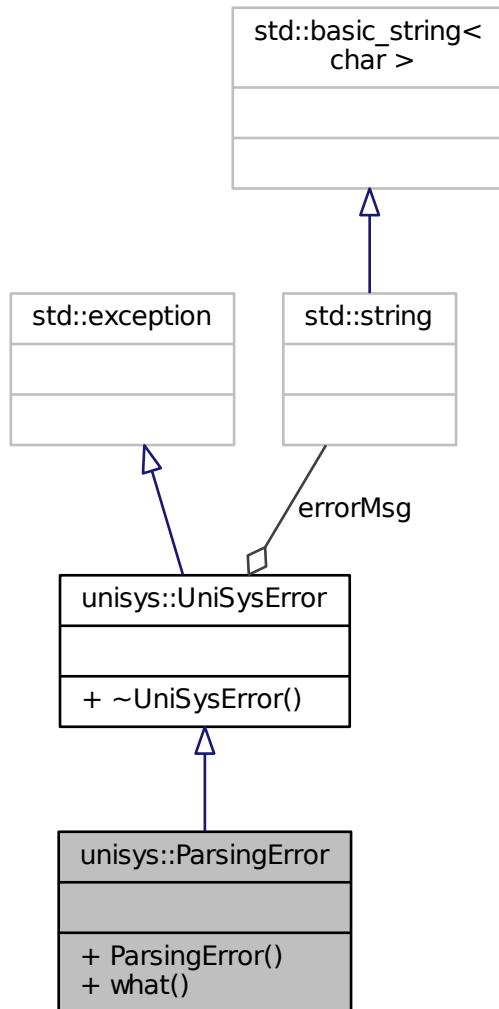
9.39 unisys::ParsingError Class Reference

```
#include <exception.h>
```

Inheritance diagram for unisys::ParsingError:



Collaboration diagram for unisys::ParsingError:



Public Member Functions

- [ParsingError](#) (std::string const &str)
- virtual const char * [what](#) () const throw ()

Additional Inherited Members

9.39.1 Constructor & Destructor Documentation

9.39.1.1 [unisys::ParsingError::ParsingError \(std::string const & str \) \[inline\]](#)

9.39.2 Member Function Documentation

9.39.2.1 virtual const char* unisys::ParsingError::what() const throw() [inline], [virtual]

The documentation for this class was generated from the following file:

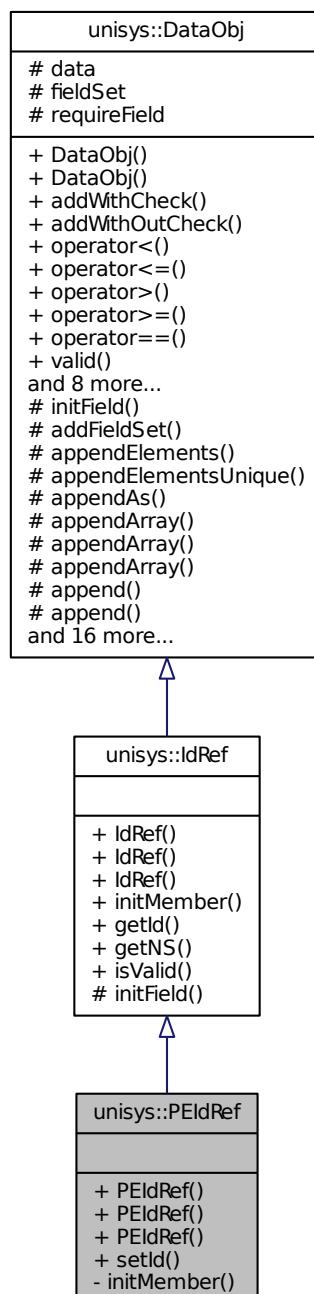
- [exception.h](#)

9.40 unisys::PEIdRef Class Reference

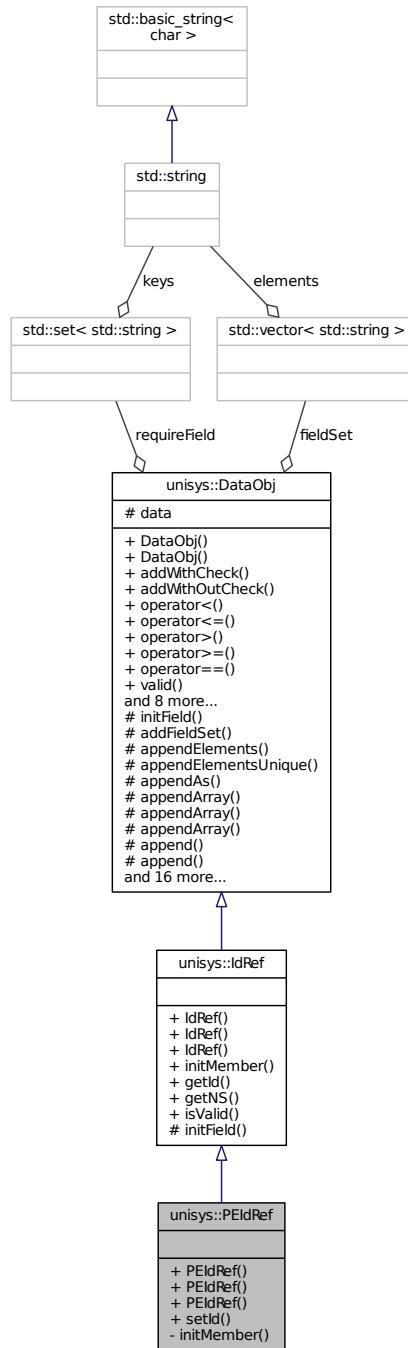
Identifier reference class specific to physicalentity collection namespace.

```
#include <DBClass.h>
```

Inheritance diagram for unisys::PEIdRef:



Collaboration diagram for unisys::PEIdRef:



Public Member Functions

- `PEIdRef ()`
Default constructor.
- `PEIdRef (mongo::BSONObj const &bsonObj)`
- `PEIdRef (std::string const &DBId)`

Overloaded constructor get a object id string as parameter.

- void `setId` (std::string const &DBId)

Private Member Functions

- void `initMember` ()

Additional Inherited Members

9.40.1 Detailed Description

Identifier reference class specific to physicalentity collection namespace.

This class is the same as `IdRef`, but has specific collection namespace to physicalentity collection namespace.

9.40.2 Constructor & Destructor Documentation

9.40.2.1 unisys::PEIdRef::PEIdRef() [inline]

Default constructor.

9.40.2.2 unisys::PEIdRef::PEIdRef(mongo::BSONObj const & bsonObj) [inline]

9.40.2.3 unisys::PEIdRef::PEIdRef(std::string const & DBId) [inline]

Overloaded constructor get a object id string as parameter.

9.40.3 Member Function Documentation

9.40.3.1 void unisys::PEIdRef::initMember() [inline], [private]

9.40.3.2 void unisys::PEIdRef::setId(std::string const & DBId) [inline]

The documentation for this class was generated from the following file:

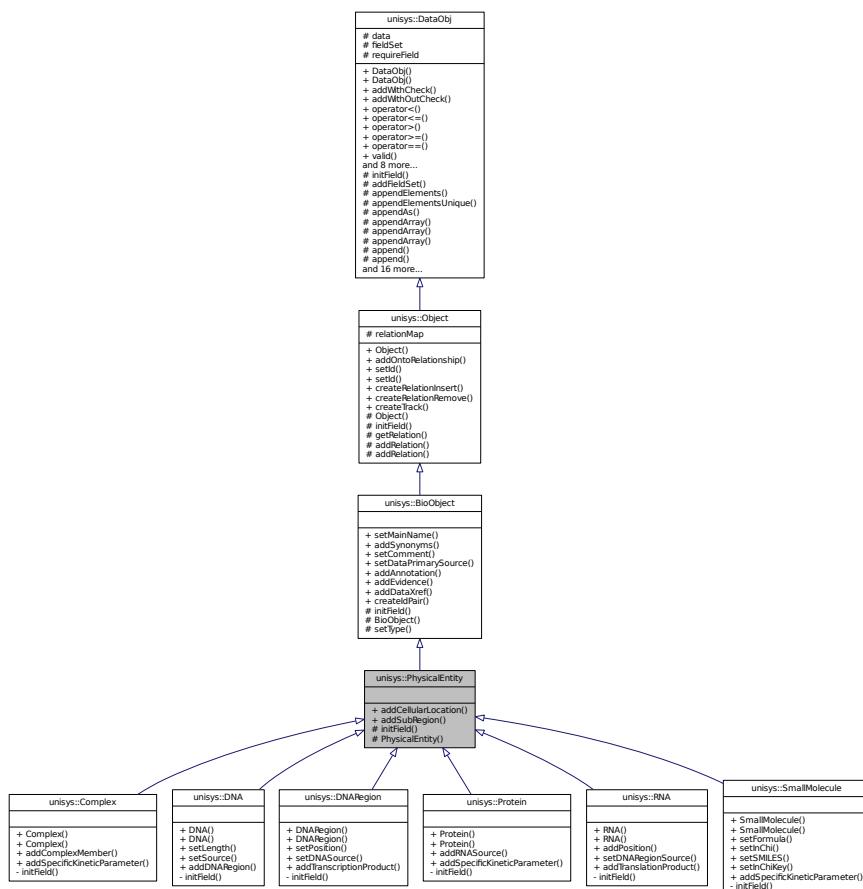
- `DBClass.h`

9.41 unisys::PhysicalEntity Class Reference

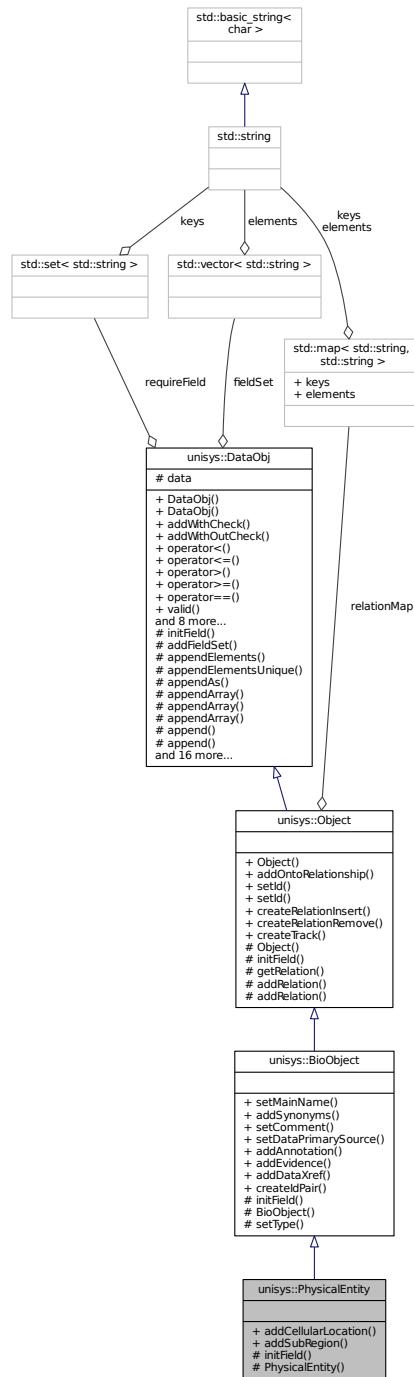
This class is for miriam cross reference annotation.

```
#include <ObjClass.h>
```

Inheritance diagram for unisys::PhysicalEntity:



Collaboration diagram for unisys::PhysicalEntity:



Public Member Functions

- void `addCellularLocation` (`CellularLocation` &`cellularLocation`)
to add
- void `addSubRegion` (`SubRegion` &`subRegion`)
To add sub region.

Protected Member Functions

- void [initField \(\)](#)
- [PhysicalEntity \(\)](#)

9.41.1 Detailed Description

This class is for miriam cross reference annotation.

```
 BSON structure:
{
    _id: <string>, #mandatory
    ontologyRelationship: {<RelationshipBOSON>, <RelationshipBOSON>, ...},
    name: {<string>, <string>, ...}
    comment: <string>,
    dataPrimarySource: <XrefBOSON>,
    functionAnnotation: {<AnnotationBOSON>, <AnnotationBOSON>, ...},
    evidence: {<EvidenceBOSON>, <EvidenceBOSON>, ...},
    dataxref: {<XrefBOSON>, <XrefBOSON>, ...},
    interaction: {<DbRef>, <DbRef>, ...},
    complex: {<DbRef>, <DbRef>, ...},
    subRegion: {<SubRegionBOSON>, <SubRegionBOSON>, ...}
}
```

9.41.2 Constructor & Destructor Documentation

9.41.2.1 `unisys::PhysicalEntity::PhysicalEntity()` [protected]

9.41.3 Member Function Documentation

9.41.3.1 `void unisys::PhysicalEntity::addCellularLocation(CellularLocation & cellularLocation)`

to add

9.41.3.2 `void unisys::PhysicalEntity::addSubRegion(SubRegion & subRegion)`

To add sub region.

9.41.3.3 `void unisys::PhysicalEntity::initField()` [protected], [virtual]

Reimplemented from [unisys::BioObject](#).

Reimplemented in [unisys::Complex](#), [unisys::Protein](#), [unisys::RNA](#), [unisys::DNARRegion](#), [unisys::DNA](#), and [unisys::SmallMolecule](#).

The documentation for this class was generated from the following file:

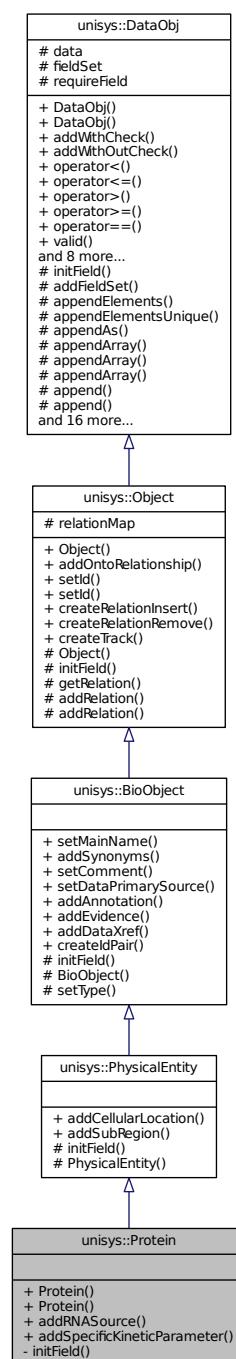
- [ObjClass.h](#)

9.42 unisys::Protein Class Reference

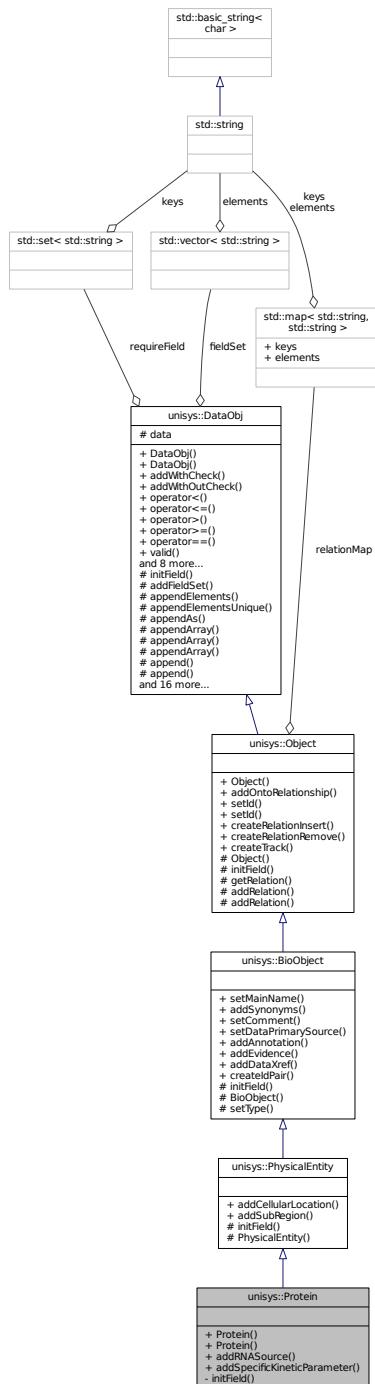
This class is for miriam cross reference annotation.

```
#include <ObjClass.h>
```

Inheritance diagram for unisys::Protein:



Collaboration diagram for unisys::Protein:



Public Member Functions

- **Protein ()**
Default constructor.
- **Protein (mongo::BSONObj const &bsonObj)**
Overloaded constructor is used when retrieving data in bson object from database and transform to C++ object.
- void **addRNAsource (PEIdRef &peldRef)**

- void [addSpecificKineticParameter \(KineticParameter &kineticParameter\)](#)

Private Member Functions

- void [initField \(\)](#)

Additional Inherited Members

9.42.1 Detailed Description

This class is for miriam cross reference annotation.

```
 BSON structure:
{
    _id: <string>, #mandatory
    ontologyRelationship: {<RelationshipBOSON>, <RelationshipBOSON>, ...},
    name: {<string>, <string>, ...}
    comment: <string>,
    dataPrimarySource: <XrefBOSON>,
    functionAnnotation: {<AnnotationBOSON>, <AnnotationBOSON>, ...},
    evidence: {<EvidenceBOSON>, <EvidenceBOSON>, ...},
    dataxref: {<XrefBOSON>, <XrefBOSON>, ...},
    interaction: {<DbRef>, <DbRef>, ...},
    complex: {<DbRef>, <DbRef>, ...},
    subRegion: {<SubRegionBOSON>, <SubRegionBOSON>, ...},
    rnaSource: {<DbRef>, <DbRef>, ...},
    specificKineticParameter: {<KineticParameterBOSON>, <KineticParameterBOSON>, ...}
}
```

9.42.2 Constructor & Destructor Documentation

9.42.2.1 unisys::Protein::Protein ()

Default constructor.

9.42.2.2 unisys::Protein::Protein (mongo::BSONObj const & bsonObj)

Overloaded constructor is used when retrieving data in boson object from database and transform to C++ object.

9.42.3 Member Function Documentation

9.42.3.1 void unisys::Protein::addRNASource (PEIdRef & peldRef)

9.42.3.2 void unisys::Protein::addSpecificKineticParameter (KineticParameter & kineticParameter)

9.42.3.3 void unisys::Protein::initField () [private], [virtual]

Reimplemented from [unisys::PhysicalEntity](#).

The documentation for this class was generated from the following file:

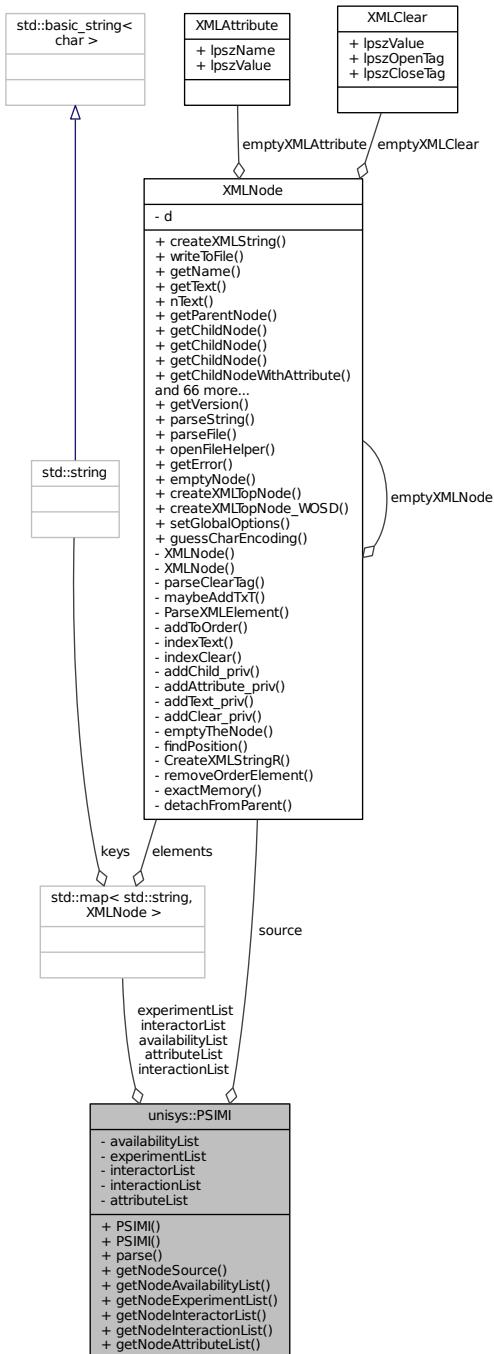
- [ObjClass.h](#)

9.43 unisys::PSIMI Class Reference

A [PSIMI](#) class is used for a data in PSI-MI XML2.5 file format.

```
#include <psimi.h>
```

Collaboration diagram for unisys::PSIMI:



Public Types

- `typedef std::map< std::string, XMLNode > mapOfXMLNode`

A map container contains list of XML data.

- `typedef std::map< std::string, std::string > mapOfStringString`

A map container contains data in std::string.

Public Member Functions

- **PSIMI ()**
Class constructor for non-parameter passing.
- **PSIMI (std::string const &filename)**
Class constructor overloaded with PSI-MI XML2.5 file as an input.
- `void parse (std::string const &filename) throw (ParsingError)`
Parse a PSI-MI XML2.5 input file.
- **XMLNode getNodeSource () const**
Get 'source' node.
- **mapOfXMLNode getNodeAvailabilityList () const**
Get 'availabilityList' node.
- **mapOfXMLNode getNodeExperimentList () const**
Get 'experimentList' node.
- **mapOfXMLNode getNodeInteractorList () const**
Get 'interactorList' node.
- **mapOfXMLNode getNodeInteractionList () const**
Get 'interactionList' node.
- **mapOfXMLNode getNodeAttributeList () const**
Get 'attributeList' node.

Private Attributes

- **XMLNode source**
an XMLNode object 'source'.
- **mapOfXMLNode availabilityList**
A map container 'availabilityList', containing 0-multiple 'availability' elements.
- **mapOfXMLNode experimentList**
A map container 'experimentList', containing 0-multiple 'experimentDescription' elements.
- **mapOfXMLNode interactorList**
A map container 'interactorList', containing 0-multiple 'interactor' elements.
- **mapOfXMLNode interactionList**
A map container 'interactionList', containing 1-multiple 'interaction' elements.
- **mapOfXMLNode attributeList**
A map container 'attributeList', containing 0-multiple 'attribute' elements.

9.43.1 Detailed Description

A **PSIMI** class is used for a data in PSI-MI XML2.5 file format.

A **PSIMI** object contains an 'entry' element of the root element 'entrySet' of PSI-MI XML2.5 file. Functions are implemented for parsing data in PSI-MI XML format to UniSysDB.

9.43.2 Member Typedef Documentation

9.43.2.1 `typedef std::map< std::string, std::string > unisys::PSIMI::mapOfStringString`

A map container contains data in std::string.

Data is kept in following structure: map<key,value> where

Parameters

<code>key</code>	= a key value and
<code>value</code>	= a mapped value

9.43.2.2 `typedef std::map< std::string, XMLNode > unisys::PSIMI::mapOfXMLNode`

A map container contains list of XML data.

`XMLNode` [name]List contains list of sub-nodes [name] These nodes [name]List are parsed to following structure: map<key, XMLNode> where

Parameters

<code>key</code>	= value of attribute 'id' (of node 'availability', 'experimentDescription', 'interactor', 'interaction') or value of attribute 'name' (of node 'attribute') and
<code>XMLNode</code>	= sub-node [name]

9.43.3 Constructor & Destructor Documentation

9.43.3.1 `unisys::PSIMI::PSIMI()`

Class constructor for non-parameter passing.

Instantiate a `PSIMI` object.

9.43.3.2 `unisys::PSIMI::PSIMI(std::string const & filename)`

Class constructor overloaded with PSI-MI XML2.5 file as an input.

Instantiate a `PSIMI` object using the given PSI-MI XML2.5 file.

Parameters

<code>fileName</code>	is the path to the PSI-MI XML2.5 file.
-----------------------	--

9.43.4 Member Function Documentation

9.43.4.1 `mapOfXMLNode unisys::PSIMI::getNodeAttributeList() const`

Get 'attributeList' node.

Return a map container of 'attribute' elements.

9.43.4.2 `mapOfXMLNode unisys::PSIMI::getNodeAvailabilityList() const`

Get 'availabilityList' node.

Return a map container of 'availability' elements.

9.43.4.3 mapOfXMLNode unisys::PSIMI::getNodeExperimentList() const

Get 'experimentList' node.

Return a map container of 'experimentDescription' elements.

9.43.4.4 mapOfXMLNode unisys::PSIMI::getNodeInteractionList() const

Get 'interactionList' node.

Return a map container of 'interaction' elements.

9.43.4.5 mapOfXMLNode unisys::PSIMI::getNodeInteractorList() const

Get 'interactorList' node.

Return a map container of 'interactor' elements.

9.43.4.6 XMLNode unisys::PSIMI::getNodeSource() const

Get 'source' node.

Return 'source' node.

9.43.4.7 void unisys::PSIMI::parse(std::string const & filename) throw (ParsingError)

Parse a PSI-MI XML2.5 input file.

Parameters

<i>fileName</i>	is the path to the PSI-MI XML2.5 file.
-----------------	--

9.43.5 Member Data Documentation**9.43.5.1 mapOfXMLNode unisys::PSIMI::attributeList [private]**

A map container 'attributeList', containing 0-multiple 'attribute' elements.

9.43.5.2 mapOfXMLNode unisys::PSIMI::availabilityList [private]

A map container 'availabilityList', containing 0-multiple 'availability' elements.

9.43.5.3 mapOfXMLNode unisys::PSIMI::experimentList [private]

A map container 'experimentList', containing 0-multiple 'experimentDescription' elements.

9.43.5.4 mapOfXMLNode unisys::PSIMI::interactionList [private]

A map container 'interactionList', containing 1-multiple 'interaction' elements.

9.43.5.5 mapOfXMLNode unisys::PSIMI::interactorList [private]

A map container 'interactorList', containing 0-multiple 'interactor' elements.

9.43.5.6 XMLNode unisys::PSIMI::source [private]

an [XMLNode](#) object 'source'.

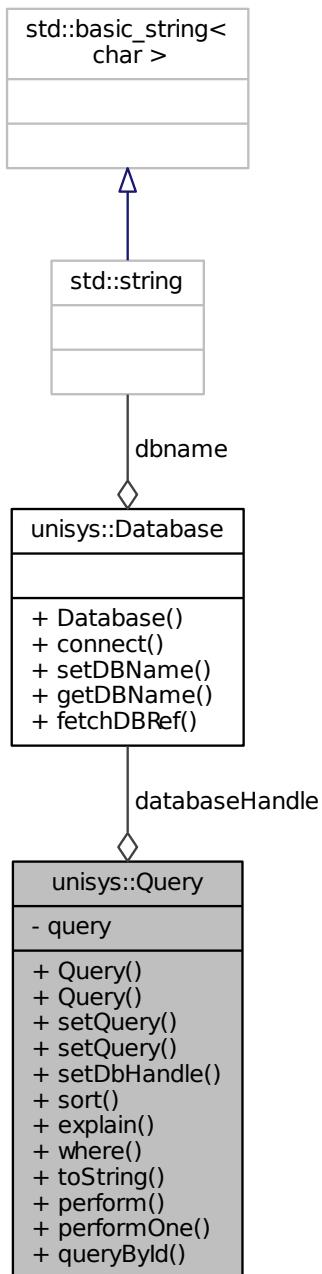
The documentation for this class was generated from the following file:

- [psimi.h](#)

9.44 unisys::Query Class Reference

```
#include <query.h>
```

Collaboration diagram for unisys::Query:



Public Member Functions

- [Query \(\)](#)
- [Query \(Database *databaseHandlePt\)](#)
- [Query & setQuery \(mongo::Query const &query\)](#)
- [Query & setQuery \(mongo::BSONObj const &query\)](#)
- [Query & setDbHandle \(Database *databaseHandlePt\)](#)

- [Query & sort](#) (std::string const &field, int asc=1)
- [Query & explain](#) ()
- [Query & where](#) (const std::string &jscode)
- std::string [toString](#) () const
- mongo::BSONObj [perform](#) (const std::string &ns, int nToReturn=0, int nToSkip=0, const mongo::BSONObj *fieldsToReturn=0, int queryOptions=0, int batchSize=0) throw (QueryError)
- mongo::BSONObj [performOne](#) (const std::string &ns, int nToReturn=0, int nToSkip=0, const mongo::BSONObj *fieldsToReturn=0, int queryOptions=0, int batchSize=0) throw (QueryError)
- mongo::BSONObj [queryByld](#) (const std::string &ns, const std::string &id) throw (QueryError)

Private Attributes

- Database * [databaseHandle](#)
- mongo::Query [query](#)

9.44.1 Constructor & Destructor Documentation

9.44.1.1 unisys::Query::Query ()

9.44.1.2 unisys::Query::Query (Database * *databaseHandlePt*)

9.44.2 Member Function Documentation

9.44.2.1 Query& unisys::Query::explain ()

9.44.2.2 mongo::BSONObj unisys::Query::perform (const std::string & ns, int *nToReturn* = 0, int *nToSkip* = 0, const mongo::BSONObj * *fieldsToReturn* = 0, int *queryOptions* = 0, int *batchSize* = 0) throw (QueryError)

9.44.2.3 mongo::BSONObj unisys::Query::performOne (const std::string & ns, int *nToReturn* = 0, int *nToSkip* = 0, const mongo::BSONObj * *fieldsToReturn* = 0, int *queryOptions* = 0, int *batchSize* = 0) throw (QueryError)

9.44.2.4 mongo::BSONObj unisys::Query::queryByld (const std::string & ns, const std::string & id) throw (QueryError)

9.44.2.5 Query& unisys::Query::setDbHandle (Database * *databaseHandlePt*)

9.44.2.6 Query& unisys::Query::setQuery (mongo::Query const & *query*)

9.44.2.7 Query& unisys::Query::setQuery (mongo::BSONObj const & *query*)

9.44.2.8 Query& unisys::Query::sort (std::string const & *field*, int *asc* = 1)

9.44.2.9 std::string unisys::Query::toString () const

9.44.2.10 Query& unisys::Query::where (const std::string & *jscode*)

9.44.3 Member Data Documentation

9.44.3.1 Database* unisys::Query::databaseHandle [private]

9.44.3.2 mongo::Query unisys::Query::query [private]

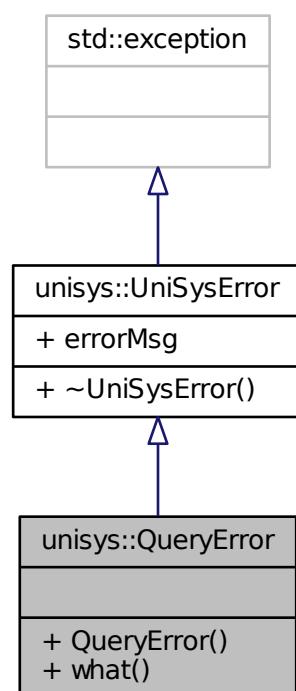
The documentation for this class was generated from the following file:

- [query.h](#)

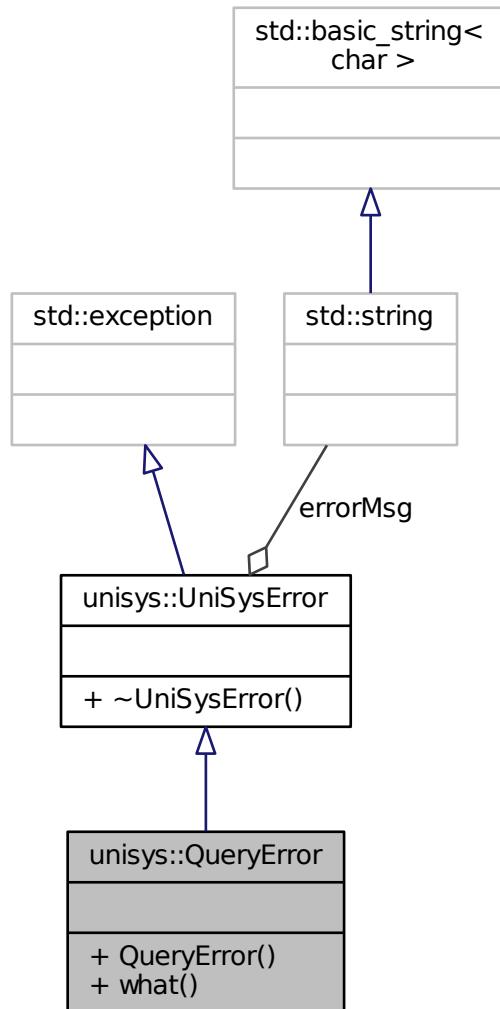
9.45 unisys::QueryError Class Reference

```
#include <exception.h>
```

Inheritance diagram for unisys::QueryError:



Collaboration diagram for unisys::QueryError:



Public Member Functions

- `QueryError (std::string const &str)`
- `virtual const char * what () const throw ()`

Additional Inherited Members

9.45.1 Constructor & Destructor Documentation

9.45.1.1 `unisys::QueryError::QueryError (std::string const & str) [inline]`

9.45.2 Member Function Documentation

9.45.2.1 virtual const char* unisys::QueryError::what() const throw() [inline], [virtual]

The documentation for this class was generated from the following file:

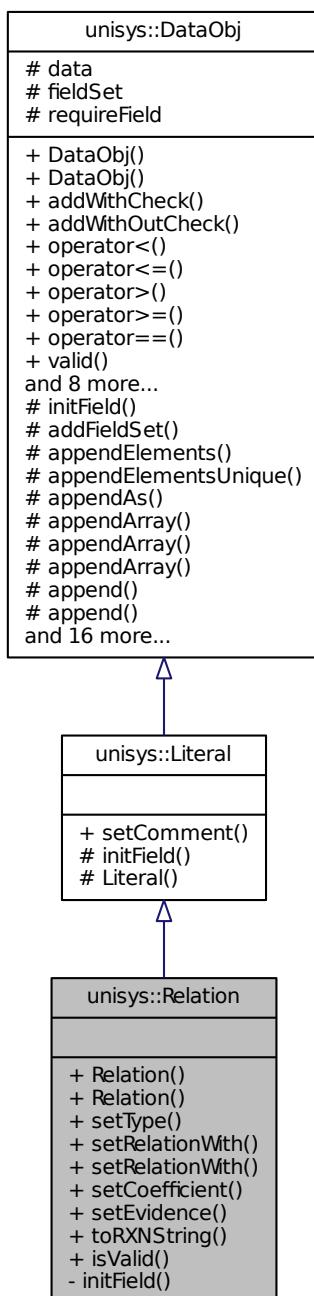
- [exception.h](#)

9.46 unisys::Relation Class Reference

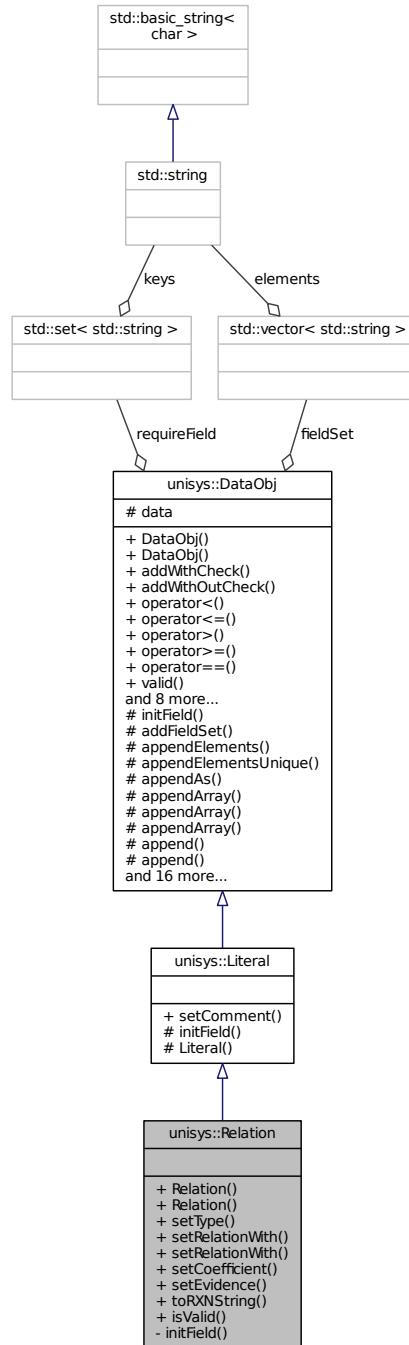
The C++ representative class for relationship data class.

```
#include <LitClass.h>
```

Inheritance diagram for unisys::Relation:



Collaboration diagram for unisys::Relation:



Public Member Functions

- [Relation \(\)](#)
 - [Relation \(mongo::BSONObj const &bsonObj\)](#)
 - void [setType \(std::string const &string\)](#)
 - void [setRelationWith \(IdRef &IdRef\)](#)
 - void [setRelationWith \(std::string const &IdRefId, std::string const &targetObjType="physicalentity"\)](#)

- void `setCoefficient` (double coefficient)
- void `setEvidence` (Evidence &evidence)
- std::string `toRXNString` (std::string const &compartment) const
- bool `isValid` ()

Private Member Functions

- void `initField` ()

function for init field in the object

Additional Inherited Members

9.46.1 Detailed Description

The C++ representative class for relationship data class.

```
 BSON structure:
{
    comment: <string>,
    type: <string>, #format by idref class
    coefficient: <float>,
    relationWith: {$ref: <collname>, $id: <idvalue>}, #format by idref class
    evidence: <EvidenceBSONStructure>
}
```

9.46.2 Constructor & Destructor Documentation

9.46.2.1 `unisys::Relation::Relation()`

9.46.2.2 `unisys::Relation::Relation(mongo::BSONObj const & bsonObj)`

9.46.3 Member Function Documentation

9.46.3.1 `void unisys::Relation::initField()` [private], [virtual]

function for init field in the object

Reimplemented from `unisys::Literal`.

9.46.3.2 `bool unisys::Relation::isValid()`

9.46.3.3 `void unisys::Relation::setCoefficient(double coefficient)`

9.46.3.4 `void unisys::Relation::setEvidence(Evidence & evidence)`

9.46.3.5 `void unisys::Relation::setRelationWith(IdRef & IdRef)`

9.46.3.6 `void unisys::Relation::setRelationWith(std::string const & IdRefId, std::string const & targetObjType = "physicalentity")`

9.46.3.7 `void unisys::Relation::setType(std::string const & string)`

9.46.3.8 `std::string unisys::Relation::toRXNString(std::string const & compartment) const`

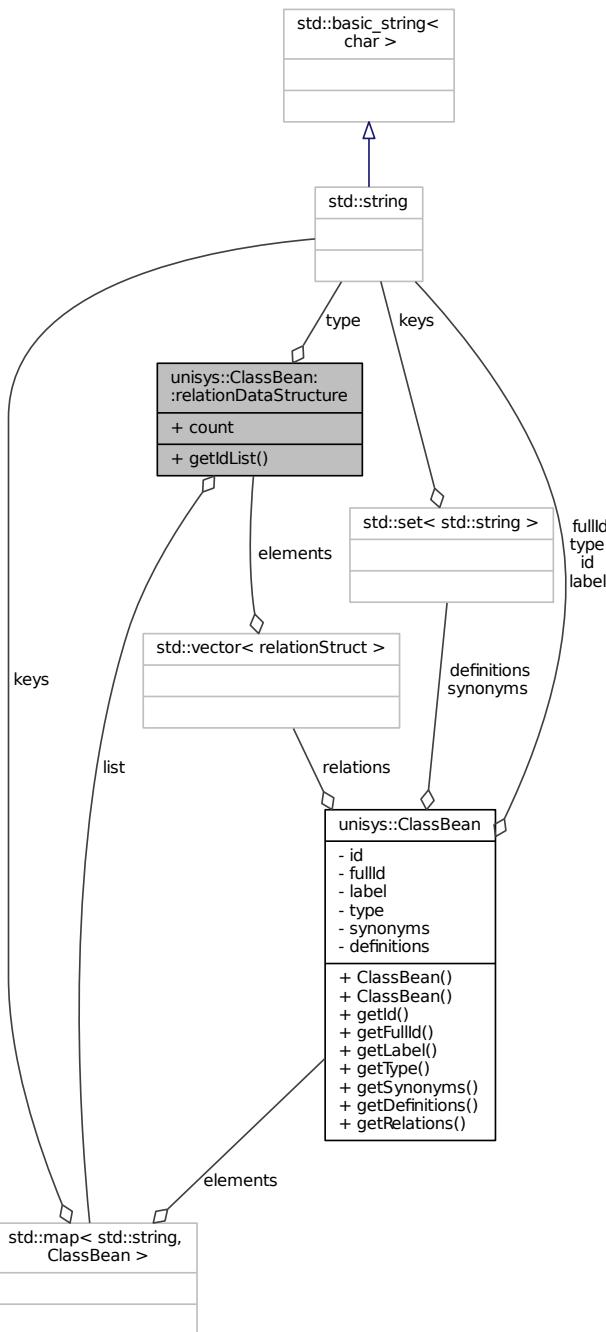
The documentation for this class was generated from the following file:

- [LitClass.h](#)

9.47 unisys::ClassBean::relationDataStructure Struct Reference

```
#include <restBioPortal.h>
```

Collaboration diagram for unisys::ClassBean::relationDataStructure:



Public Member Functions

- std::set< std::string > [getIdList \(\)](#)

Return a list of element id.

Public Attributes

- std::string [type](#)

Type of relation.

- int [count](#)

Number of element that related with the relation.

- std::map< std::string, [ClassBean](#) > [list](#)

List of element.

9.47.1 Member Function Documentation

9.47.1.1 std::set<std::string> [unisys::ClassBean::relationDataStructure::getIdList \(\)](#)

Return a list of element id.

9.47.2 Member Data Documentation

9.47.2.1 int [unisys::ClassBean::relationDataStructure::count](#)

Number of element that related with the relation.

9.47.2.2 std::map<std::string, [ClassBean](#)> [unisys::ClassBean::relationDataStructure::list](#)

List of element.

9.47.2.3 std::string [unisys::ClassBean::relationDataStructure::type](#)

Type of relation.

The documentation for this struct was generated from the following file:

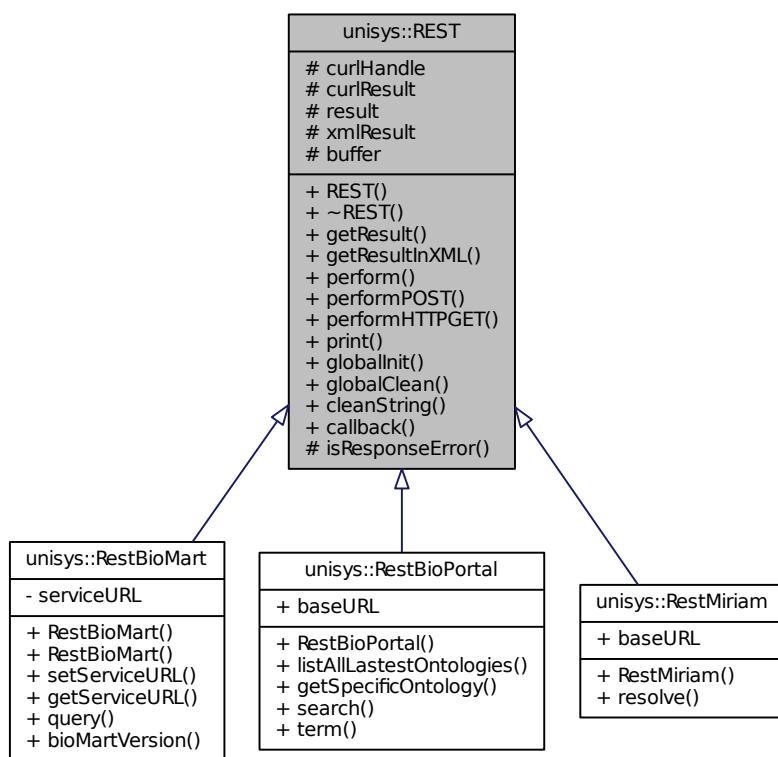
- [restBioPortal.h](#)

9.48 unisys::REST Class Reference

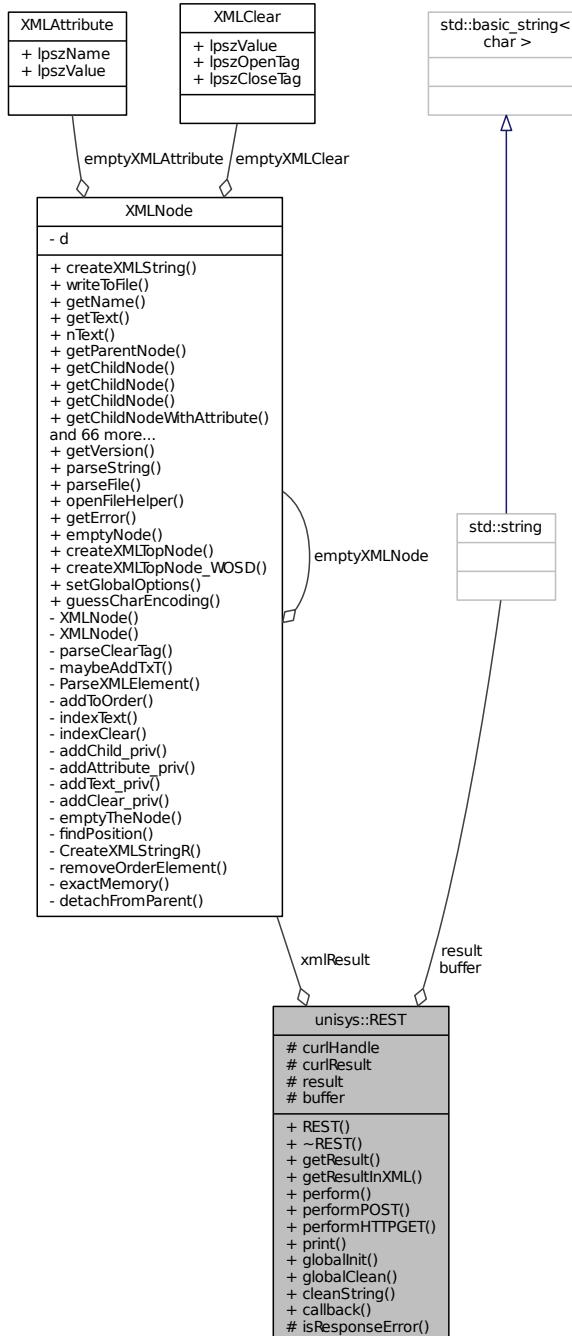
Used for managing the request and response of [REST](#) service.

```
#include <rest.h>
```

Inheritance diagram for unisys::REST:



Collaboration diagram for unisys::REST:



Public Member Functions

- [REST \(\)](#)

Default constructor. Do the curl_easy_init and set some option to cURL library handler.

- [~REST \(\)](#)

Default destructor. Do the curl_easy_clean and release the memory of cURL library handler.

- [std::string getResult \(\)](#)

- `XMLNode getResultInXML () throw (ParsingError)`
- `void perform (std::string const &url) throw (RESTServiceError)`
Perform the request with HEAD method to the service and take the response back.
- `void performPOST (std::string const &url, std::string const &postFields) throw (RESTServiceError)`
Perform the request with POST method to the service and take the response back.
- `void performHTTPGET (std::string const &url) throw (RESTServiceError)`
Perform the request with HTTPGET method to the service and take the response back.
- `void print ()`
Print out the value in `REST::xmlResult`. (Just for debugging)

Static Public Member Functions

- `static void globalInit ()`
Init function of cURL library. Just do it on time in one program.
- `static void globalClean ()`
Setting destructor of cURL library. Just do it on time in one program.
- `static std::string cleanString (std::string const &str)`
Used for clean the restrict charactors in string; like space, &, \$, % etc.
- `static size_t callback (void *ptr, size_t size, size_t count, FILE *data)`
static member function for processing results from libcurl

Protected Member Functions

- `bool isResponseError () throw (RESTServiceError)`

Protected Attributes

- `CURL * curlHandle`
cURL library handler
- `CURLcode curlResult`
cURL library response handler
- `std::string result`
The results of `REST` service in string.
- `XMLNode xmlResult`
The results of `REST` service in string.

Static Protected Attributes

- `static std::string buffer`
Static member function for getting results from static callback member function.

9.48.1 Detailed Description

Used for managing the request and response of `REST` service.

This class connects to `REST` service through the cURL library, but it has very limit capability when comparing with cURL library.

9.48.2 Constructor & Destructor Documentation

9.48.2.1 unisys::REST::REST()

Default constructor. Do the curl_easy_init and set some option to cURL library handler.

9.48.2.2 unisys::REST::~REST()

Default destructor. Do the curl_easy_clean and release the memory of cURL library handler.

9.48.3 Member Function Documentation

9.48.3.1 static size_t unisys::REST::callback(void *ptr, size_t size, size_t count, FILE *data) [static]

static member function for processing results from libcurl

9.48.3.2 static std::string unisys::REST::cleanString(std::string const & str) [static]

Used for clean the restrict charactors in string; like space, &, \$, % etc.

9.48.3.3 std::string unisys::REST::getResult()

9.48.3.4 XMLNode unisys::REST::getResultInXML() throw (ParsingError)

9.48.3.5 static void unisys::REST::globalClean() [static]

Setting destructor of cURL library. Just do it on time in one program.

9.48.3.6 static void unisys::REST::globalInit() [static]

Init function of cURL library. Just do it on time in one program.

9.48.3.7 bool unisys::REST::isResponseError() throw (RESTServiceError) [protected]

9.48.3.8 void unisys::REST::perform(std::string const & url) throw (RESTServiceError)

Perform the request with HEAD method to the service and take the response back.

9.48.3.9 void unisys::REST::performHTTPGET(std::string const & url) throw (RESTServiceError)

Perform the request with HTTPGET method to the service and take the response back.

9.48.3.10 void unisys::REST::performPOST(std::string const & url, std::string const & postFields) throw (RESTServiceError)

Perform the request with POST method to the service and take the response back.

9.48.3.11 void unisys::REST::print()

Print out the value in [REST::xmlResult](#). (Just for debugging)

9.48.4 Member Data Documentation

9.48.4.1 std::string unisys::REST::buffer [static], [protected]

Static member function for getting results from static callback member function.

9.48.4.2 CURL* unisys::REST::curlHandle [protected]

cURL library handler

9.48.4.3 CURLcode unisys::REST::curlResult [protected]

cURL library response handler

9.48.4.4 std::string unisys::REST::result [protected]

The results of [REST](#) service in string.

9.48.4.5 XMLNode unisys::REST::xmlResult [protected]

The results of [REST](#) service in string.

The documentation for this class was generated from the following file:

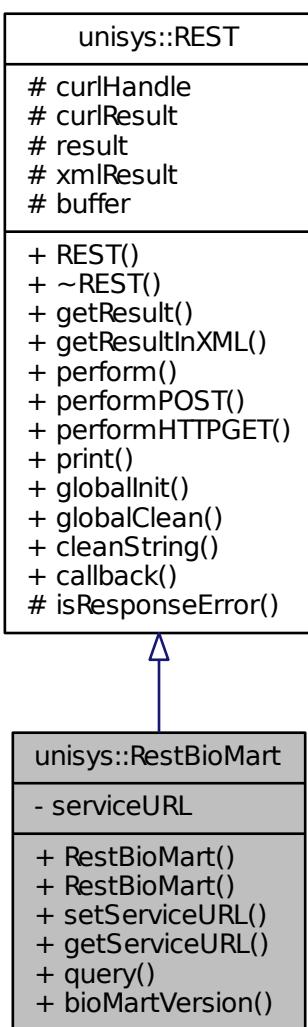
- [rest.h](#)

9.49 unisys::RestBioMart Class Reference

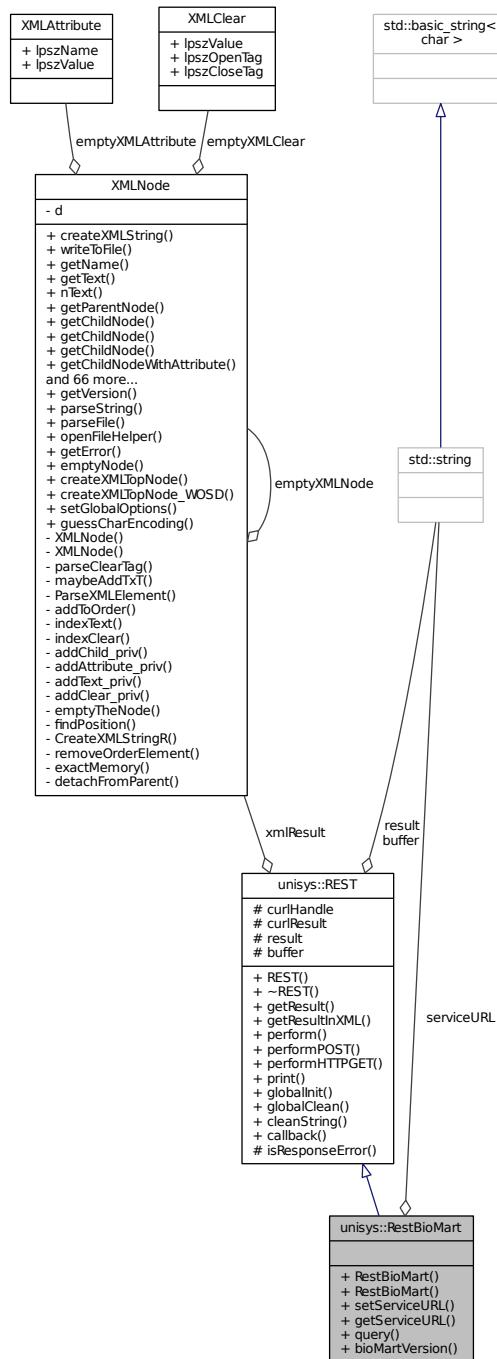
General class for taking care the response from BioPortal's [REST](#) service.

```
#include <restBioMart.h>
```

Inheritance diagram for unisys::RestBioMart:



Collaboration diagram for unisys::RestBioMart:



Public Member Functions

- [RestBioMart \(\)](#)
Default constructor.
- [RestBioMart \(std::string const &URL\)](#)
- [void setServiceURL \(std::string const &URL\)](#)
- [std::string getServiceURL \(\) const](#)

- `XMLNode query (XMLNode queryXML)`

Static Public Member Functions

- `static std::string bioMartVersion ()`

Private Attributes

- `std::string serviceURL`

Additional Inherited Members

9.49.1 Detailed Description

General class for taking care the response from BioPortal's [REST](#) service.

9.49.2 Constructor & Destructor Documentation

9.49.2.1 `unisys::RestBioMart::RestBioMart ()`

Default constructor.

9.49.2.2 `unisys::RestBioMart::RestBioMart (std::string const & URL)`

9.49.3 Member Function Documentation

9.49.3.1 `static std::string unisys::RestBioMart::bioMartVersion () [inline], [static]`

9.49.3.2 `std::string unisys::RestBioMart::getServiceURL () const`

9.49.3.3 `XMLNode unisys::RestBioMart::query (XMLNode queryXML)`

9.49.3.4 `void unisys::RestBioMart::setServiceURL (std::string const & URL)`

9.49.4 Member Data Documentation

9.49.4.1 `std::string unisys::RestBioMart::serviceURL [private]`

The documentation for this class was generated from the following file:

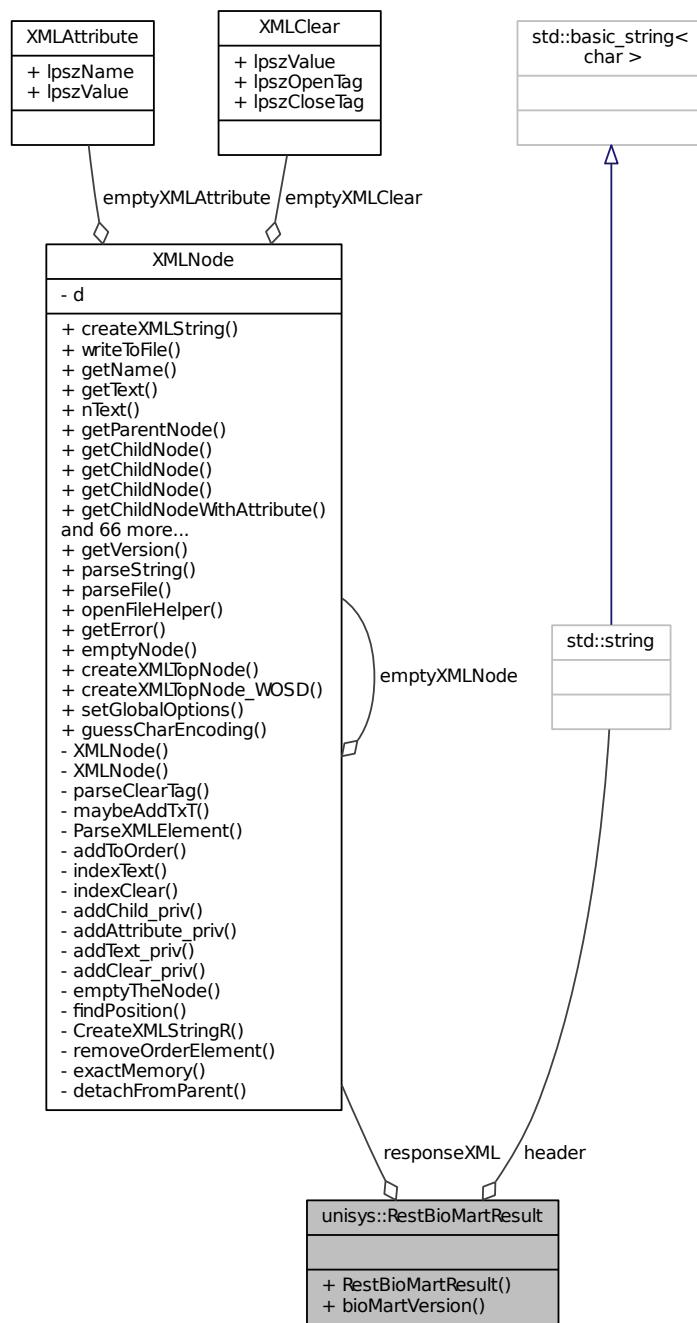
- `restBioMart.h`

9.50 `unisys::RestBioMartResult` Class Reference

General class for taking care the response from BioPortal's [REST](#) service.

```
#include <restBioMart.h>
```

Collaboration diagram for unisys::RestBioMartResult:



Public Member Functions

- [RestBioMartResult \(\)](#)

Static Public Member Functions

- [static std::string bioMartVersion \(\)](#)

Private Attributes

- `XMLNode responseXML`
- `std::string header`

Friends

- class `RestBioMart`

9.50.1 Detailed Description

General class for taking care the response from BioPortal's [REST](#) service.

9.50.2 Constructor & Destructor Documentation

9.50.2.1 `unisys::RestBioMartResult::RestBioMartResult()`

9.50.3 Member Function Documentation

9.50.3.1 `static std::string unisys::RestBioMartResult::bioMartVersion()` [inline], [static]

9.50.4 Friends And Related Function Documentation

9.50.4.1 `friend class RestBioMart` [friend]

9.50.5 Member Data Documentation

9.50.5.1 `std::string unisys::RestBioMartResult::header` [private]

9.50.5.2 `XMLNode unisys::RestBioMartResult::responseXML` [private]

The documentation for this class was generated from the following file:

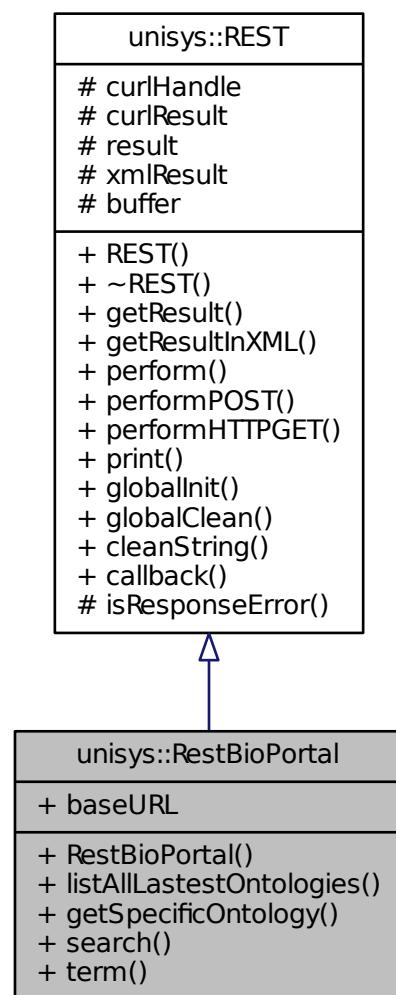
- `restBioMart.h`

9.51 `unisys::RestBioPortal` Class Reference

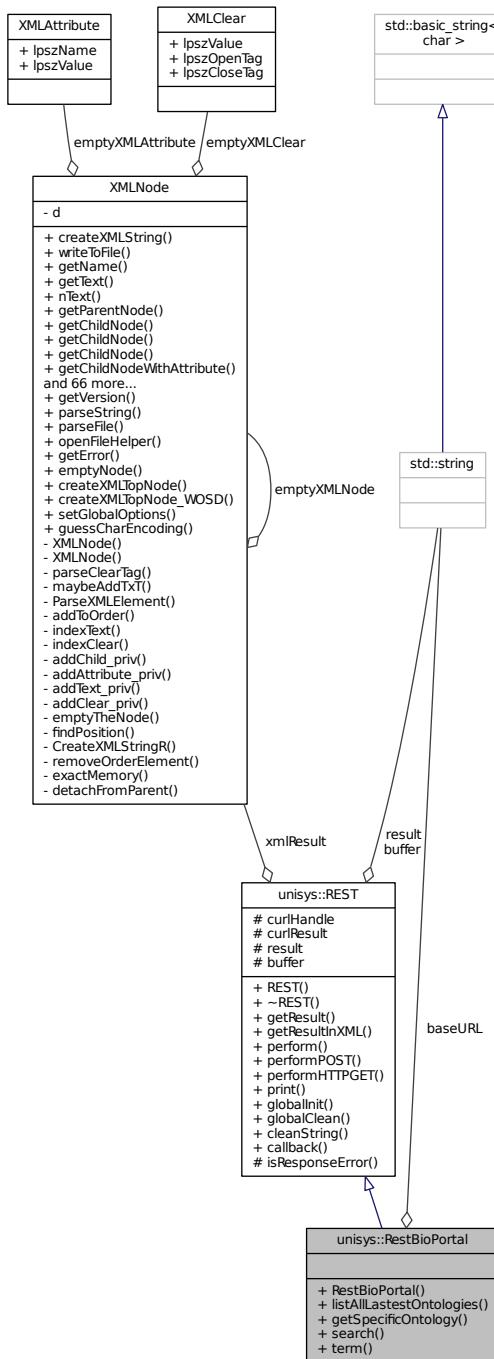
The main class of BioPortal [REST](#) service library.

```
#include <restBioPortal.h>
```

Inheritance diagram for unisys::RestBioPortal:



Collaboration diagram for unisys::RestBioPortal:



Public Member Functions

- **RestBioPortal ()**
Default constructor.
- **XMLNode listAllLatestOntologies (std::string const &apikey)**
List all of the lastest ontology classes and thier details.
- **XMLNode getSpecificOntology (std::string const &apikey, std::string const &ontologyVersionId)**

Retern the detail of specific ontology class id.

- **RestBioPortalSearchResult search** (std::string const &apikey, std::string const &query, std::string const &ontologyids, bool isexactmatch=false, bool includeproperties=false, int maxnumhits=1000, bool includedefinitions=true)

Searching for related ontology term.

- **RestBioPortalTermResult term** (std::string const &apikey, std::string const &ontology_virtual_id, std::string const &concept_id, bool light=false, bool norelation=false, int maxnumchildren=-1)

Retrieving the information of the specific ontology id from specific ontology class.

Static Public Attributes

- static const std::string **baseURL**

The base URI of BioPortal REST service.

Additional Inherited Members

9.51.1 Detailed Description

The main class of BioPortal REST service library.

9.51.2 Constructor & Destructor Documentation

9.51.2.1 unisys::RestBioPortal::RestBioPortal ()

Default constructor.

9.51.3 Member Function Documentation

9.51.3.1 XMLNode unisys::RestBioPortal::getSpecificOntology (std::string const & apikey, std::string const & ontologyVersionId)

Retern the detail of specific ontology class id.

9.51.3.2 XMLNode unisys::RestBioPortal::listAllLastestOntologies (std::string const & apikey)

List all of the lastest ontology classes and thier details.

9.51.3.3 RestBioPortalSearchResult unisys::RestBioPortal::search (std::string const & apikey, std::string const & query, std::string const & ontologyids, bool isexactmatch = false, bool includeproperties = false, int maxnumhits = 1000, bool includedefinitions = true)

Searching for related ontology term.

search The REST service structure for seach command. Signature: ./search/{query}[?{optional args}]{&apikey={YourAPIKey}} Alt Signature: ./search/?query={uri-encoded query}[&{optional args}]{&apikey={YourAPIKey}}

Parameters

apikey	- API key. This is a mandatory parameter. You can get one from BioPortal website.
query	- Query word (examples: "aluminium", "software")
ontologyids	- Ontology class id. You can specify nothing or more than one seperated with comma without space. ("1007" for ChEBI ontology, "1007,1104")
isexactmatch	- Exact match or not. (default: false)

<i>includeproperties</i>	- Return the property information or not. (default: false)
<i>maxnumhits</i>	- Maximum number of result. (default: 1000)
<i>includedefinitions</i>	- If a search result is a hit for a term, adding this parameter will include the definition in the search result xml. (default: true)

9.51.3.4 **RestBioPortalTermResult** unisys::RestBioPortal::term (std::string const & *apikey*, std::string const & *ontology_virtual_id*, std::string const & *concept_id*, bool *light* = false, bool *norelation* = false, int *maxnumchildren* = -1)

Retrieving the information of the specific ontology id from specific ontology class.

term The REST service structure for seach term. Virtual Signature: ./virtual/ontology/{ontology virtual id}/{concept id} Alt Virtual Signature: ./virtual/ontology/{ontology virtual id}?conceptid={uri-encoded concept id}

Parameters

<i>apikey</i>	- API key. This is a mandatory parameter. You can get one from BioPortal website.
<i>ontology_virtual_id</i>	- Ontology class identifier (examples: "1007", "1104")
<i>concept_id</i>	- Ontology term identifier (examples: "CHEBI:28984")
<i>light</i>	- Light version of result or not(default: false)
<i>norelation</i>	- Don't return the relation fields (default: false)
<i>maxnumchildren</i>	- An integer that sets threshold on the number of children in the SubClass relation for a term. If a term contains more children than the "maxnumchildren", the SubClass relation returns an empty list. The ChildCount relation still contains the correct number of children. If this parameter set to minus number, this parameter will not set.(default: -1)

9.51.4 Member Data Documentation

9.51.4.1 const std::string unisys::RestBioPortal::baseURL [static]

The base URI of BioPortal REST service.

The documentation for this class was generated from the following file:

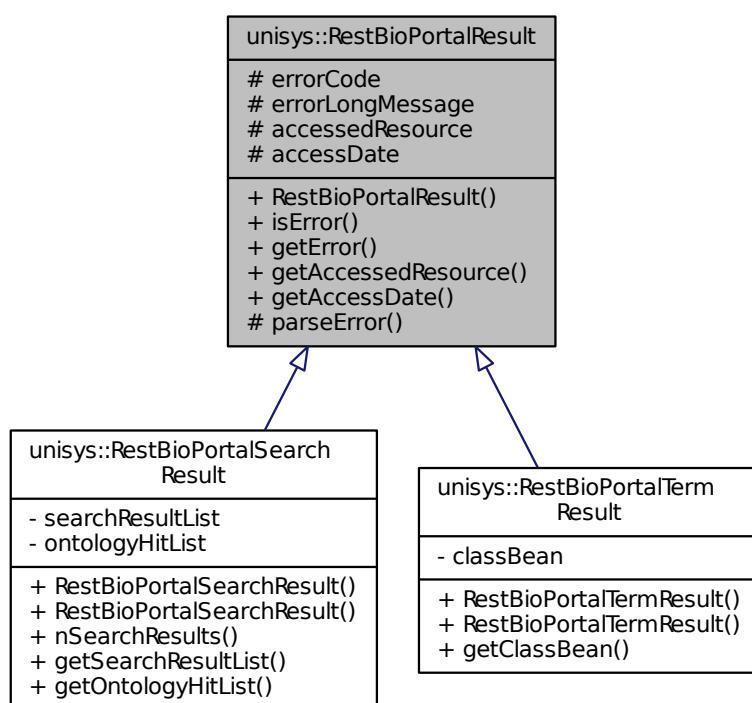
- [restBioPortal.h](#)

9.52 unisys::RestBioPortalResult Class Reference

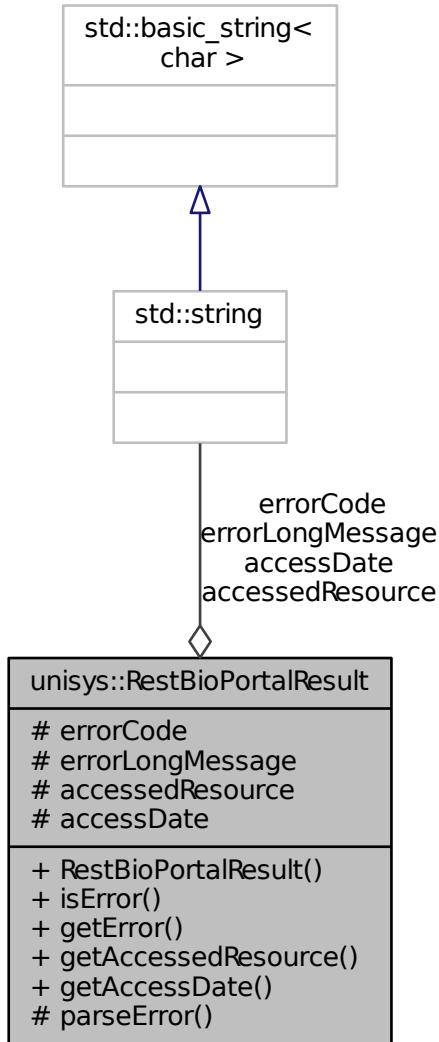
General class for taking care the response from BioPortal's REST service.

```
#include <restBioPortal.h>
```

Inheritance diagram for unisys::RestBioPortalResult:



Collaboration diagram for unisys::RestBioPortalResult:



Public Member Functions

- **RestBioPortalResult ()**
Default constructor.
- **bool isError () const**
To check the response, is it error or not?
- **std::string getError (bool codeOnly=false) const**
Return the error message.
- **std::string getAccessedResource () const**
Return accessedResource.
- **std::string getAccessDate () const**
Return accessDate.

Protected Member Functions

- void `parseError (XMLNode xml)`

The protected member function is that for parsing the error response.

Protected Attributes

- std::string `errorCode`

Store an error code from REST response. This value will be presented only when REST response the error back.

- std::string `errorLongMessage`

Store an error message from REST response, same as errorCode.

- std::string `accessedResource`

Store The resource name that the library has accessed.

- std::string `accessDate`

Store The time that the REST service taking response.

9.52.1 Detailed Description

General class for taking care the response from BioPortal's REST service.

9.52.2 Constructor & Destructor Documentation

9.52.2.1 unisys::RestBioPortalResult::RestBioPortalResult ()

Default constructor.

9.52.3 Member Function Documentation

9.52.3.1 std::string unisys::RestBioPortalResult::getAccessDate () const

Return accessDate.

9.52.3.2 std::string unisys::RestBioPortalResult::getAccessedResource () const

Return accessedResource.

9.52.3.3 std::string unisys::RestBioPortalResult::getError (bool codeOnly = false) const

Return the error message.

9.52.3.4 bool unisys::RestBioPortalResult::isError () const

To check the response, is it error or not?

9.52.3.5 void unisys::RestBioPortalResult::parseError (XMLNode xml) [protected]

The protected member function is that for parsing the error response.

9.52.4 Member Data Documentation

9.52.4.1 `std::string unisys::RestBioPortalResult::accessDate [protected]`

Store The time that the REST service taking response.

9.52.4.2 `std::string unisys::RestBioPortalResult::accessedResource [protected]`

Store The resource name that the library has accessed.

9.52.4.3 `std::string unisys::RestBioPortalResult::errorCode [protected]`

Store an error code from REST response. This value will be presented only when REST response the error back.

9.52.4.4 `std::string unisys::RestBioPortalResult::errorLongMessage [protected]`

Store an error message from REST response, same as errorCode.

The documentation for this class was generated from the following file:

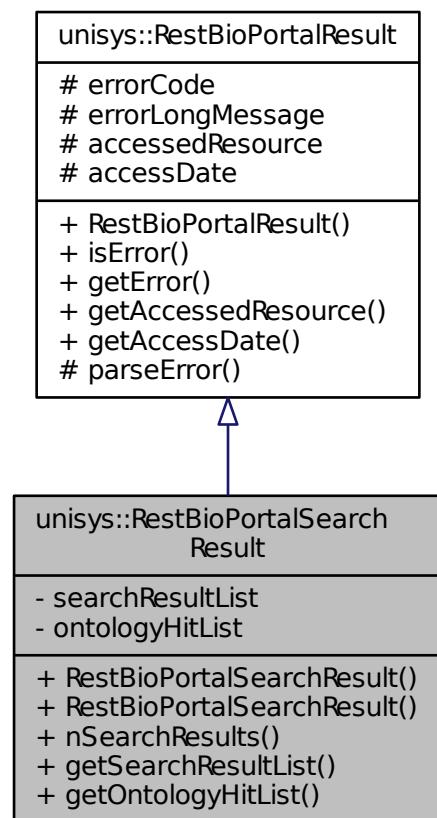
- [restBioPortal.h](#)

9.53 unisys::RestBioPortalSearchResult Class Reference

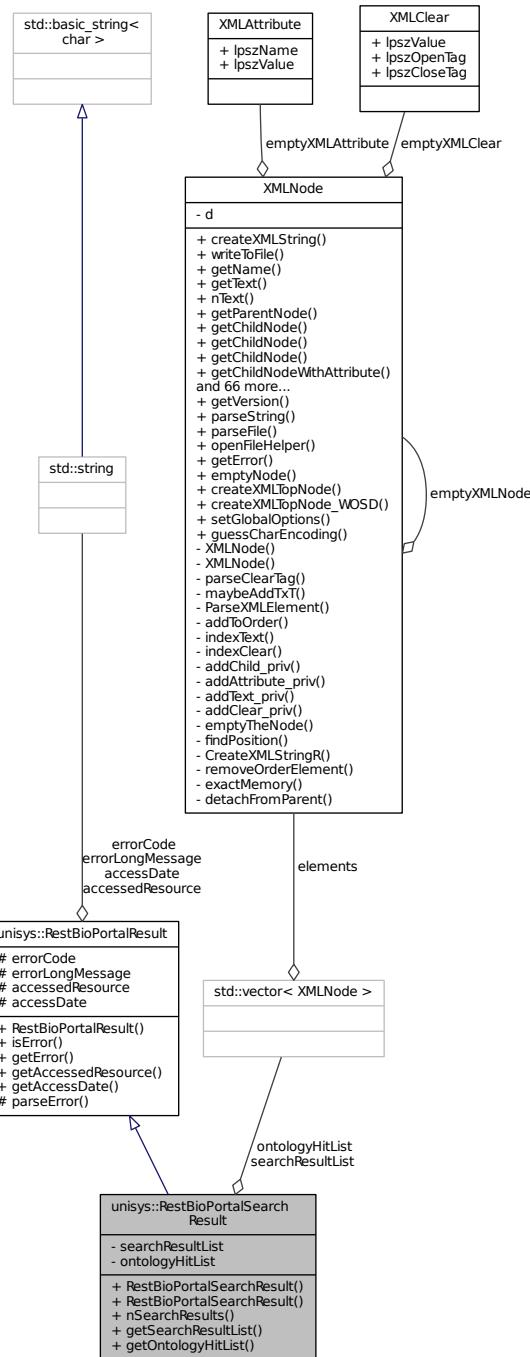
General class for taking care the response from BioPortal's search command of REST service.

```
#include <restBioPortal.h>
```

Inheritance diagram for unisys::RestBioPortalSearchResult:



Collaboration diagram for unisys::RestBioPortalSearchResult:



Public Member Functions

- [RestBioPortalSearchResult \(\)](#)
Default constructor.
- [RestBioPortalSearchResult \(XMLNode xml\)](#)
Overload constructor for parsing XML response.
- [size_t nSearchResults \(\) const](#)

- `std::vector< XMLNode > getSearchResultList () const`
Return the number of result.
- `std::vector< XMLNode > getOntologyHitList () const`
Return the list of result.
- `std::vector< XMLNode > ontologyHitList () const`
Return the ontology details of result.

Private Attributes

- `std::vector< XMLNode > searchResultList`
Store the ontology terms and their information which are match with the query.
- `std::vector< XMLNode > ontologyHitList`
Store the detail of ontology class which the result come from.

Additional Inherited Members

9.53.1 Detailed Description

General class for taking care the response from BioPortal's search command of REST service.

This class is derived from restBioPortalResult class with the specific constructor to parse the response XML document from the BioPortal's search command.

9.53.2 Constructor & Destructor Documentation

9.53.2.1 unisys::RestBioPortalSearchResult::RestBioPortalSearchResult ()

Default constructor.

9.53.2.2 unisys::RestBioPortalSearchResult::RestBioPortalSearchResult (XMLNode xml)

Overload constructor for parsing XML response.

9.53.3 Member Function Documentation

9.53.3.1 std::vector<XMLNode> unisys::RestBioPortalSearchResult::getOntologyHitList () const

Return the ontology details of result.

9.53.3.2 std::vector<XMLNode> unisys::RestBioPortalSearchResult::getSearchResultList () const

Return the list of result.

9.53.3.3 size_t unisys::RestBioPortalSearchResult::nSearchResults () const

Return the number of result.

9.53.4 Member Data Documentation

9.53.4.1 std::vector<XMLNode> unisys::RestBioPortalSearchResult::ontologyHitList [private]

Store the detail of ontology class which the result come from.

9.53.4.2 std::vector<XMLNode> unisys::RestBioPortalSearchResult::searchResultList [private]

Store the ontology terms and their information which are match with the query.

The documentation for this class was generated from the following file:

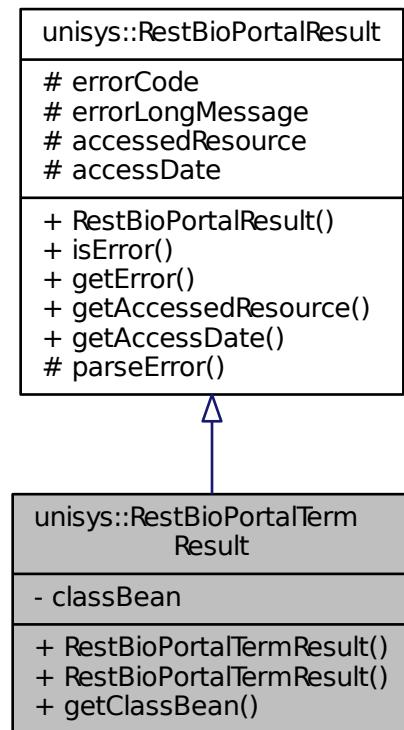
- [restBioPortal.h](#)

9.54 unisys::RestBioPortalTermResult Class Reference

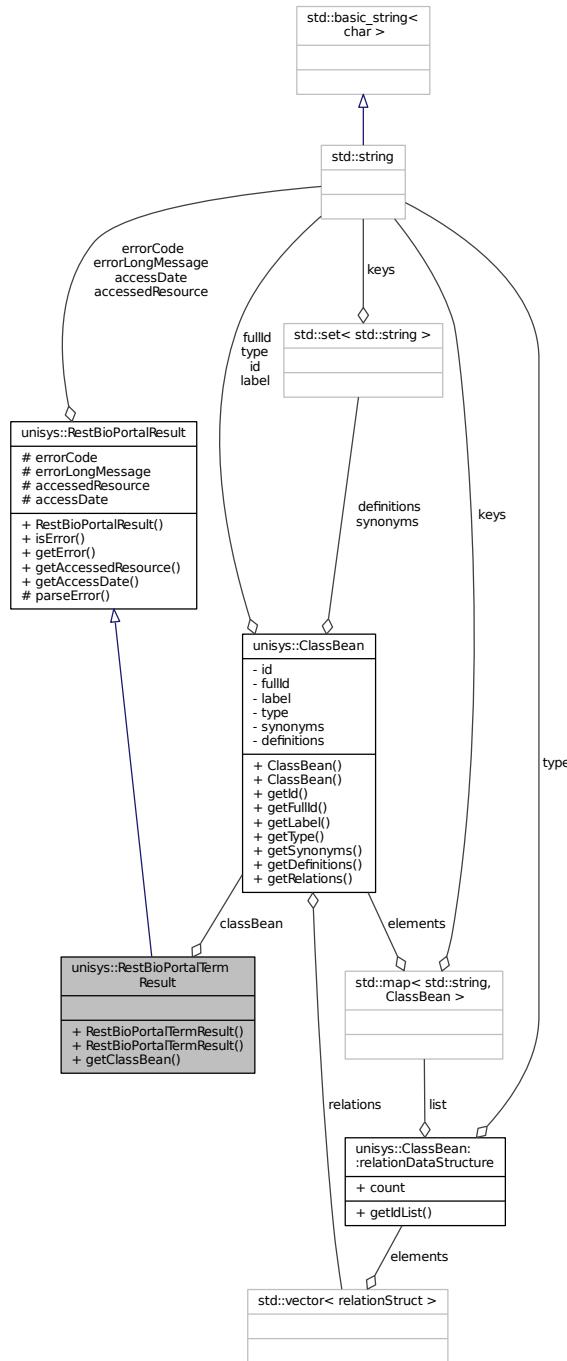
This class is for parsing the response of term command.

```
#include <restBioPortal.h>
```

Inheritance diagram for unisys::RestBioPortalTermResult:



Collaboration diagram for unisys::RestBioPortalTermResult:



Public Member Functions

- [RestBioPortalTermResult \(\)](#)
Default constructor.
- [RestBioPortalTermResult \(XMLNode xml\)](#)
Overload constructor for parsing XML response.
- [ClassBean getClassBean \(\) const](#)

Return classBean.

Private Attributes

- [ClassBean classBean](#)

A response has one classBean.

Additional Inherited Members

9.54.1 Detailed Description

This class is for parsing the response of term command.

9.54.2 Constructor & Destructor Documentation

9.54.2.1 [unisys::RestBioPortalTermResult::RestBioPortalTermResult \(\)](#)

Default constructor.

9.54.2.2 [unisys::RestBioPortalTermResult::RestBioPortalTermResult \(XMLNode xml \)](#)

Overload constructor for parsing XML response.

9.54.3 Member Function Documentation

9.54.3.1 [ClassBean unisys::RestBioPortalTermResult::getClassBean \(\) const](#)

Return classBean.

9.54.4 Member Data Documentation

9.54.4.1 [ClassBean unisys::RestBioPortalTermResult::classBean \[private\]](#)

A response has one classBean.

The documentation for this class was generated from the following file:

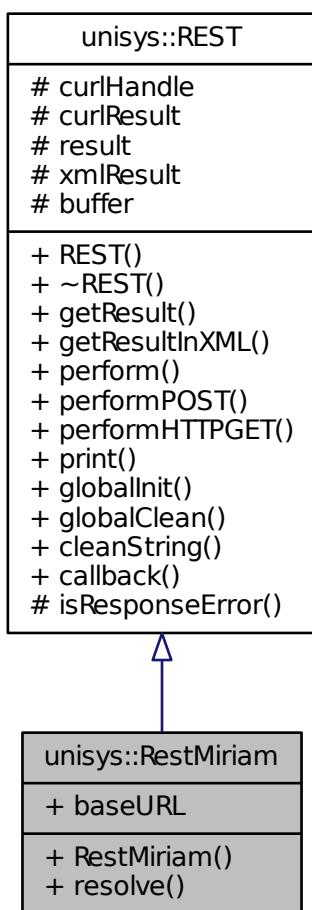
- [restBioPortal.h](#)

9.55 [unisys::RestMiriam Class Reference](#)

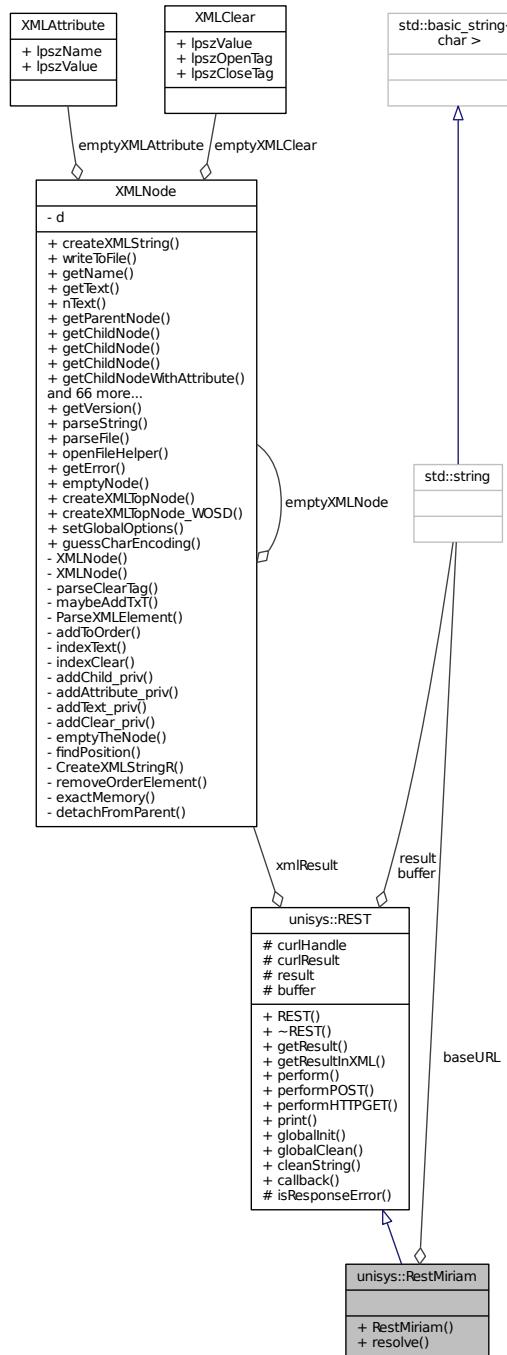
General class for taking care the response from BioPortal's [REST](#) service.

```
#include <restMiriam.h>
```

Inheritance diagram for unisys::RestMiriam:



Collaboration diagram for unisys::RestMiriam:



Public Member Functions

- `RestMiriam ()`
- `std::vector< std::string > resolve (std::string urn, bool getDeprecated=false) throw (RESTServiceError)`
return URL from provided Miriam URN

Static Public Attributes

- static const std::string [baseURL](#)

Additional Inherited Members

9.55.1 Detailed Description

General class for taking care the response from BioPortal's [REST](#) service.

9.55.2 Constructor & Destructor Documentation

9.55.2.1 [unisys::RestMiriam::RestMiriam\(\)](#)

9.55.3 Member Function Documentation

9.55.3.1 [std::vector<std::string> unisys::RestMiriam::resolve\(std::string urn, bool getDeprecated = false \) throw\(RESTServiceError \)](#)

return URL from provided [Miriam](#) URN

Parameters

<i>urn</i>	input miriam URN
<i>getDeprecated</i>	-

9.55.4 Member Data Documentation

9.55.4.1 [const std::string unisys::RestMiriam::baseURL \[static\]](#)

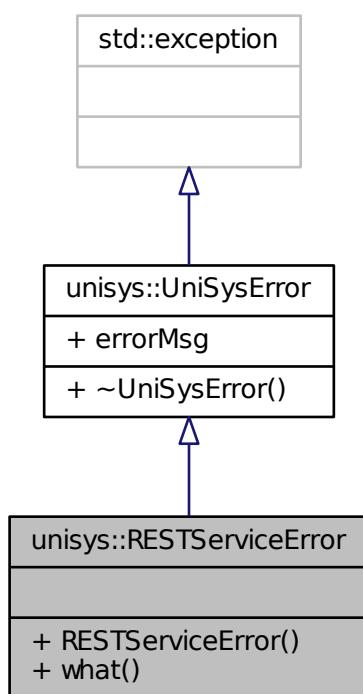
The documentation for this class was generated from the following file:

- [restMiriam.h](#)

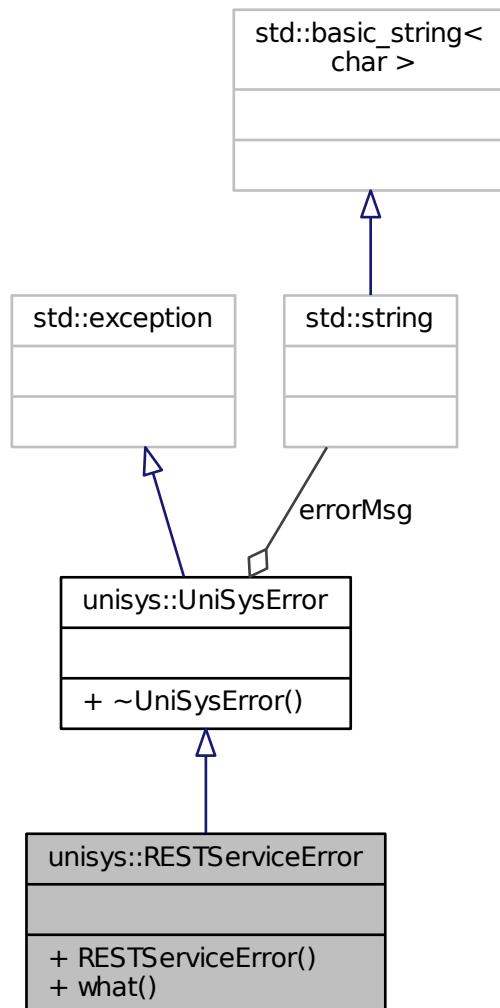
9.56 unisys::RESTServiceError Class Reference

```
#include <exception.h>
```

Inheritance diagram for unisys::RESTServiceError:



Collaboration diagram for unisys::RESTServiceError:



Public Member Functions

- [RESTServiceError \(std::string const &str\)](#)
- [virtual const char * what \(\) const throw \(\)](#)

Additional Inherited Members

9.56.1 Constructor & Destructor Documentation

[9.56.1.1 unisys::RESTServiceError::RESTServiceError \(std::string const & str \) \[inline\]](#)

9.56.2 Member Function Documentation

9.56.2.1 virtual const char* unisys::RESTServiceError::what() const throw() [inline], [virtual]

The documentation for this class was generated from the following file:

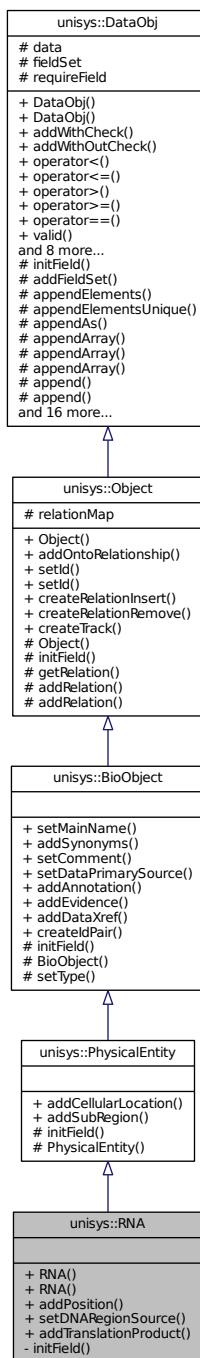
- [exception.h](#)

9.57 unisys::RNA Class Reference

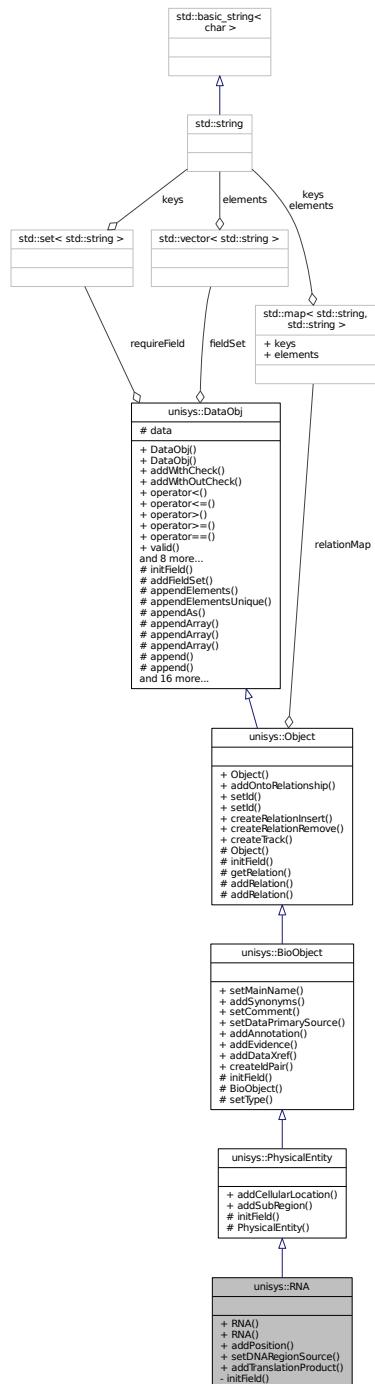
This class is for miriam cross reference annotation.

```
#include <ObjClass.h>
```

Inheritance diagram for unisys::RNA:



Collaboration diagram for unisys::RNA:



Public Member Functions

- [RNA \(\)](#)

Default constructor.

- [RNA \(mongo::BSONObj const &bsonObj\)](#)

Overloaded constructor is used when retrieving data in bson object from database and transform to C++ object.

- void [addPosition](#) (unsigned int start, unsigned int stop)

- void [setDNARegionSource \(PEIdRef &peldRef\)](#)
- void [addTranslationProduct \(PEIdRef &trans\)](#)

Private Member Functions

- void [initField \(\)](#)

Additional Inherited Members

9.57.1 Detailed Description

This class is for miriam cross reference annotation.

```
 BSON structure:
{
    _id: <string>, #madatory
    ontologyRelationship: {<RelationshipBOSON>, <RelationshipBOSON>, ...},
    name: {<string>, <string>, ...}
    comment: <string>,
    dataPrimarySource: <XrefBOSON>,
    functionAnnotation: {<AnnotationBOSON>, <AnnotationBOSON>, ...},
    evidence: {<EvidenceBOSON>, <EvidenceBOSON>, ...},
    dataxref: {<XrefBOSON>, <XrefBOSON>, ...},
    interaction: {<DbRef>, <DbRef>, ...},
    complex: {<DbRef>, <DbRef>, ...},
    subRegion: {<SubRegionBOSON>, <SubRegionBOSON>, ...},
    position: {<positionPairBSON>, <positionPairBSON>, ...},
    dnaRegionSource: <DbRef>,
    translationProduct: {<DbRef>, <DbRef>, ...}
}
```

9.57.2 Constructor & Destructor Documentation

9.57.2.1 unisys::RNA::RNA ()

Default constructor.

9.57.2.2 unisys::RNA::RNA (mongo::BSONObj const & *bsonObj*)

Overloaded constructor is used when retrieving data in boson object from database and tranform to C++ object.

9.57.3 Member Function Documentation

9.57.3.1 void unisys::RNA::addPosition (unsigned int *start*, unsigned int *stop*)

9.57.3.2 void unisys::RNA::addTranslationProduct (PEIdRef & *trans*)

9.57.3.3 void unisys::RNA::initField () [private], [virtual]

Reimplemented from [unisys::PhysicalEntity](#).

9.57.3.4 void unisys::RNA::setDNARegionSource (PEIdRef & *peldRef*)

The documentation for this class was generated from the following file:

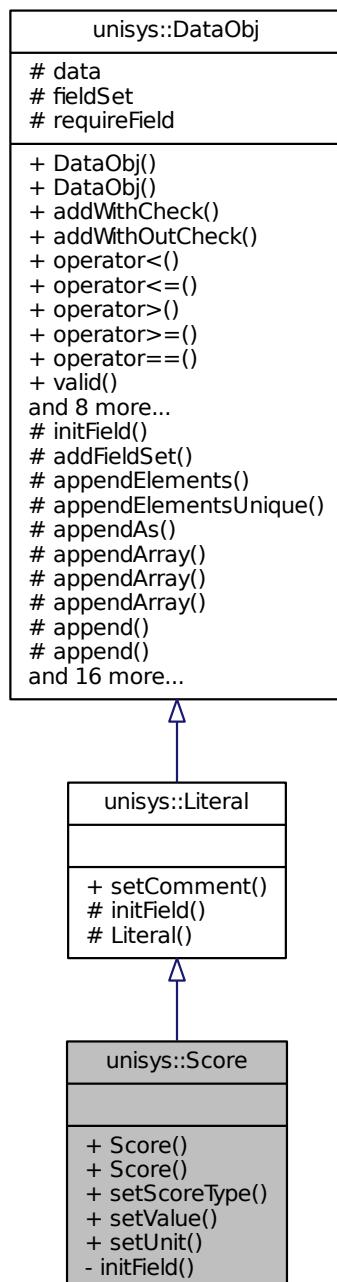
- [ObjClass.h](#)

9.58 unisys::Score Class Reference

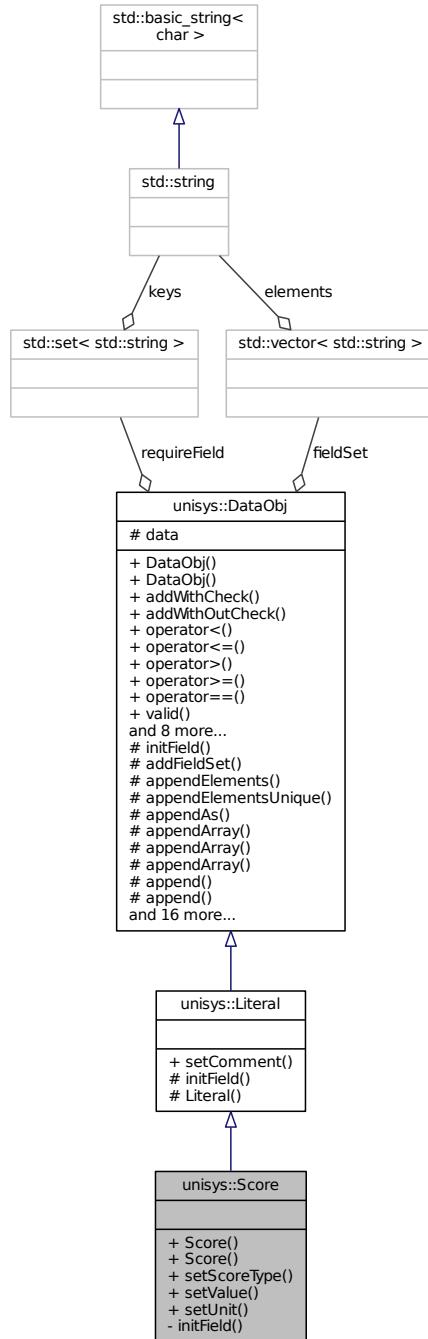
The C++ representative class for [Score](#) class in data structure.

```
#include <LitClass.h>
```

Inheritance diagram for unisys::Score:



Collaboration diagram for unisys::Score:



Public Member Functions

- [Score \(\)](#)
Default constructor.
- [Score \(mongo::BSONObj const &bsonObj\)](#)
Overloaded constructor is used when retrieving data in bson object from database and transform to C++ object.
- [void setScoreType \(OntoldRef &ontoldRef\)](#)

- void [setValue](#) (double value)
- void [setUnit](#) (OntoldRef &ontoldRef)

Private Member Functions

- void [initField](#) ()

function for init field in the object

Additional Inherited Members

9.58.1 Detailed Description

The C++ representative class for [Score](#) class in data structure.

```
 BSON structure:
{
    comment: <string>,
    scoreType: {$ref: <collname>, $id: <idvalue>} , #format by idref class
    value: <number>,
    unit: {$ref: <collname>, $id: <idvalue>} #format by idref class
}
```

9.58.2 Constructor & Destructor Documentation

9.58.2.1 [unisys::Score::Score\(\)](#)

Default constructor.

9.58.2.2 [unisys::Score::Score\(mongo::BSONObj const & bsonObj \)](#)

Overloaded constructor is used when retrieving data in boson object from database and tranform to C++ object.

9.58.3 Member Function Documentation

9.58.3.1 [void unisys::Score::initField\(\) \[private\], \[virtual\]](#)

function for init field in the object

Reimplemented from [unisys::Literal](#).

9.58.3.2 [void unisys::Score::setScoreType\(OntoldRef & ontoldRef \)](#)

9.58.3.3 [void unisys::Score::setUnit\(OntoldRef & ontoldRef \)](#)

9.58.3.4 [void unisys::Score::setValue\(double value \)](#)

The documentation for this class was generated from the following file:

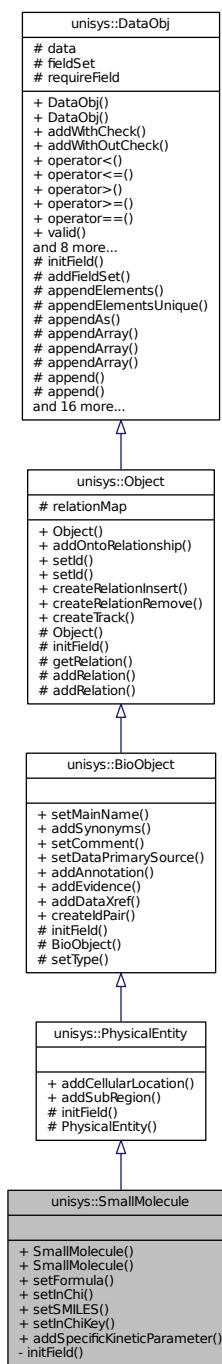
- [LitClass.h](#)

9.59 unisys::SmallMolecule Class Reference

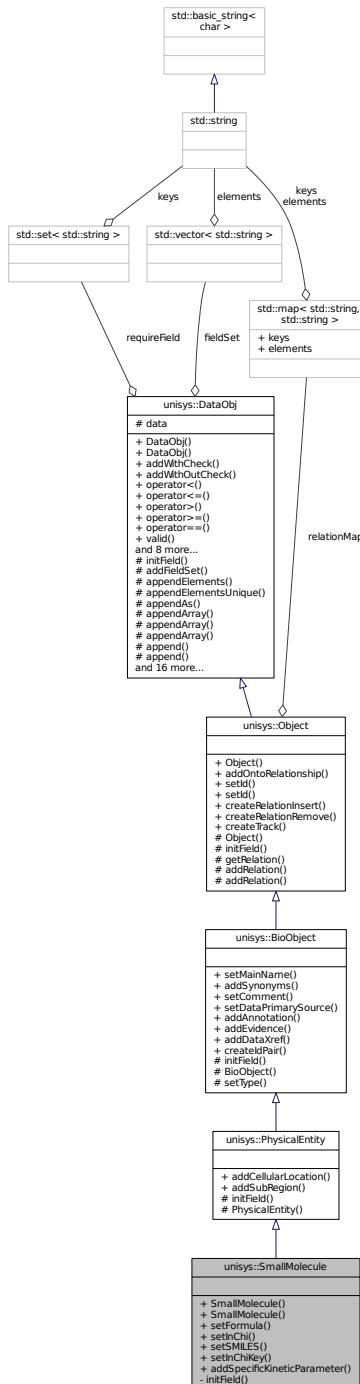
This class is for miriam cross reference annotation.

```
#include <ObjClass.h>
```

Inheritance diagram for unisys::SmallMolecule:



Collaboration diagram for unisys::SmallMolecule:



Public Member Functions

- [SmallMolecule \(\)](#)
Default constructor.
- [SmallMolecule \(mongo::BSONObj const &bsonObj\)](#)
Overloaded constructor is used when retrieving data in bson object from database and transform to C++ object.
- void [setFormula \(std::string const &formula\)](#)

- void [setInChi](#) (std::string const &inchi, bool isParseFormula=true)
- void [setSMILES](#) (std::string const &smiles)
- void [setInChiKey](#) (std::string const &inchiKey)
- void [addSpecificKineticParameter](#) (KineticParameter &kineticParameter)

Private Member Functions

- void [initField](#) ()

Additional Inherited Members

9.59.1 Detailed Description

This class is for miriam cross reference annotation.

```
BSON structure:
{
    _id: <string>, #madatory
    ontologyRelationship: {<RelationshipBOSON>, <RelationshipBOSON>, ...},
    names: {<string>, <string>, ...}
    comment: <string>,
    dataPrimarySource: <XrefBOSON>,
    functionAnnotation: {<AnnotationBOSON>, <AnnotationBOSON>, ...},
    evidence: {<EvidenceBOSON>, <EvidenceBOSON>, ...},
    dataxref: {<XrefBOSON>, <XrefBOSON>, ...},
    interaction: {<DbRef>, <DbRef>, ...},
    complex: {<DbRef>, <DbRef>, ...},
    subRegion: {<SubRegionBOSON>, <SubRegionBOSON>, ...},
    formular: <string>,
    InChi: <string>,
    smiles: <string>,
    InChiKey: <string>
}
```

9.59.2 Constructor & Destructor Documentation

9.59.2.1 unisys::SmallMolecule::SmallMolecule()

Default constructor.

9.59.2.2 unisys::SmallMolecule::SmallMolecule(mongo::BSONObj const & bsonObj)

Overloaded constructor is used when retrieving data in boson object from database and transform to C++ object.

9.59.3 Member Function Documentation

9.59.3.1 void unisys::SmallMolecule::addSpecificKineticParameter(KineticParameter & kineticParameter)

9.59.3.2 void unisys::SmallMolecule::initField() [private], [virtual]

Reimplemented from [unisys::PhysicalEntity](#).

9.59.3.3 void unisys::SmallMolecule::setFormula(std::string const & formula)

9.59.3.4 void unisys::SmallMolecule::setInChi(std::string const & inchi, bool isParseFormula = true)

9.59.3.5 void unisys::SmallMolecule::setInChiKey (std::string const & *inchiKey*)

9.59.3.6 void unisys::SmallMolecule::setSMILES (std::string const & *smiles*)

The documentation for this class was generated from the following file:

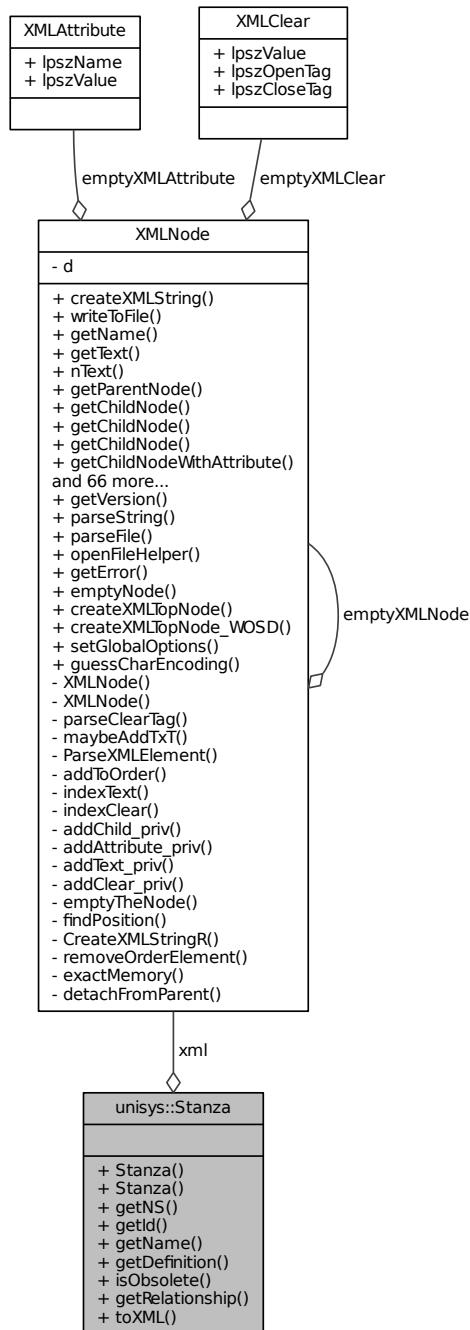
- [ObjClass.h](#)

9.60 unisys::Stanza Class Reference

This class is used to parse OWL format.

```
#include <oboXML.h>
```

Collaboration diagram for unisys::Stanza:



Public Member Functions

- `Stanza ()`
- `Stanza (XMLNode const &xmlnode)`
- `std::string getNS () const`
- `std::string getId () const`
- `std::string getName () const`

- std::string [getDefinition \(\) const](#)
- bool [isObsolete \(\) const](#)
- std::map< std::string, std::set< std::string > > [getRelationship \(\) const](#)
- [XMLNode toXML \(\) const](#)

Private Attributes

- [XMLNode xml](#)

9.60.1 Detailed Description

This class is used to parse OWL format.
some description.

9.60.2 Constructor & Destructor Documentation

9.60.2.1 [unisys::Stanza::Stanza \(\)](#)

9.60.2.2 [unisys::Stanza::Stanza \(XMLNode const & *xmlnode* \)](#)

9.60.3 Member Function Documentation

9.60.3.1 [std::string unisys::Stanza::getDefinition \(\) const](#)

9.60.3.2 [std::string unisys::Stanza::getId \(\) const](#)

9.60.3.3 [std::string unisys::Stanza::getName \(\) const](#)

9.60.3.4 [std::string unisys::Stanza::getNS \(\) const](#)

9.60.3.5 [std::map<std::string, std::set<std::string> > unisys::Stanza::getRelationship \(\) const](#)

9.60.3.6 [bool unisys::Stanza::isObsolete \(\) const](#)

9.60.3.7 [XMLNode unisys::Stanza::toXML \(\) const](#)

9.60.4 Member Data Documentation

9.60.4.1 [XMLNode unisys::Stanza::xml \[private\]](#)

The documentation for this class was generated from the following file:

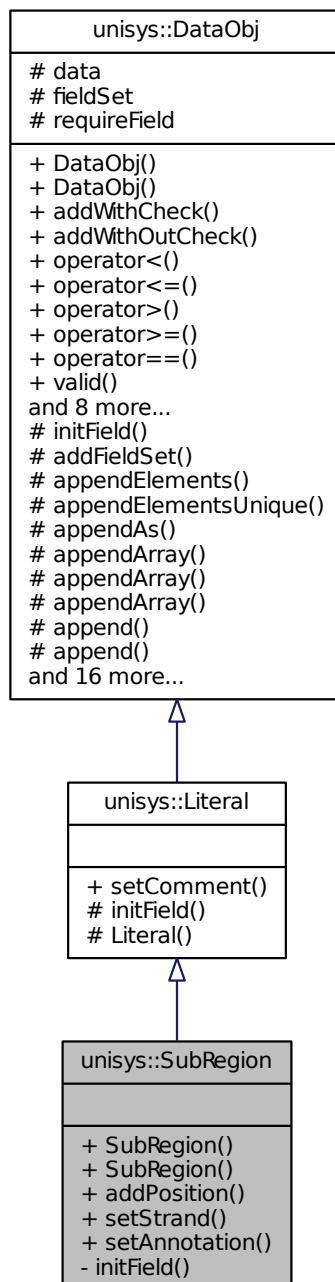
- [oboXML.h](#)

9.61 unisys::SubRegion Class Reference

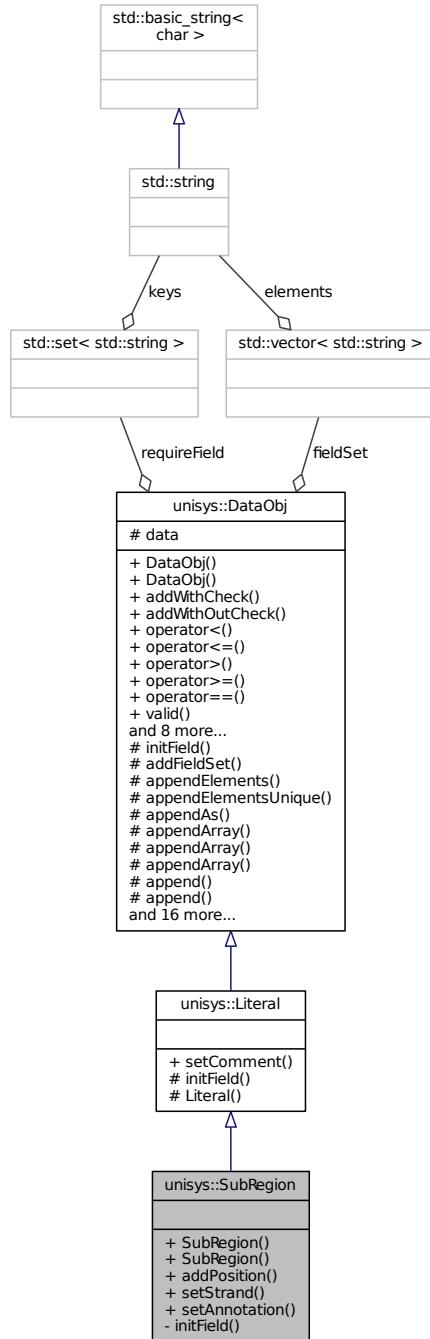
The C++ representative class for sub-region data class.

```
#include <LitClass.h>
```

Inheritance diagram for unisys::SubRegion:



Collaboration diagram for unisys::SubRegion:



Public Member Functions

- [SubRegion \(\)](#)
Default constructor.
- [SubRegion \(mongo::BSONObj const &bsonObj\)](#)
Overloaded constructor is used when retrieving data in bson object from database and transform to C++ object.
- void [addPosition \(unsigned int start, unsigned int stop\)](#)

- void [setStrand](#) (int strand=0)
- void [setAnnotation](#) ([Annotation](#) &annotation)

Private Member Functions

- void [initField](#) ()
function for init field in the object

Additional Inherited Members

9.61.1 Detailed Description

The C++ representative class for sub-region data class.

```
 BSON structure:
{
    comment: <string>,
    position: {start: <number>, stop: <number>}, {start: <number>, stop: <number>}, ...},
    strand: <[-1|0|1]> default:0 = unknown,
    annotation: <AnnotationBSONStructure>
}
```

9.61.2 Constructor & Destructor Documentation

9.61.2.1 [unisys::SubRegion::SubRegion\(\)](#)

Default constructor.

9.61.2.2 [unisys::SubRegion::SubRegion\(mongo::BSONObj const & bsonObj \)](#)

Overloaded constructor is used when retrieving data in boson object from database and transform to C++ object.

9.61.3 Member Function Documentation

9.61.3.1 [void unisys::SubRegion::addPosition\(unsigned int start, unsigned int stop \)](#)

9.61.3.2 [void unisys::SubRegion::initField\(\) \[private\], \[virtual\]](#)

function for init field in the object

Reimplemented from [unisys::Literal](#).

9.61.3.3 [void unisys::SubRegion::setAnnotation\(Annotation & annotation \)](#)

9.61.3.4 [void unisys::SubRegion::setStrand\(int strand = 0 \)](#)

The documentation for this class was generated from the following file:

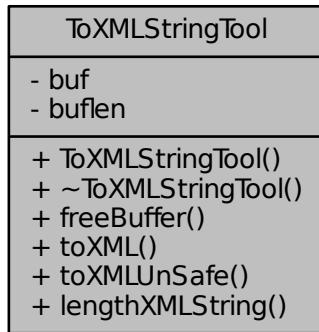
- [LitClass.h](#)

9.62 ToXMLStringTool Struct Reference

Helper class to create XML files using "printf", "fprintf", "cout",... functions.

```
#include <xmlParser.h>
```

Collaboration diagram for ToXMLStringTool:



Public Member Functions

- [ToXMLStringTool \(\)](#)
- [~ToXMLStringTool \(\)](#)
- [void freeBuffer \(\)](#)
call this function when you have finished using this object to release memory used by the internal buffer.
- [XMLSTR toXML \(XMLCSTR source\)](#)
returns a pointer to an internal buffer that contains a XML-encoded string based on the "source" parameter.

Static Public Member Functions

- static [XMLSTR toXMLUnSafe \(XMLSTR dest, XMLCSTR source\)](#)
deprecated: use "toXML" instead
- static int [lengthXMLString \(XMLCSTR source\)](#)
deprecated: use "toXML" instead

Private Attributes

- [XMLSTR buf](#)
- [int buflen](#)

9.62.1 Detailed Description

Helper class to create XML files using "printf", "fprintf", "cout",... functions.

The [ToXMLStringTool](#) class helps you creating XML files using "printf", "fprintf", "cout",... functions. The "ToXMLStringTool" class is processing strings so that all the characters &,"',<,> are replaced by their XML equivalent:

```
&amp;, &quot;, &apos;, &lt;, &gt;
```

Using the "ToXMLStringTool class" and the "fprintf function" is THE most efficient way to produce VERY large XML documents VERY fast.

Note

If you are creating from scratch an XML file using the provided `XMLNode` class you must not use the "ToXMLStringTool" class (because the "XMLNode" class does the processing job for you during rendering).

9.62.2 Constructor & Destructor Documentation

9.62.2.1 `ToXMLStringTool::ToXMLStringTool() [inline]`

9.62.2.2 `ToXMLStringTool::~ToXMLStringTool()`

9.62.3 Member Function Documentation

9.62.3.1 `void ToXMLStringTool::freeBuffer()`

call this function when you have finished using this object to release memory used by the internal buffer.

9.62.3.2 `static int ToXMLStringTool::lengthXMLString(XMLCSTR source) [static]`

deprecated: use "toXML" instead

9.62.3.3 `XMLSTR ToXMLStringTool::toXML(XMLCSTR source)`

returns a pointer to an internal buffer that contains a XML-encoded string based on the "source" parameter.

9.62.3.4 `static XMLSTR ToXMLStringTool::toXMLUnSafe(XMLSTR dest, XMLCSTR source) [static]`

deprecated: use "toXML" instead

The "toXMLUnSafe" function is deprecated because there is a possibility of "destination-buffer-overflow". It converts the string "source" to the string "dest".

9.62.4 Member Data Documentation

9.62.4.1 `XMLSTR ToXMLStringTool::buf [private]`

9.62.4.2 `int ToXMLStringTool::buflen [private]`

The documentation for this struct was generated from the following file:

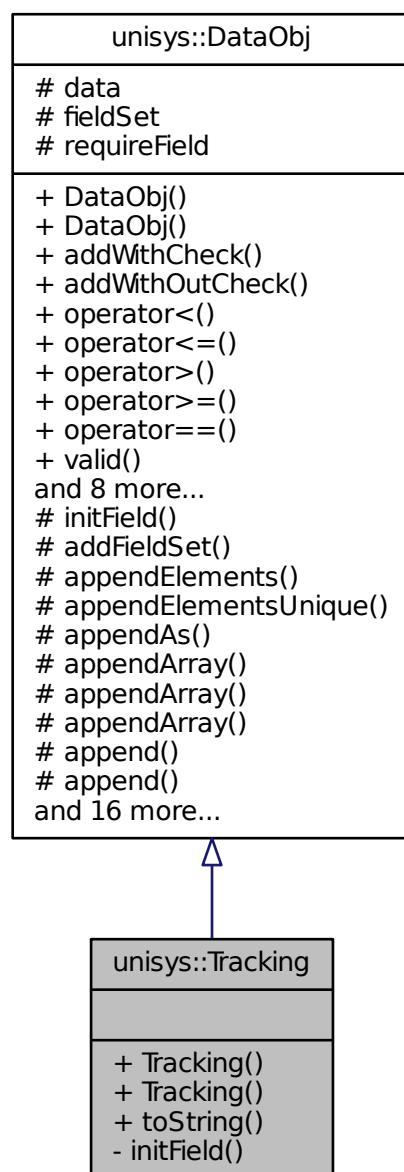
- [xmlParser.h](#)

9.63 unisys::Tracking Class Reference

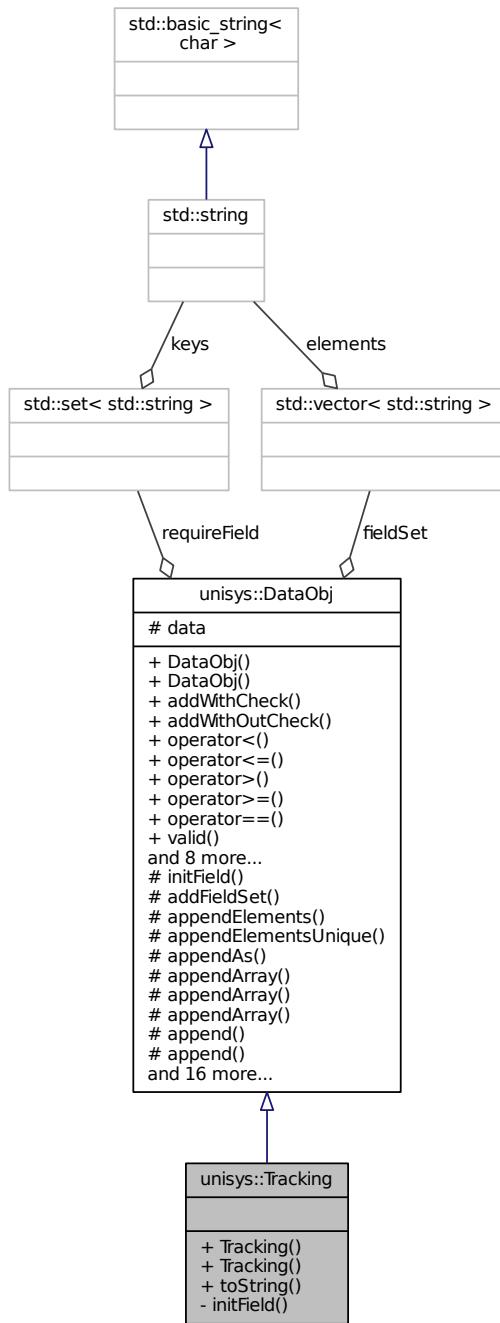
Data updating history class.

```
#include <DBClass.h>
```

Inheritance diagram for unisys::Tracking:



Collaboration diagram for unisys::Tracking:



Public Member Functions

- [Tracking](#) (mongo::BSONObj const &bsonObj)

- [Tracking](#) (std::string const &objectId, std::string const &activity)

Overloaded constructor to build class instance.

- std::string [toString](#) () const

Return the string.

Private Member Functions

- void [initField \(\)](#)

Additional Inherited Members

9.63.1 Detailed Description

Data updating history class.

This class makes a editing time stamp of each data documents. Structure of [Tracking](#) class in boson document is {oid: "Updated object id", activity: "activity of the update; such as, new object or change detail in abc field", timeStamp: "time stamp"}.

BSON structure: { oid: <idvalue>, activity: "xxx", timeStamp: <dateTimeValue>, }

9.63.2 Constructor & Destructor Documentation

[9.63.2.1 unisys::Tracking::Tracking \(mongo::BSONObj const & bsonObj \)](#)

[9.63.2.2 unisys::Tracking::Tracking \(std::string const & objectID, std::string const & activity \)](#)

Overloaded constructor to build class instance.

9.63.3 Member Function Documentation

[9.63.3.1 void unisys::Tracking::initField \(\) \[private\], \[virtual\]](#)

Implements [unisys::DataObj](#).

[9.63.3.2 std::string unisys::Tracking::toString \(\) const](#)

Return the string.

The documentation for this class was generated from the following file:

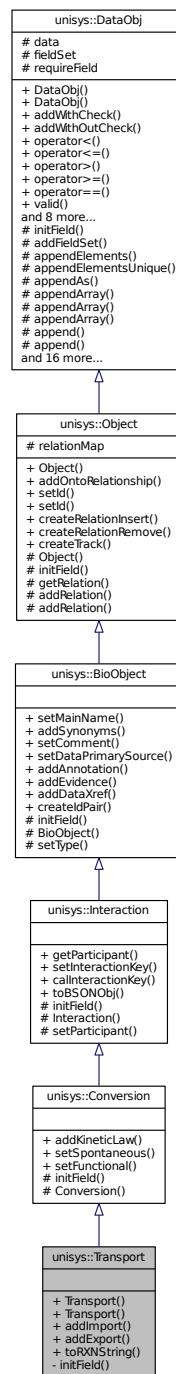
- [DBClass.h](#)

9.64 unisys::Transport Class Reference

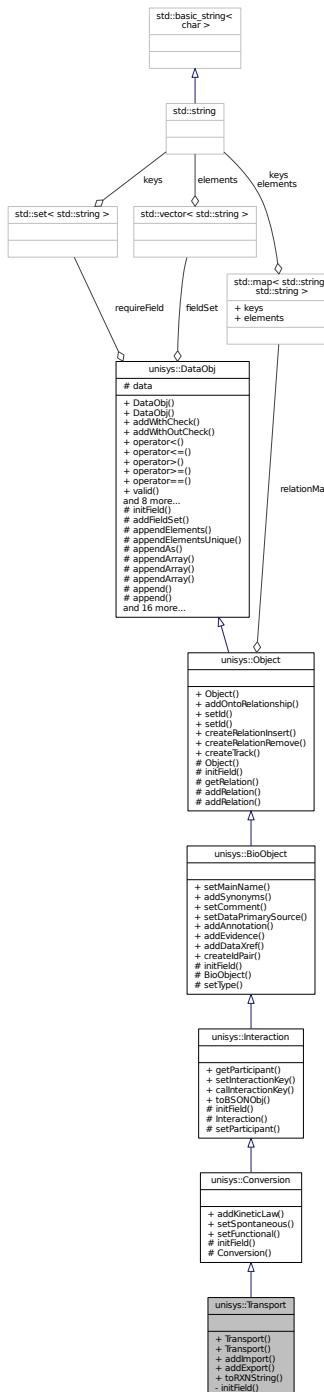
This class is for miriam cross reference annotation.

```
#include <ObjClass.h>
```

Inheritance diagram for unisys::Transport:



Collaboration diagram for unisys::Transport:



Public Member Functions

- [Transport \(\)](#)
Default constructor.
- [Transport \(mongo::BSONObj const &bsonObj\)](#)
Overloaded constructor is used when retrieving data in boson object from database and transform to C++ object.
- [void addImport \(PEldRef &peldRef, double coefficient\)](#)

- void [addExport \(PEIdRef &peldRef, double coefficient\)](#)
- std::string [toRXNString \(std::string const &outside, std::string const &inside\) const](#)

Private Member Functions

- void [initField \(\)](#)

Additional Inherited Members

9.64.1 Detailed Description

This class is for miriam cross reference annotation.

```
 BSON structure:
{
    _id: <string>, #mandatory
    type: <string>,
    ontologyRelationship: {<RelationshipBOSON>, <RelationshipBOSON>, ...},
    name: {<string>, <string>, ...}
    comment: <string>,
    dataPrimarySource: <XrefBOSON>,
    functionAnnotation: {<AnnotationBOSON>, <AnnotationBOSON>, ...},
    evidence: {<EvidenceBOSON>, <EvidenceBOSON>, ...},
    dataxref: {<XrefBOSON>, <XrefBOSON>, ...},
    participant: {<StoichiometryBOSON>, <StoichiometryBOSON>, ...},
    interactionKey: <string>,
    conversionDirection: <DbRef>,
    kineticLaw: {<MathMLBOSON>, <MathMLBOSON>, ...}
    spontaneous: <string>,
    functional: <string>
}
```

9.64.2 Constructor & Destructor Documentation

9.64.2.1 unisys::Transport::Transport()

Default constructor.

9.64.2.2 unisys::Transport::Transport(mongo::BSONObj const & bsonObj)

Overloaded constructor is used when retrieving data in boson object from database and transform to C++ object.

9.64.3 Member Function Documentation

9.64.3.1 void unisys::Transport::addExport (PEIdRef & peldRef, double coefficient)

9.64.3.2 void unisys::Transport::addImport (PEIdRef & peldRef, double coefficient)

9.64.3.3 void unisys::Transport::initField () [private], [virtual]

Reimplemented from [unisys::Conversion](#).

9.64.3.4 std::string unisys::Transport::toRXNString (std::string const & outside, std::string const & inside) const

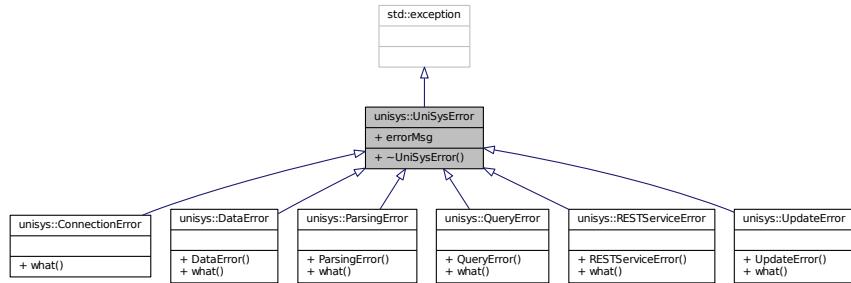
The documentation for this class was generated from the following file:

- [ObjClass.h](#)

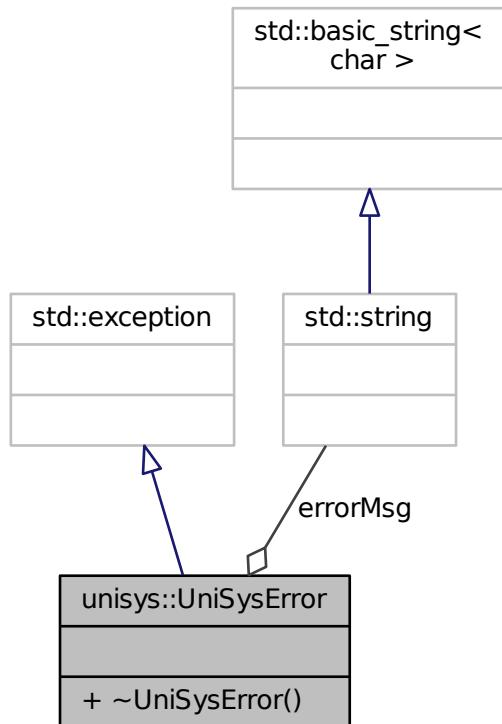
9.65 unisys::UniSysError Class Reference

```
#include <exception.h>
```

Inheritance diagram for unisys::UniSysError:



Collaboration diagram for unisys::UniSysError:



Public Member Functions

- `~UniSysError () throw ()`

Public Attributes

- std::string [errorMsg](#)

9.65.1 Constructor & Destructor Documentation

9.65.1.1 `unisys::UniSysError::~UniSysError() throw () [inline]`

9.65.2 Member Data Documentation

9.65.2.1 std::string `unisys::UniSysError::errorMsg`

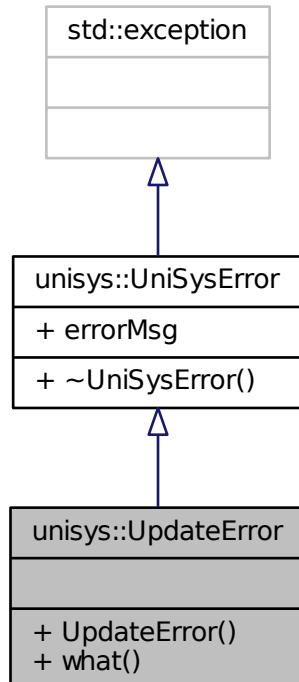
The documentation for this class was generated from the following file:

- [exception.h](#)

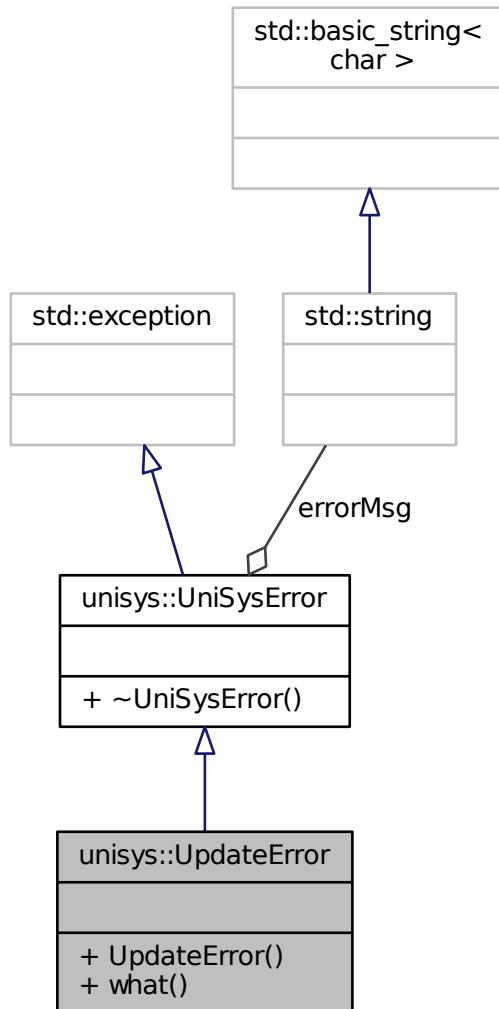
9.66 unisys::UpdateError Class Reference

```
#include <exception.h>
```

Inheritance diagram for unisys::UpdateError:



Collaboration diagram for unisys::UpdateError:



Public Member Functions

- [UpdateError](#) (`std::string const &str`)
- virtual const char * [what](#) () const throw ()

Additional Inherited Members

9.66.1 Constructor & Destructor Documentation

9.66.1.1 `unisys::UpdateError::UpdateError(std::string const & str) [inline]`

9.66.2 Member Function Documentation

9.66.2.1 virtual const char* unisys::UpdateError::what() const throw() [inline], [virtual]

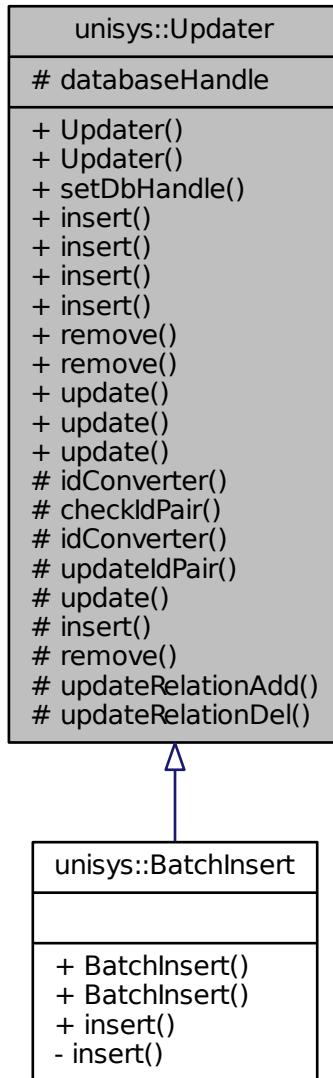
The documentation for this class was generated from the following file:

- [exception.h](#)

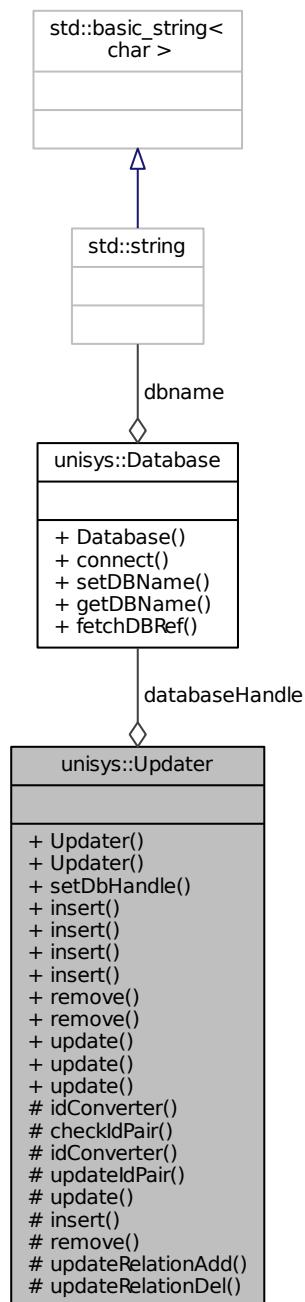
9.67 unisys::Updater Class Reference

```
#include <updater.h>
```

Inheritance diagram for unisys::Updater:



Collaboration diagram for unisys::Updater:



Public Member Functions

- `Updater ()`
- `Updater (Database *databaseHandlePt)`
- `void setDbHandle (Database *databaseHandlePt)`
- `void insert (Tracking tracking) throw (UpdateError, DataError)`
- `void insert (Ontology &ontology) throw (UpdateError, DataError)`

- void `insert (PhysicalEntity &data, bool strict=true) throw (UpdateError, DataError)`
- void `insert (Interaction &data, bool strict=true) throw (UpdateError, DataError)`
- void `remove (std::string const &collectionNS, std::string const &id, bool removeRela=true, bool removeProduct=true) throw (UpdateError)`
`compare cross reference to object in database`
- void `remove (std::string const &collectionNS, std::set< std::string > const &ids, bool removeRela=true, bool removeProduct=true) throw (UpdateError)`
`compare cross reference to object in database`
- void `update (Ontology const &ontology, bool check=true) throw (UpdateError, DataError)`
`compare cross reference to object in database`
- void `update (PhysicalEntity &physicalentity, bool check=true) throw (UpdateError, DataError)`
`compare cross reference to object in database`
- void `update (Interaction &interaction, bool check=true) throw (UpdateError, DataError)`
`compare cross reference to object in database`

Protected Member Functions

- `IdRef idConverter (IdRef const &ref) throw (QueryError)`
- bool `checkIdPair (BioObject obj, bool strict=true) const`
`compare cross reference to object in database compare all cross references to IdPair collection. If found, return true.`
- mongo::Query `idConverter (std::string const &ns, mongo::BSONObj const &bsonobj) throw (QueryError)`
`compare cross reference to object in database compare all cross references to IdPair collection. If found, return true.`
- void `updateIdPair (mongo::BSONObj const &bsonobj, bool isInsert=true) throw (UpdateError)`
`compare cross reference to object in database compare all cross references to IdPair collection. If found, return true.`
- void `update (std::string const &collectionNS, mongo::Query query, mongo::BSONObj bsonObj, bool upsert=false, bool multi=false) throw (UpdateError)`
`compare cross reference to object in database`
- void `insert (std::string const &collectionNS, mongo::BSONObj bsonObj) throw (UpdateError)`
`compare cross reference to object in database`
- void `remove (std::string const &collectionNS, mongo::Query q, bool justOne) throw (UpdateError)`
`compare cross reference to object in database`
- void `updateRelationAdd (Object const &obj, bool upsert=false, bool multi=false) throw (UpdateError)`
`compare cross reference to object in database`
- void `updateRelationDel (Object const &obj, bool isRemove=true, bool justOne=0, bool upsert=false, bool multi=false) throw (UpdateError)`
`compare cross reference to object in database`

Protected Attributes

- `Database * databaseHandle`

9.67.1 Constructor & Destructor Documentation

9.67.1.1 `unisys::Updater::Updater ()`

9.67.1.2 `unisys::Updater::Updater (Database * databaseHandlePt)`

9.67.2 Member Function Documentation

9.67.2.1 `bool unisys::Updater::checkIdPair (BioObject obj, bool strict=true) const [protected]`

`compare cross reference to object in database compare all cross references to IdPair collection. If found, return true.`

Parameters

<i>obj</i>	boson object of BioObject class
<i>strict</i>	compare all cross refernces and if obj is Interaction object, also compare with interactionKey

9.67.2.2 **IdRef unisys::Updater::idConverter (IdRef const & ref) throw (QueryError)** [protected]

9.67.2.3 **mongo::Query unisys::Updater::idConverter (std::string const & ns, mongo::BSONObj const & bsonobj) throw (QueryError)** [protected]

compare cross reference to object in database compare all cross references to IdPair collection. If found, return true.

Parameters

<i>ns</i>	boson object of BioObject class
<i>bsonobj</i>	compare all cross refernces and if obj is Interaction object, also compare with interactionKey

9.67.2.4 **void unisys::Updater::insert (std::string const & collectionNS, mongo::BSONObj bsonObj) throw (UpdateError)** [protected]

compare cross reference to object in database

Parameters

<i>collectionNS</i>	boson object of BioObject class
<i>bsonObj</i>	compare all cross refernces and if obj is Interaction object, also compare with interactionKey

9.67.2.5 **void unisys::Updater::insert (Tracking tracking) throw (UpdateError, DataError)**

9.67.2.6 **void unisys::Updater::insert (Ontology & ontology) throw (UpdateError, DataError)**

9.67.2.7 **void unisys::Updater::insert (PhysicalEntity & data, bool strict = true) throw (UpdateError, DataError)**

9.67.2.8 **void unisys::Updater::insert (Interaction & data, bool strict = true) throw (UpdateError, DataError)**

9.67.2.9 **void unisys::Updater::remove (std::string const & collectionNS, mongo::Query q, bool justOne) throw (UpdateError) [protected]**

compare cross reference to object in database

Parameters

<i>collectionNS</i>	boson object of BioObject class
<i>q</i>	compare all cross refernces and if obj is Interaction object, also compare with interactionKey
<i>justOne</i>	compare all cross refernces and if obj is Interaction object, also compare with interactionKey

9.67.2.10 **void unisys::Updater::remove (std::string const & collectionNS, std::string const & id, bool removeRela = true, bool removeProduct = true) throw (UpdateError)**

compare cross reference to object in database

Parameters

<i>collectionNS</i>	boson object of BioObject class
<i>id</i>	compare all cross references and if obj is Interaction object, also compare with interactionKey
<i>removeRela</i>	boson object of BioObject class
<i>removeProduct</i>	compare all cross references and if obj is Interaction object, also compare with interactionKey

9.67.2.11 void unisys::Updater::remove (std::string const & *collectionNS*, std::set< std::string > const & *ids*, bool *removeRela* = true, bool *removeProduct* = true) throw ([UpdateError](#))

compare cross reference to object in database

Parameters

<i>collectionNS</i>	boson object of BioObject class
<i>ids</i>	compare all cross references and if obj is Interaction object, also compare with interactionKey
<i>removeRela</i>	boson object of BioObject class
<i>removeProduct</i>	compare all cross references and if obj is Interaction object, also compare with interactionKey

9.67.2.12 void unisys::Updater::setDbHandle (Database * *databaseHandlePt*)

9.67.2.13 void unisys::Updater::update (std::string const & *collectionNS*, mongo::Query *query*, mongo::BSONObj *bsonObj*, bool *upsert* = false, bool *multi* = false) throw ([UpdateError](#)) [protected]

compare cross reference to object in database

Parameters

<i>collectionNS</i>	boson object of BioObject class
<i>query</i>	compare all cross references and if obj is Interaction object, also compare with interactionKey
<i>bsonObj</i>	boson object of BioObject class
<i>upsert</i>	compare all cross references and if obj is Interaction object, also compare with interactionKey
<i>multi</i>	compare all cross references and if obj is Interaction object, also compare with interactionKey

9.67.2.14 void unisys::Updater::update (Ontology const & *ontology*, bool *check* = true) throw ([UpdateError](#), [DataError](#))

compare cross reference to object in database

Parameters

<i>ontology</i>	boson object of BioObject class
<i>check</i>	compare all cross references and if obj is Interaction object, also compare with interactionKey

9.67.2.15 void unisys::Updater::update (PhysicalEntity & *physicalentity*, bool *check* = true) throw ([UpdateError](#), [DataError](#))

compare cross reference to object in database

Parameters

<i>physicalentity</i>	boson object of BioObject class
<i>check</i>	compare all cross references and if obj is Interaction object, also compare with interactionKey

9.67.2.16 void unisys::Updater::update (*Interaction & interaction*, bool *check* = true) throw (UpdateError, DataError)

compare cross reference to object in database

Parameters

<i>interaction</i>	boson object of BioObject class
<i>check</i>	compare all cross references and if obj is Interaction object, also compare with interactionKey

9.67.2.17 void unisys::Updater::updateIdPair (mongo::BSONObj const & *bsonobj*, bool *isInsert* = true) throw (UpdateError) [protected]

compare cross reference to object in database compare all cross references to IdPair collection. If found, return true.

Parameters

<i>bsonobj</i>	boson object of BioObject class
<i>isInsert</i>	compare all cross references and if obj is Interaction object, also compare with interactionKey

9.67.2.18 void unisys::Updater::updateRelationAdd (Object const & *obj*, bool *upsert* = false, bool *multi* = false) throw (UpdateError) [protected]

compare cross reference to object in database

Parameters

<i>obj</i>	boson object of BioObject class
<i>upsert</i>	compare all cross references and if obj is Interaction object, also compare with interactionKey
<i>multi</i>	boson object of BioObject class

9.67.2.19 void unisys::Updater::updateRelationDel (Object const & *obj*, bool *isRemove* = true, bool *justOne* = 0, bool *upsert* = false, bool *multi* = false) throw (UpdateError) [protected]

compare cross reference to object in database

Parameters

<i>obj</i>	boson object of BioObject class
<i>isRemove</i>	compare all cross references and if obj is Interaction object, also compare with interactionKey
<i>justOne</i>	boson object of BioObject class
<i>upsert</i>	compare all cross references and if obj is Interaction object, also compare with interactionKey
<i>multi</i>	boson object of BioObject class

9.67.3 Member Data Documentation

9.67.3.1 Database* unisys::Updater::databaseHandle [protected]

The documentation for this class was generated from the following file:

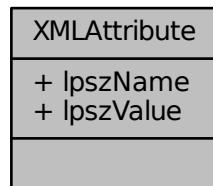
- [updater.h](#)

9.68 XMLAttribute Struct Reference

Structure for XML attribute.

```
#include <xmlParser.h>
```

Collaboration diagram for XMLAttribute:



Public Attributes

- [XMLCSTR lpszName](#)
- [XMLCSTR lpszValue](#)

9.68.1 Detailed Description

Structure for XML attribute.

9.68.2 Member Data Documentation

9.68.2.1 XMLCSTR XMLAttribute::lpszName

9.68.2.2 XMLCSTR XMLAttribute::lpszValue

The documentation for this struct was generated from the following file:

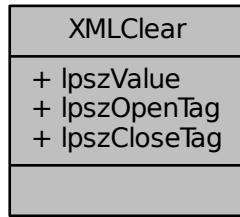
- [xmlParser.h](#)

9.69 XMLClear Struct Reference

Structure for XML clear (unformatted) node (usually comments)

```
#include <xmlParser.h>
```

Collaboration diagram for XMLClear:



Public Attributes

- [XMLCSTR lpszValue](#)
- [XMLCSTR lpszOpenTag](#)
- [XMLCSTR lpszCloseTag](#)

9.69.1 Detailed Description

Structure for XML clear (unformatted) node (usually comments)

9.69.2 Member Data Documentation

9.69.2.1 XMLCSTR XMLClear::lpszCloseTag

9.69.2.2 XMLCSTR XMLClear::lpszOpenTag

9.69.2.3 XMLCSTR XMLClear::lpszValue

The documentation for this struct was generated from the following file:

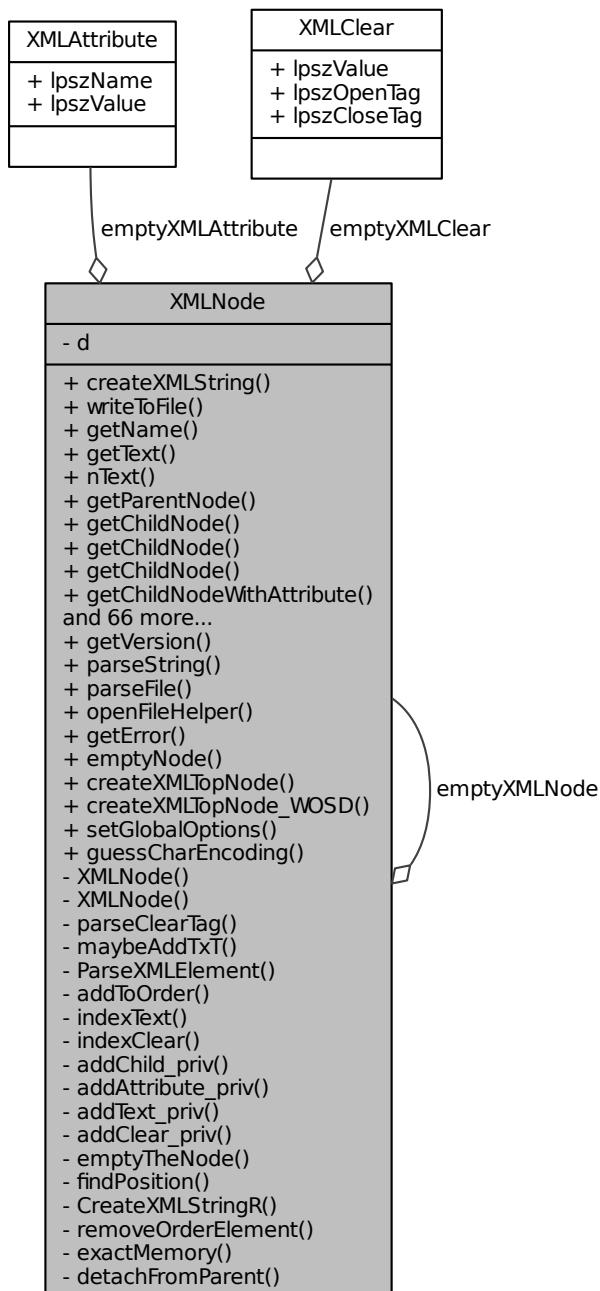
- [xmlParser.h](#)

9.70 XMLNode Struct Reference

Main Class representing a XML node.

```
#include <xmlParser.h>
```

Collaboration diagram for XMLNode:



Classes

- struct [XMLNodeDataTag](#)

Public Types

- enum `XMLCharEncoding` {
 `char_encoding_error` = 0, `char_encoding_UTF8` = 1, `char_encoding_legacy` = 2, `char_encoding_ShiftJIS` = 3,
`char_encoding_GB2312` = 4, `char_encoding_Big5` = 5, `char_encoding_GBK` = 6 }

Enumeration for XML character encoding.
- typedef enum
`XMLNode::XMLCharEncoding XMLCharEncoding`

Enumeration for XML character encoding.

Public Member Functions

- `XMLSTR createXMLString` (int nFormat=1, int *pnSize=NULL) const

Create an XML string starting from the current `XMLNode`.
- `XMLError writeToFile` (`XMLCSTR` filename, const char *encoding=NULL, char nFormat=1) const

Save the content of an `xmlNode` inside a file.
- `XMLCSTR getName` () const

name of the node
- `XMLCSTR getText` (int i=0) const

return ith text field
- int `nText` () const

nbr of text field
- `XMLNode getParentNode` () const

return the parent node
- `XMLNode getChildNode` (int i=0) const

return ith child node
- `XMLNode getChildNode` (`XMLCSTR` name, int i) const

return ith child node with specific name (return an empty node if failing). If i== -1, this returns the last `XMLNode` with the given name.
- `XMLNode getChildNode` (`XMLCSTR` name, int *i=NULL) const

return next child node with specific name (return an empty node if failing)
- `XMLNode getChildNodeWithAttribute` (`XMLCSTR` tagName, `XMLCSTR` attributeName, `XMLCSTR` attributeValue=NULL, int *i=NULL) const

return child node with specific name/attribute (return an empty node if failing)
- `XMLNode getChildNodeByPath` (`XMLCSTR` path, char createNodeIfMissing=0, `XMLCHAR` sep='/')

return the first child node with specific path
- `XMLNode getChildNodeByPathNonConst` (`XMLSTR` path, char createNodeIfMissing=0, `XMLCHAR` sep='/')

return the first child node with specific path.
- int `nChildNode` (`XMLCSTR` name) const

return the number of child node with specific name
- int `nChildNode` () const

nbr of child node
- `XMLAttribute getAttribute` (int i=0) const

return ith attribute
- `XMLCSTR getAttributeName` (int i=0) const

return ith attribute name
- `XMLCSTR getAttributeValue` (int i=0) const

return ith attribute value
- char `isAttributeSet` (`XMLCSTR` name) const

test if an attribute with a specific name is given
- `XMLCSTR getAttribute` (`XMLCSTR` name, int i) const

- `return ith attribute content with specific name (return a NULL if failing)`
- `XMLCSTR getAttribute (XMLCSTR name, int *i=NULL) const`
`return next attribute content with specific name (return a NULL if failing)`
- `int nAttribute () const`
`nbr of attribute`
- `XMLClear getClear (int i=0) const`
`return ith clear field (comments)`
- `int nClear () const`
`nbr of clear field`
- `XMLNodeContents enumContents (XMLElementPosition i) const`
`enumerate all the different contents (attribute,child,text, clear) of the current XMLNode. The order is reflecting the order of the original file/string. NOTE: 0 <= i < nElement();`
- `int nElement () const`
`nbr of different contents for current node`
- `char isEmpty () const`
`is this node Empty?`
- `char isDeclaration () const`
`is this node a declaration <? ?>`
- `XMLNode deepCopy () const`
`deep copy (duplicate/clone) a XMLNode`
- `~XMLNode ()`
- `XMLNode (const XMLNode &A)`
`to allow shallow/fast copy:`
- `XMLNode & operator= (const XMLNode &A)`
`to allow shallow/fast copy:`
- `XMLNode ()`
- `XMLNode addChild (XMLCSTR lpszName, char isDeclaration=FALSE, XMLElementPosition pos=-1)`
`Add a new child node.`
- `XMLNode addChild (XMLNode nodeToAdd, XMLElementPosition pos=-1)`
`If the "nodeToAdd" has some parents, it will be detached from its parents before being attached to the current XMLNode.`
- `XMLAttribute * addAttribute (XMLCSTR lpszName, XMLCSTR lpszValuev)`
`Add a new attribute.`
- `XMLCSTR addText (XMLCSTR lpszValue, XMLElementPosition pos=-1)`
`Add a new text content.`
- `XMLClear * addClear (XMLCSTR lpszValue, XMLCSTR lpszOpen=NULL, XMLCSTR lpszClose=NULL, XMLElementPosition pos=-1)`
- `XMLCSTR updateName (XMLCSTR lpszName)`
`change node's name`
- `XMLAttribute * updateAttribute (XMLAttribute *newAttribute, XMLAttribute *oldAttribute)`
`if the attribute to update is missing, a new one will be added`
- `XMLAttribute * updateAttribute (XMLCSTR lpsznewValue, XMLCSTR lpszNewName=NULL, int i=0)`
`if the attribute to update is missing, a new one will be added`
- `XMLAttribute * updateAttribute (XMLCSTR lpsznewValue, XMLCSTR lpszNewName, XMLCSTR lpszOldName)`
`set lpszNewName=NULL if you don't want to change the name of the attribute if the attribute to update is missing, a new one will be added`
- `XMLCSTR updateText (XMLCSTR lpsznewValue, int i=0)`
`if the text to update is missing, a new one will be added`
- `XMLCSTR updateText (XMLCSTR lpsznewValue, XMLCSTR lpszOldValue)`
`if the text to update is missing, a new one will be added`
- `XMLClear * updateClear (XMLCSTR lpszNewContent, int i=0)`

- if the clearTag to update is missing, a new one will be added
- `XMLClear * updateClear (XMLClear *newP, XMLClear *oldP)`
if the clearTag to update is missing, a new one will be added
- `XMLClear * updateClear (XMLCSTR lpsznewValue, XMLCSTR lpsz oldValue)`
if the clearTag to update is missing, a new one will be added
- void `deleteNodeContent ()`
The "deleteNodeContent" function forces the deletion of the content of this [XMLNode](#) and the subtree.
- void `deleteAttribute (int i=0)`
Delete the ith attribute of the current [XMLNode](#).
- void `deleteAttribute (XMLCSTR lpszName)`
Delete the attribute with the given name (the "strcmp" function is used to find the right attribute)
- void `deleteAttribute (XMLAttribute *anAttribute)`
Delete the attribute with the name "anAttribute->lpszName" (the "strcmp" function is used to find the right attribute)
- void `deleteText (int i=0)`
Delete the ith text content of the current [XMLNode](#).
- void `deleteText (XMLCSTR lpszValue)`
Delete the text content "lpszValue" inside the current [XMLNode](#) (direct "pointer-to-pointer" comparison is used to find the right text)
- void `deleteClear (int i=0)`
Delete the ith clear tag inside the current [XMLNode](#).
- void `deleteClear (XMLCSTR lpszValue)`
Delete the clear tag "lpszValue" inside the current [XMLNode](#) (direct "pointer-to-pointer" comparison is used to find the clear tag)
- void `deleteClear (XMLClear *p)`
Delete the clear tag "p" inside the current [XMLNode](#) (direct "pointer-to-pointer" comparison on the lpszName of the clear tag is used to find the clear tag)
- `XMLNode addChild_WOSD (XMLSTR lpszName, char isDeclaration=FALSE, XMLElementPosition pos=-1)`
Add a new child node.
- `XMLAttribute * addAttribute_WOSD (XMLSTR lpszName, XMLSTR lpszValue)`
Add a new attribute.
- `XMLCSTR addText_WOSD (XMLSTR lpszValue, XMLElementPosition pos=-1)`
Add a new text content.
- `XMLClear * addClear_WOSD (XMLSTR lpszValue, XMLCSTR lpszOpen=NULL, XMLCSTR lpszClose=NULL, XMLElementPosition pos=-1)`
Add a new clear Tag.
- `XMLCSTR updateName_WOSD (XMLSTR lpszName)`
change node's name
- `XMLAttribute * updateAttribute_WOSD (XMLAttribute *newAttribute, XMLAttribute *oldAttribute)`
if the attribute to update is missing, a new one will be added
- `XMLAttribute * updateAttribute_WOSD (XMLSTR lpsznewValue, XMLSTR lpsz newName=NULL, int i=0)`
if the attribute to update is missing, a new one will be added
- `XMLAttribute * updateAttribute_WOSD (XMLSTR lpsznewValue, XMLSTR lpsz newName, XMLCSTR lpszOldName)`
set lpsz newName=NULL if you don't want to change the name of the attribute if the attribute to update is missing, a new one will be added
- `XMLCSTR updateText_WOSD (XMLSTR lpsznewValue, int i=0)`
if the text to update is missing, a new one will be added
- `XMLCSTR updateText_WOSD (XMLSTR lpsznewValue, XMLCSTR lpsz oldValue)`
if the text to update is missing, a new one will be added
- `XMLClear * updateClear_WOSD (XMLSTR lpszNewContent, int i=0)`
if the clearTag to update is missing, a new one will be added

- `XMLClear * updateClear_WOSD (XMLClear *newP, XMLClear *oldP)`
if the clearTag to update is missing, a new one will be added
- `XMLClear * updateClear_WOSD (XMLSTR lpsznewValue, XMLCSTR lpszOldValue)`
if the clearTag to update is missing, a new one will be added
- `XMLElementPosition positionOfText (int i=0) const`
- `XMLElementPosition positionOfText (XMLCSTR lpszValue) const`
- `XMLElementPosition positionOfClear (int i=0) const`
- `XMLElementPosition positionOfClear (XMLCSTR lpszValue) const`
- `XMLElementPosition positionOfClear (XMLClear *a) const`
- `XMLElementPosition positionOfChildNode (int i=0) const`
- `XMLElementPosition positionOfChildNode (XMLNode x) const`
- `XMLElementPosition positionOfChildNode (XMLCSTR name, int i=0) const`
*return the position of the ith childNode with the specified name if (name==NULL) return the position of the ith child-
Node*

Static Public Member Functions

- static `XMLCSTR getVersion ()`
Return the XMLParser library version number.
- static `XMLNode parseString (XMLCSTR lpXMLString, XMLCSTR tag=NULL, XMLResults *pResults=NULL)`
Parse an XML string and return the root of a `XMLNode` tree representing the string.
- static `XMLNode parseFile (XMLCSTR filename, XMLCSTR tag=NULL, XMLResults *pResults=NULL)`
Parse an XML file and return the root of a `XMLNode` tree representing the file.
- static `XMLNode openFileHelper (XMLCSTR filename, XMLCSTR tag=NULL)`
*Parse an XML file and return the root of a `XMLNode` tree representing the file. A very crude error checking is made.
An attempt to guess the Char Encoding used in the file is made.*
- static `XMLCSTR getError (XMLError error)`
this gives you a user-friendly explanation of the parsing error
- static `XMLNode emptyNode ()`
return `XMLNode::emptyXMLNode`;
- static `XMLNode createXMLTopNode (XMLCSTR lpszName, char isDeclaration=FALSE)`
Create the top node of an `XMLNode` structure.
- static `XMLNode createXMLTopNode_WOSD (XMLSTR lpszName, char isDeclaration=FALSE)`
Create the top node of an `XMLNode` structure.
- static `char setGlobalOptions (XMLCharEncoding characterEncoding=XMLNode::char_encoding_UTF8, char guessWideCharChars=1, char dropWhiteSpace=1, char removeCommentsInMiddleOfText=1)`
Sets the global options for the conversions.
- static `XMLCharEncoding guessCharEncoding (void *buffer, int bufLen, char useXMLEncodingAttribute=1)`
Guess the character encoding of the string (ascii, utf8 or shift-JIS)

Static Public Attributes

- static `XMLNode emptyXMLNode`
- static `XMLClear emptyXMLClear`
- static `XMLAttribute emptyXMLAttribute`

Private Types

- `typedef struct
XMLNode::XMLNodeDataTag XMLNodeData`

Private Member Functions

- `XMLNode` (struct `XMLNodeDataTag` *`pParent`, `XMLSTR` `lpszName`, char `isDeclaration`)
Constructors are protected, so use instead one of: `XMLNode::parseString`, `XMLNode::parseFile`, `XMLNode::openFileHelper`, `XMLNode::createXMLTopNode`.
- `XMLNode` (struct `XMLNodeDataTag` *`p`)
Constructors are protected, so use instead one of: `XMLNode::parseString`, `XMLNode::parseFile`, `XMLNode::openFileHelper`, `XMLNode::createXMLTopNode`.
- `char parseClearTag` (`void` *`px`, `void` *`pa`)
- `char maybeAddTxt` (`void` *`pa`, `XMLCSTR` `tokenPStr`)
- `int ParseXMLElement` (`void` *`pXML`)
- `void * addToOrder` (`int` `memInc`, `int` *`_pos`, `int` `nc`, `void` *`p`, `int` `size`, `XMLElementType` `xtype`)
- `int indexText` (`XMLCSTR` `lpszValue`) const
- `int indexClear` (`XMLCSTR` `lpszValue`) const
- `XMLNode addChild_priv` (`int`, `XMLSTR`, char, `int`)
- `XMLAttribute * addAttribute_priv` (`int`, `XMLSTR`, `XMLSTR`)
- `XMLCSTR addText_priv` (`int`, `XMLSTR`, `int`)
- `XMLClear * addClear_priv` (`int`, `XMLSTR`, `XMLCSTR`, `XMLCSTR`, `int`)
- `void emptyTheNode` (char `force`)

Static Private Member Functions

- static `XMLElementPosition findPosition` (`XMLNodeData` *`d`, `int` `index`, `XMLElementType` `xtype`)
- static `int CreateXMLStringR` (`XMLNodeData` *`pEntry`, `XMLSTR` `lpszMarker`, `int` `nFormat`)
- static `int removeOrderElement` (`XMLNodeData` *`d`, `XMLElementType` `t`, `int` `index`)
- static `void exactMemory` (`XMLNodeData` *`d`)
- static `int detachFromParent` (`XMLNodeData` *`d`)

Private Attributes

- `XMLNodeData` * `d`

9.70.1 Detailed Description

Main Class representing a XML node.

All operations are performed using this class.

Note

The constructors of the `XMLNode` class are protected, so use instead one of these four methods to get your first instance of `XMLNode`:

- `XMLNode::parseString`
- `XMLNode::parseFile`
- `XMLNode::openFileHelper`
- `XMLNode::createXMLTopNode` (or `XMLNode::createXMLTopNode_WOSD`)

9.70.2 Member Typedef Documentation

9.70.2.1 `typedef enum XMLNode::XMLCharEncoding XMLNode::XMLCharEncoding`

Enumeration for XML character encoding.

9.70.2.2 `typedef struct XMLNode::XMLNodeDataTag XMLNode::XMLNodeData [private]`

9.70.3 Member Enumeration Documentation

9.70.3.1 `enum XMLNode::XMLCharEncoding`

Enumeration for XML character encoding.

Enumerator:

- `char_encoding_error`
- `char_encoding_UTF8`
- `char_encoding_legacy`
- `char_encoding_ShiftJIS`
- `char_encoding_GB2312`
- `char_encoding_Big5`
- `char_encoding_GBK`

9.70.4 Constructor & Destructor Documentation

9.70.4.1 `XMLNode::XMLNode (struct XMLNodeDataTag * pParent, XMLSTR lpszName, char isDeclaration) [private]`

Constructors are protected, so use instead one of: [XMLNode::parseString](#), [XMLNode::parseFile](#), [XMLNode::openFileHelper](#), [XMLNode::createXMLTopNode](#).

9.70.4.2 `XMLNode::XMLNode (struct XMLNodeDataTag * p) [private]`

Constructors are protected, so use instead one of: [XMLNode::parseString](#), [XMLNode::parseFile](#), [XMLNode::openFileHelper](#), [XMLNode::createXMLTopNode](#).

9.70.4.3 `XMLNode::~XMLNode ()`

9.70.4.4 `XMLNode::XMLNode (const XMLNode & A)`

to allow shallow/fast copy:

9.70.4.5 `XMLNode::XMLNode () [inline]`

9.70.5 Member Function Documentation

9.70.5.1 `XMLAttribute* XMLNode::addAttribute_priv (int , XMLSTR , XMLSTR) [private]`

9.70.5.2 `XMLNode XMLNode::addChild_priv (int , XMLSTR , char , int) [private]`

9.70.5.3 `XMLClear* XMLNode::addClear_priv (int , XMLSTR , XMLCSTR , XMLCSTR , int) [private]`

9.70.5.4 `XMLCSTR XMLNode::addText_priv (int , XMLSTR , int) [private]`

9.70.5.5 `void* XMLNode::addToOrder (int memInc, int * pos, int nc, void * p, int size, XMLElementType xtype) [private]`

9.70.5.6 static int XMLNode::CreateXMLStringR (**XMLNodeData** * *pEntry*, **XMLSTR** *lpszMarker*, int *nFormat*)
 [static], [private]

9.70.5.7 static int XMLNode::detachFromParent (**XMLNodeData** * *d*) [static], [private]

9.70.5.8 void XMLNode::emptyTheNode (char *force*) [private]

9.70.5.9 static void XMLNode::exactMemory (**XMLNodeData** * *d*) [static], [private]

9.70.5.10 static **XMLElementPosition** XMLNode::findPosition (**XMLNodeData** * *d*, int *index*, **XMLElementType** *xtype*) [inline], [static], [private]

9.70.5.11 static **XMLCSTR** XMLNode::getVersion () [static]

Return the XMLParser library version number.

9.70.5.12 int XMLNode::indexClear (**XMLCSTR** *lpszValue*) const [private]

9.70.5.13 int XMLNode::indexText (**XMLCSTR** *lpszValue*) const [private]

9.70.5.14 char XMLNode::maybeAddTxt (void * *pa*, **XMLCSTR** *tokenPStr*) [private]

9.70.5.15 **XMLNode&** XMLNode::operator= (const **XMLNode &** *A*)

to allow shallow/fast copy:

9.70.5.16 char XMLNode::parseClearTag (void * *px*, void * *pa*) [private]

9.70.5.17 int XMLNode::ParseXMLElement (void * *pXML*) [private]

9.70.5.18 static int XMLNode::removeOrderElement (**XMLNodeData** * *d*, **XMLElementType** *t*, int *index*) [static], [private]

9.70.6 Member Data Documentation

9.70.6.1 **XMLNodeData*** XMLNode::*d* [private]

9.70.6.2 **XMLAttribute** XMLNode::emptyXMLAttribute [static]

9.70.6.3 **XMLClear** XMLNode::emptyXMLClear [static]

9.70.6.4 **XMLNode** XMLNode::emptyXMLNode [static]

The documentation for this struct was generated from the following file:

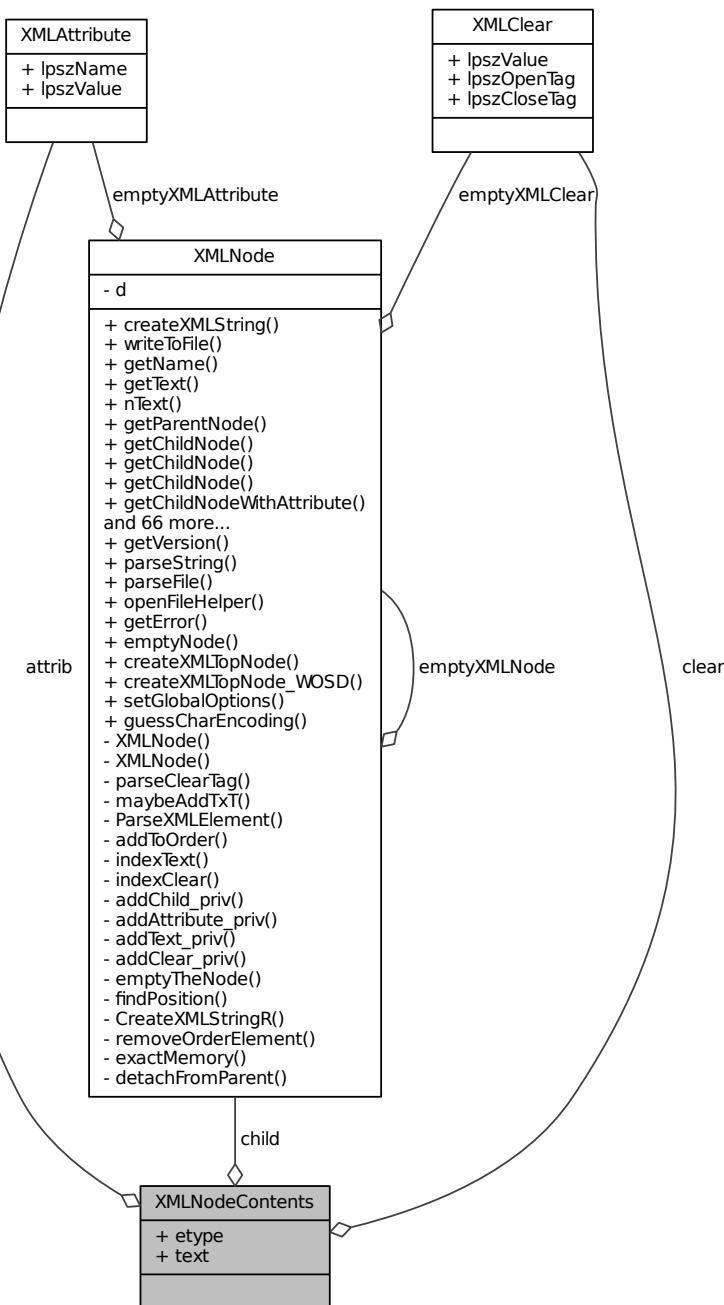
- [xmlParser.h](#)

9.71 XMLNodeContents Struct Reference

This structure is given by the function [XMLNode::enumContents](#).

```
#include <xmlParser.h>
```

Collaboration diagram for XMLNodeContents:



Public Attributes

- enum [XMLElementType etype](#)
This dictates what's the content of the XMLNodeContent.
- [XMLNode child](#)
- [XmlAttribute attrib](#)
- [XMLCSTR text](#)
- [XMLClear clear](#)

9.71.1 Detailed Description

This structure is given by the function [XMLNode::enumContents](#).

9.71.2 Member Data Documentation

9.71.2.1 **XMLAttribute XMLNodeContents::attrib**

9.71.2.2 **XMLNode XMLNodeContents::child**

9.71.2.3 **XMLClear XMLNodeContents::clear**

9.71.2.4 **enum XMLElementType XMLNodeContents::etype**

This dictates what's the content of the XMLNodeContent.

should be an union to access the appropriate data. Compiler does not allow union of object with constructor... too bad.

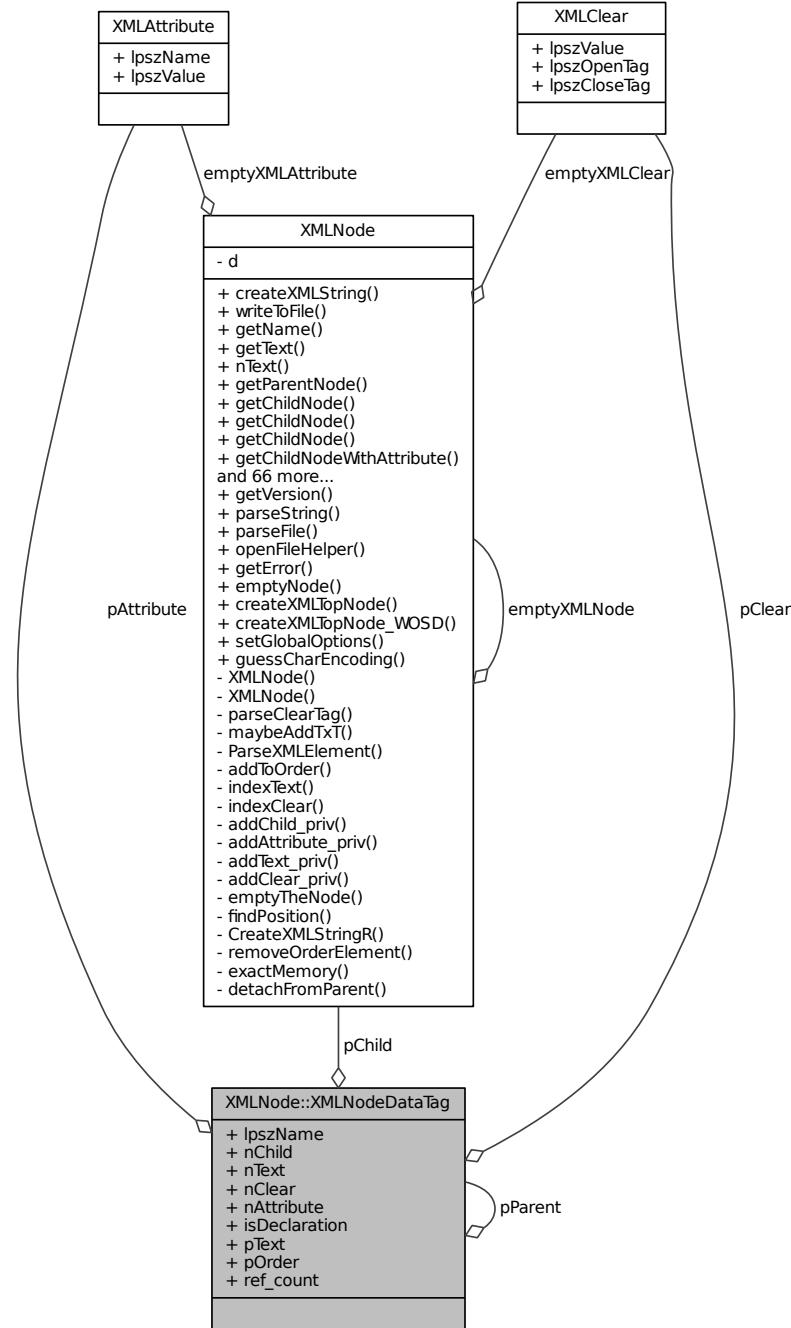
9.71.2.5 **XMLCSTR XMLNodeContents::text**

The documentation for this struct was generated from the following file:

- [xmlParser.h](#)

9.72 XMLNode::XMLNodeDataTag Struct Reference

Collaboration diagram for XMLNode::XMLNodeDataTag:



Public Attributes

- `XMLCSTR lpszName`
- `int nChild`
- `int nText`

- int `nClear`
- int `nAttribute`
- char `isDeclaration`
- struct `XMLNodeDataTag` * `pParent`
- `XMLNode` * `pChild`
- `XMLCSTR` * `pText`
- `XMLClear` * `pClear`
- `XMLAttribute` * `pAttribute`
- int * `pOrder`
- int `ref_count`

9.72.1 Member Data Documentation

9.72.1.1 char `XMLNode::XMLNodeDataTag::isDeclaration`

9.72.1.2 `XMLCSTR XMLNode::XMLNodeDataTag::lpszName`

9.72.1.3 int `XMLNode::XMLNodeDataTag::nAttribute`

9.72.1.4 int `XMLNode::XMLNodeDataTag::nChild`

9.72.1.5 int `XMLNode::XMLNodeDataTag::nClear`

9.72.1.6 int `XMLNode::XMLNodeDataTag::nText`

9.72.1.7 `XMLAttribute* XMLNode::XMLNodeDataTag::pAttribute`

9.72.1.8 `XMLNode* XMLNode::XMLNodeDataTag::pChild`

9.72.1.9 `XMLClear* XMLNode::XMLNodeDataTag::pClear`

9.72.1.10 int* `XMLNode::XMLNodeDataTag::pOrder`

9.72.1.11 struct `XMLNodeDataTag* XMLNode::XMLNodeDataTag::pParent`

9.72.1.12 `XMLCSTR* XMLNode::XMLNodeDataTag::pText`

9.72.1.13 int `XMLNode::XMLNodeDataTag::ref_count`

The documentation for this struct was generated from the following file:

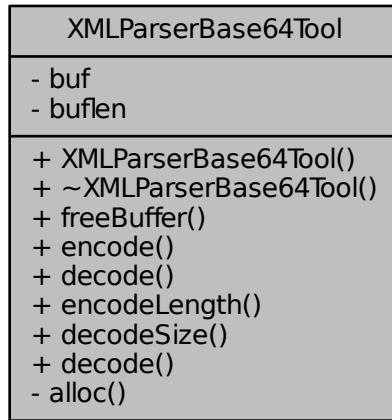
- `xmlParser.h`

9.73 XMLParserBase64Tool Struct Reference

Helper class to include binary data inside XML strings using "Base64 encoding".

```
#include <xmlParser.h>
```

Collaboration diagram for XMLParserBase64Tool:



Public Member Functions

- [XMLParserBase64Tool \(\)](#)
- [~XMLParserBase64Tool \(\)](#)
- void [freeBuffer \(\)](#)
Call this function when you have finished using this object to release memory used by the internal buffer.
- [XMLSTR encode \(unsigned char *inByteBuf, unsigned int inByteLen, char formatted=0\)](#)
returns a pointer to an internal buffer containing the base64 string containing the binary data encoded from "inByteBuf"
- unsigned char * [decode \(XMLCSTR inString, int *outByteLen=NULL, XMLError *xe=NULL\)](#)
returns a pointer to an internal buffer containing the binary data decoded from "inString"

Static Public Member Functions

- static int [encodeLength \(int inBufLen, char formatted=0\)](#)
return the length of the base64 string that encodes a data buffer of size inBufLen bytes.
- static unsigned int [decodeSize \(XMLCSTR inString, XMLError *xe=NULL\)](#)
returns the number of bytes which will be decoded from "inString".
- static unsigned char [decode \(XMLCSTR inString, unsigned char *outByteBuf, int inMaxByteOutBuflen, XMLError *xe=NULL\)](#)
deprecated.

Private Member Functions

- void [alloc \(int newsize\)](#)

Private Attributes

- void * [buf](#)
- int [buflen](#)

9.73.1 Detailed Description

Helper class to include binary data inside XML strings using "Base64 encoding".

The "XMLParserBase64Tool" class allows you to include any binary data (images, sounds,...) into an XML document using "Base64 encoding". This class is completely separated from the rest of the xmlParser library and can be removed without any problem. To include some binary data into an XML file, you must convert the binary data into standard text (using "encode"). To retrieve the original binary data from the b64-encoded text included inside the XML file, use "decode". Alternatively, these functions can also be used to "encrypt/decrypt" some critical data contained inside the XML (it's not a strong encryption at all, but sometimes it can be useful).

9.73.2 Constructor & Destructor Documentation

9.73.2.1 `XMLParserBase64Tool::XMLParserBase64Tool() [inline]`

9.73.2.2 `XMLParserBase64Tool::~XMLParserBase64Tool()`

9.73.3 Member Function Documentation

9.73.3.1 `void XMLParserBase64Tool::alloc(int newsize) [private]`

9.73.3.2 `unsigned char* XMLParserBase64Tool::decode(XMLCSTR inString, int * outByteLen = NULL, XMLError * xe = NULL)`

returns a pointer to an internal buffer containing the binary data decoded from "inString"

The "decode" function returns a pointer to a buffer containing the binary data decoded from "inString". The output buffer will be free'd when the `XMLParserBase64Tool` object is deleted. All output buffer are sharing the same memory space.

Parameters

<code>inString</code>	If "instring" is malformed, NULL will be returned
<code>outByteLen</code>	
<code>xe</code>	

9.73.3.3 `static unsigned char XMLParserBase64Tool::decode(XMLCSTR inString, unsigned char * outByteBuf, int inMaxByteOutBufLen, XMLError * xe = NULL) [static]`

deprecated.

decodes data from "inString" to "outByteBuf". You need to provide the size (in byte) of "outByteBuf" in "inMaxByteOutBufLen". If "outByteBuf" is not large enough or if data is malformed, then "FALSE" will be returned; otherwise "TRUE".

9.73.3.4 `static unsigned int XMLParserBase64Tool::decodeSize(XMLCSTR inString, XMLError * xe = NULL) [static]`

returns the number of bytes which will be decoded from "inString".

9.73.3.5 `XMLSTR XMLParserBase64Tool::encode(unsigned char * inByteBuf, unsigned int inByteLen, char formatted = 0)`

returns a pointer to an internal buffer containing the base64 string containing the binary data encoded from "inByteBuf"

The "base64Encode" function returns a string containing the base64 encoding of "inByteLen" bytes from "inByteBuf". If "formatted" parameter is true, then there will be a carriage-return every 72 chars. The string will be free'd when the [XMLParserBase64Tool](#) object is deleted. All returned strings are sharing the same memory space.

9.73.3.6 static int XMLParserBase64Tool::encodeLength (int *inBufLen*, char *formatted* = 0) [static]

return the length of the base64 string that encodes a data buffer of size *inBufLen* bytes.

Parameters

<i>formatted</i>	If "formatted"=true, some space will be reserved for a carriage-return every 72 chars.
<i>inBufLen</i>	

9.73.3.7 void XMLParserBase64Tool::freeBuffer ()

Call this function when you have finished using this object to release memory used by the internal buffer.

9.73.4 Member Data Documentation

9.73.4.1 void* XMLParserBase64Tool::buf [private]

9.73.4.2 int XMLParserBase64Tool::buflen [private]

The documentation for this struct was generated from the following file:

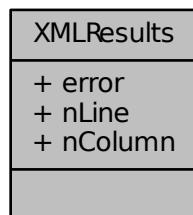
- [xmlParser.h](#)

9.74 XMLResults Struct Reference

Structure used to obtain error details if the parse fails.

```
#include <xmlParser.h>
```

Collaboration diagram for XMLResults:



Public Attributes

- enum [XMLError](#) error

- int [nLine](#)
- int [nColumn](#)

9.74.1 Detailed Description

Structure used to obtain error details if the parse fails.

9.74.2 Member Data Documentation

9.74.2.1 enum XMLError [XMLResults::error](#)

9.74.2.2 int [XMLResults::nColumn](#)

9.74.2.3 int [XMLResults::nLine](#)

The documentation for this struct was generated from the following file:

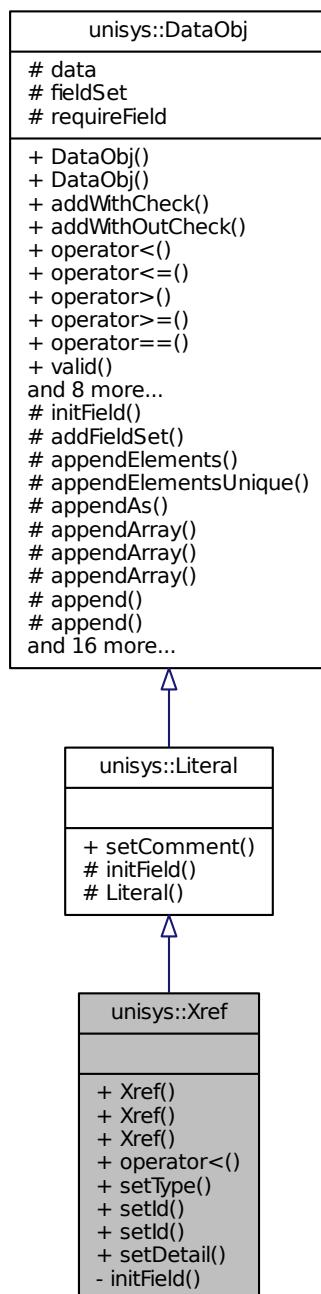
- [xmlParser.h](#)

9.75 unisys::Xref Class Reference

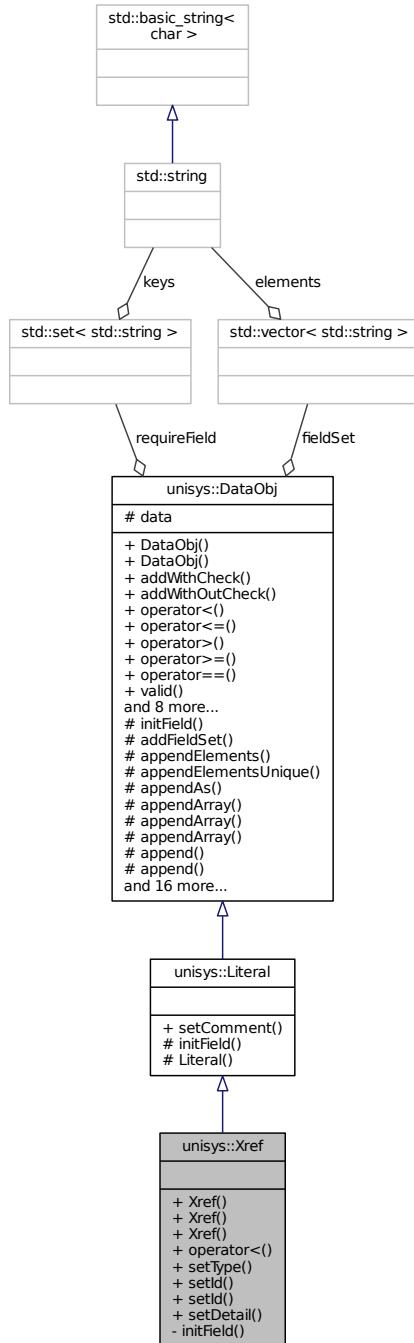
The C++ representative class for Cross reference data class.

```
#include <LitClass.h>
```

Inheritance diagram for unisys::Xref:



Collaboration diagram for unisys::Xref:



Public Member Functions

- [Xref \(\)](#)
Default constructor.
- [Xref \(mongo::BSONObj const &bsonObj, bool isDatabase=true\)](#)
Overloaded constructor is used when retrieving data in bson object from database and transform to C++ object.
- [Xref \(std::string const &uri, std::string const &detail="", bool isDatabase=true\)](#)

- Overloaded constructor.*
- `bool operator< (Xref const &other) const`
Less than operator overload.
 - `void setType (bool isDatabase=true)`
Less than operator overload.
 - `void setId (Miriam const &miriam, bool withVer=true)`
Define cross reference ID with [Miriam](#) object.
 - `void setId (std::string const &uri, std::string const &ns="", unsigned int version=0, bool withVer=true)`
Define cross reference ID.
 - `void setDetail (std::string const &detail)`
define some detail of this cross reference

Private Member Functions

- `void initField ()`
function for init field in the object

Additional Inherited Members

9.75.1 Detailed Description

The C++ representative class for Cross reference data class.

```
BSON structure:
{
    type: <database|publication>,
    comment: <string>,
    id: <stringIdOfXref>,
    detail: <string>,
}
```

9.75.2 Constructor & Destructor Documentation

9.75.2.1 unisys::Xref::Xref()

Default constructor.

9.75.2.2 unisys::Xref::Xref(mongo::BSONObj const & bsonObj, bool isDatabase = true)

Overloaded constructor is used when retrieving data in bson object from database and transform to C++ object.

9.75.2.3 unisys::Xref::Xref(std::string const & uri, std::string const & detail = "", bool isDatabase = true)

Overloaded constructor.

9.75.3 Member Function Documentation

9.75.3.1 void unisys::Xref::initField() [private], [virtual]

function for init field in the object

Reimplemented from [unisys::Literal](#).

9.75.3.2 `bool unisys::Xref::operator< (Xref const & other) const`

Less than operator overload.

9.75.3.3 `void unisys::Xref::setDetail (std::string const & detail)`

define some detail of this cross reference

9.75.3.4 `void unisys::Xref::setId (Miriam const & miriam, bool withVer = true)`

Define cross reference ID with [Miriam](#) object.

9.75.3.5 `void unisys::Xref::setId (std::string const & uri, std::string const & ns = "", unsigned int version = 0, bool withVer = true)`

Define cross reference ID.

9.75.3.6 `void unisys::Xref::setType (bool isDatabase = true)`

Less than operator overload.

The documentation for this class was generated from the following file:

- [LitClass.h](#)

Chapter 10

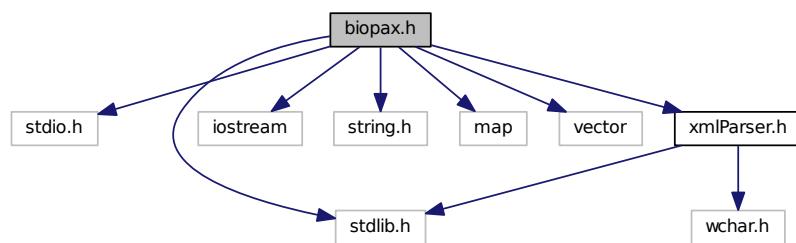
File Documentation

10.1 biopax.h File Reference

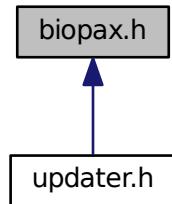
It contain BIOPAX class using for parsing OWL (Web Ontology Language) format especially BIOPAX.

```
#include <stdio.h>
#include <stdlib.h>
#include <iostream>
#include <string.h>
#include <map>
#include <vector>
#include "xmlParser.h"
```

Include dependency graph for biopax.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [unisys::BIOPAX_obj](#)
This class is used to parse OWL format.
- class [unisys::BIOPAX2](#)
This class is used to parse OWL format.

Namespaces

- namespace [unisys](#)
This namespace.

Macros

- `#define _UNISYSDB_BIOPAX_H_`

10.1.1 Detailed Description

It contain BIOPAX class using for parsing OWL (Web Ontology Language) format especially BIOPAX.

10.1.2 Macro Definition Documentation

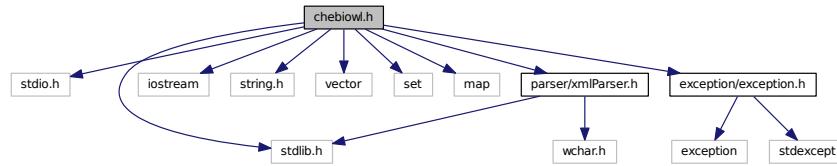
10.1.2.1 `#define _UNISYSDB_BIOPAX_H_`

10.2 chebiowl.h File Reference

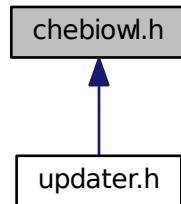
```

#include <stdio.h>
#include <stdlib.h>
#include <iostream>
#include <string.h>
#include <vector>
#include <set>
#include <map>
#include "parser/xmlParser.h"
#include "exception/exception.h"
  
```

Include dependency graph for chebiowl.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [unisys::ChEBIOWLClass](#)
To handle owl:class tag of ChEBI OWL format.
- class [unisys::ChEBIOWL](#)
Used to parse ChEBI OWL format.

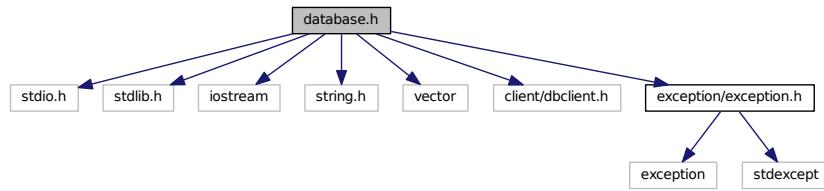
Namespaces

- namespace [unisys](#)
This namespace.

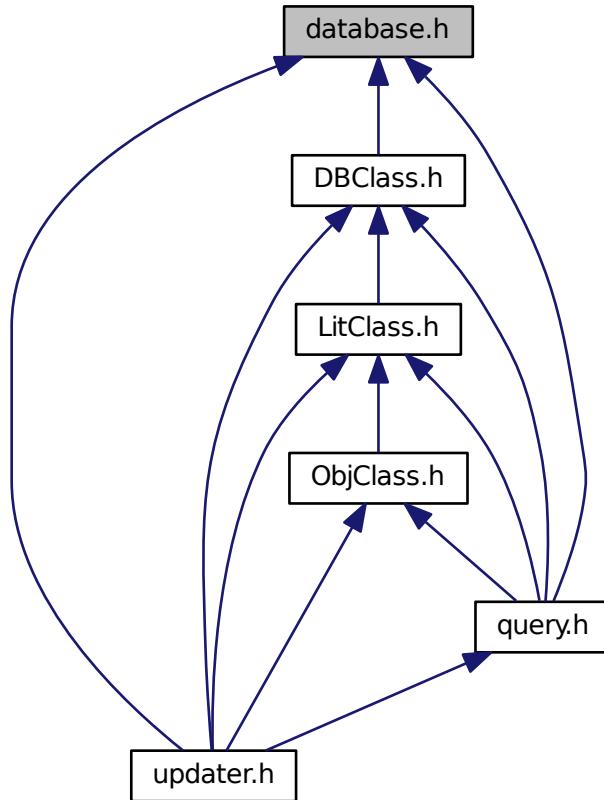
10.3 database.h File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <iostream>
#include <string.h>
#include <vector>
#include "client/dbclient.h"
#include "exception/exception.h"
```

Include dependency graph for database.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [unisys::Database](#)
This class is a database handle class.

Namespaces

- namespace [unisys](#)

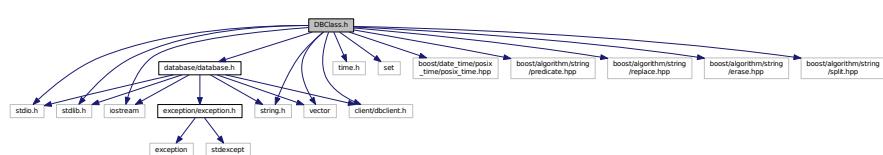
This namespace.

10.4 DBClass.h File Reference

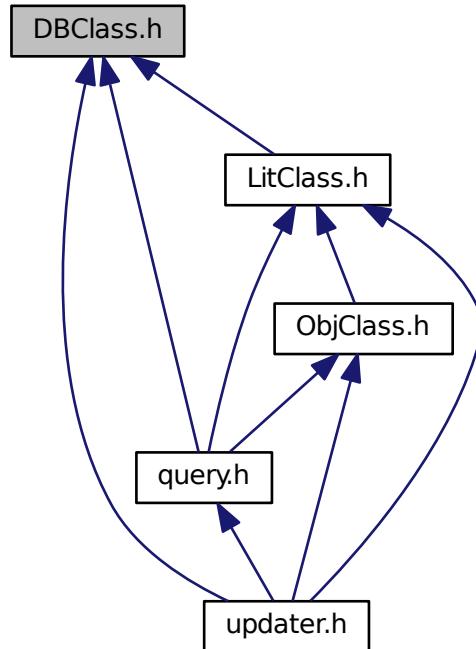
It contain Miriam, IdRef, Tracking and LiteralBSON class using for formating data to database structure.

```
#include <stdio.h>
#include <stdlib.h>
#include <iostream>
#include <string.h>
#include <time.h>
#include <vector>
#include <set>
#include <boost/date_time posix_time posix_time.hpp>
#include <boost/algorithm/string/predicate.hpp>
#include <boost/algorithm/string/replace.hpp>
#include <boost/algorithm/string/erase.hpp>
#include <boost/algorithm/string/split.hpp>
#include "client/dbclient.h"
#include "database/database.h"
```

Include dependency graph for DBClass.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [unisys::DataObj](#)
- class [unisys::Miriam](#)
This class is for miriam cross reference annotation.
- class [unisys::IdRef](#)
Identifier reference class.
- class [unisys::PEIdRef](#)
Identifier reference class specific to physicalentity collection namespace.
- class [unisys::IntIdRef](#)
Identifier reference class specific to interaction collection namespace.
- class [unisys::OntIdRef](#)
Identifier reference class specific to obo collection namespace.
- class [unisys::Tracking](#)
Data updating history class.
- class [unisys::LiteralBSON](#)
Specific BSON object to manage general [Literal](#) data object.

Namespaces

- namespace [unisys](#)
This namespace.

Functions

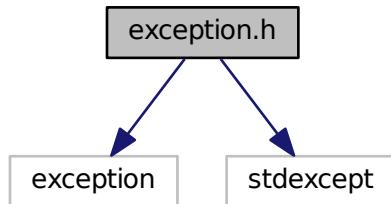
- template<class type >
mongo::BSONArray [unisys::toBSONArray](#) (type const &input)
- template<class type >
std::set< type > [unisys::BSONToset](#) (mongo::BSONObj const &bsonObj)
- template<class type >
std::vector< type > [unisys::BSONTovector](#) (mongo::BSONObj const &bsonObj)

10.4.1 Detailed Description

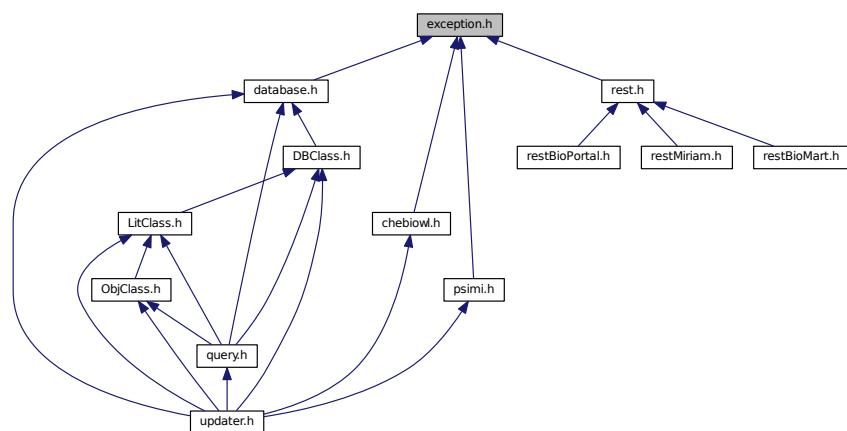
It contain Miriam, IdRef, Tracking and LiteralBSON class using for formating data to database structure.

10.5 exception.h File Reference

```
#include <exception>
#include <stdexcept>
Include dependency graph for exception.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class `unisys::UniSysError`
- class `unisys::ConnectionError`
- class `unisys::UpdateError`
- class `unisys::QueryError`
- class `unisys::DataError`
- class `unisys::ParsingError`
- class `unisys::RESTServiceError`

Namespaces

- namespace `unisys`

This namespace.

Enumerations

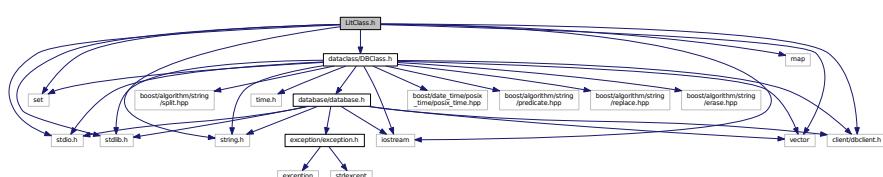
- enum `unisys::dbStatus` { `unisys::DB_OK`, `unisys::DB_CANNOT_CONNECT`, `unisys::DB_CANNOT_INSERT` }

10.6 LitClass.h File Reference

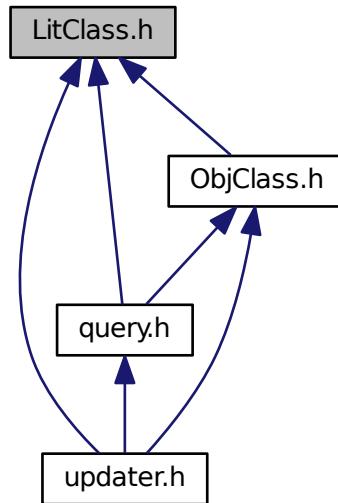
It contain all literal dirived class.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <iostream>
#include <vector>
#include <set>
#include <map>
#include "client/dbclient.h"
#include "dataclass/DBClass.h"
```

Include dependency graph for LitClass.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [unisys::Literal](#)
Literal class.
- class [unisys::Xref](#)
The C++ representative class for Cross reference data class.
- class [unisys::Score](#)
The C++ representative class for Score class in data structure.
- class [unisys::BioSource](#)
The C++ representative class for Biological Source data class. This class is designed to manage the biological data of source organism.
- class [unisys::Evidence](#)
The C++ representative class for evidence data class.
- class [unisys::Annotation](#)
The C++ representative class for annotation data class.
- class [unisys::OntoRelationship](#)
The C++ representative class for relationship data class.
- class [unisys::Relation](#)
The C++ representative class for relationship data class.
- class [unisys::CellularLocation](#)
The C++ representative class for cellular location data class.
- class [unisys::MathML](#)
The C++ representative class for MathML data class.
- class [unisys::KineticParameter](#)
The C++ representative class for kinetic parameter data class.
- class [unisys::SubRegion](#)
The C++ representative class for sub-region data class.
- class [unisys::Metadata](#)
The C++ representative class for metadata data class.

Namespaces

- namespace `unisys`

This namespace.

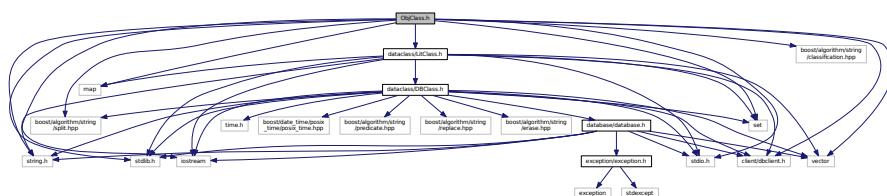
10.6.1 Detailed Description

It contain all literal dirived class.

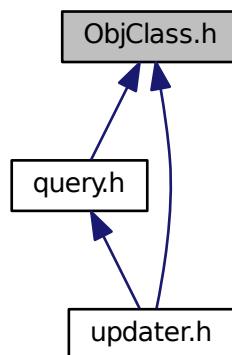
10.7 ObjClass.h File Reference

It contain all object derived class.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <iostream>
#include <vector>
#include <set>
#include <map>
#include <boost/algorithm/string/split.hpp>
#include <boost/algorithm/string/classification.hpp>
#include "client/dbclient.h"
#include "dataclass/LitClass.h"
Include dependency graph for ObjClass.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class [unisys::Object](#)
Root of all object classes This class does not have any interfaces.
- class [unisys::Ontology](#)
Ontology data class.
- class [unisys::BioObject](#)
This class is for miriam cross reference annotation.
- class [unisys::PhysicalEntity](#)
This class is for miriam cross reference annotation.
- class [unisys::SmallMolecule](#)
This class is for miriam cross reference annotation.
- class [unisys::DNA](#)
This class is for miriam cross reference annotation.
- class [unisys::DNARegion](#)
This class is for miriam cross reference annotation.
- class [unisys::RNA](#)
This class is for miriam cross reference annotation.
- class [unisys::Protein](#)
This class is for miriam cross reference annotation.
- class [unisys::Complex](#)
This class is for miriam cross reference annotation.
- class [unisys::Interaction](#)
This class is for miriam cross reference annotation.
- class [unisys::Control](#)
This class is for miriam cross reference annotation.
- class [unisys::MolecularInteraction](#)
This class is for miriam cross reference annotation.
- class [unisys::GeneticInteraction](#)
This class is for miriam cross reference annotation.
- class [unisys::Conversion](#)
This class is for miriam cross reference annotation.
- class [unisys::BiochemicalReaction](#)
This class is for miriam cross reference annotation.
- class [unisys::Transport](#)
This class is for miriam cross reference annotation.
- class [unisys::BiochemicalReactionWithTransport](#)
This class is for miriam cross reference annotation.

Namespaces

- namespace [unisys](#)
This namespace.

10.7.1 Detailed Description

It contain all object derived class.

10.8 objElement.h File Reference

Namespaces

- namespace [unisys](#)

This namespace.

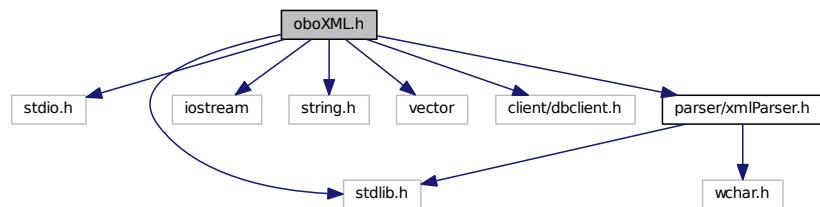
10.8.1 Detailed Description

10.9 oboXML.h File Reference

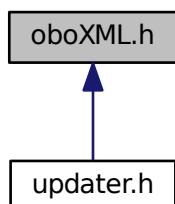
It contain OBOXML and Stanza class using for parsing OBO-XML format.

```
#include <stdio.h>
#include <stdlib.h>
#include <iostream>
#include <string.h>
#include <vector>
#include "client/dbclient.h"
#include "parser/xmlParser.h"
```

Include dependency graph for oboXML.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [unisys::Stanza](#)

This class is used to parse OWL format.

- class [unisys::OBOXML](#)

This class is used to parse OWL format.

Namespaces

- namespace [unisys](#)

This namespace.

Macros

- `#define _UNISYSDB_OBOXML_H_`

10.9.1 Detailed Description

It contain OBOXML and Stanza class using for parsing OBO-XML format.

10.9.2 Macro Definition Documentation

10.9.2.1 `#define _UNISYSDB_OBOXML_H_`

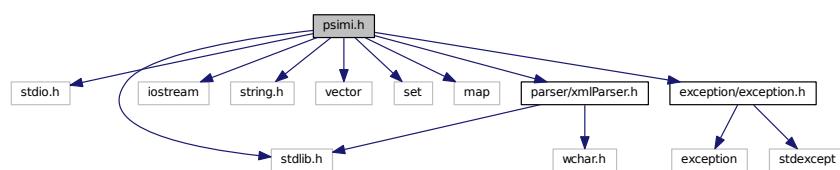
10.10 parser.h File Reference

10.11 psimi.h File Reference

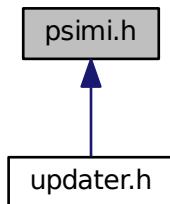
It contains PSIMI class for parsing PSI-MI XML2.5 file.

```
#include <stdio.h>
#include <stdlib.h>
#include <iostream>
#include <string.h>
#include <vector>
#include <set>
#include <map>
#include "parser/xmlParser.h"
#include "exception/exception.h"
```

Include dependency graph for psimi.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [unisys::PSIMI](#)

A PSIMI class is used for a data in PSI-MI XML2.5 file format.

Namespaces

- namespace [unisys](#)

This namespace.

10.11.1 Detailed Description

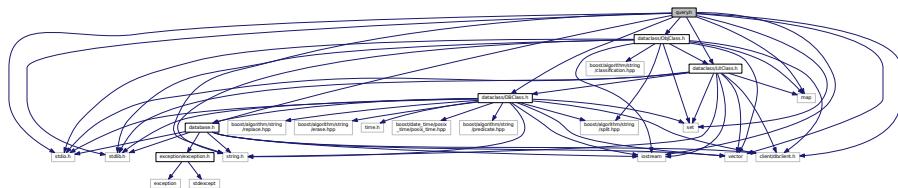
It contains PSIMI class for parsing PSI-MI XML2.5 file. Molecular interaction data in Proteomics Standards Initiative Molecular Interaction XML (PSI-MI XML) format will be parsed as a PSIMI object.

10.12 query.h File Reference

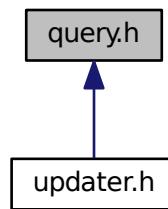
```

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <iostream>
#include <vector>
#include <set>
#include <map>
#include "client/dbclient.h"
#include "database.h"
#include "dataclass/LitClass.h"
#include "dataclass/ObjClass.h"
#include "dataclass/DBClass.h"
  
```

Include dependency graph for query.h:



This graph shows which files directly or indirectly include this file:



Classes

- class unisys::Query

Namespaces

- namespace `unisys`

This namespace.

Macros

- `#define _UNISYSDB_QUERY_H_`

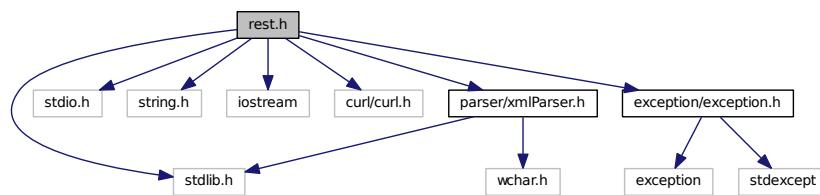
10.12.1 Macro Definition Documentation

10.12.1.1 #define _UNISYSDB_QUERY_H_

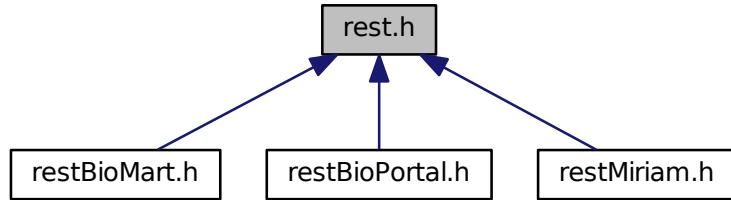
10.13 rest.h File Reference

It contains the library to handle the REST service request and response.

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <iostream>
#include <curl/curl.h>
#include "parser/xmlParser.h"
#include "exception/exception.h"
Include dependency graph for rest.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class [unisys::REST](#)
Used for managing the request and response of REST service.

Namespaces

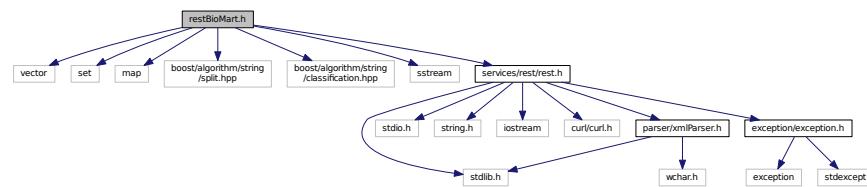
- namespace [unisys](#)
This namespace.

10.13.1 Detailed Description

It contains the library to handle the REST service request and response.

10.14 restBioMart.h File Reference

```
#include <vector>
#include <set>
#include <map>
#include <boost/algorithm/string/split.hpp>
#include <boost/algorithm/string/classification.hpp>
#include <sstream>
#include "services/rest/rest.h"
Include dependency graph for restBioMart.h:
```



Classes

- class [unisys::RestBioMartResult](#)
General class for taking care the response from BioPortal's REST service.
- class [unisys::RestBioMart](#)
General class for taking care the response from BioPortal's REST service.

Namespaces

- namespace [unisys](#)
This namespace.

Macros

- #define [BIOMART_VERSION](#) "0.8"

10.14.1 Macro Definition Documentation

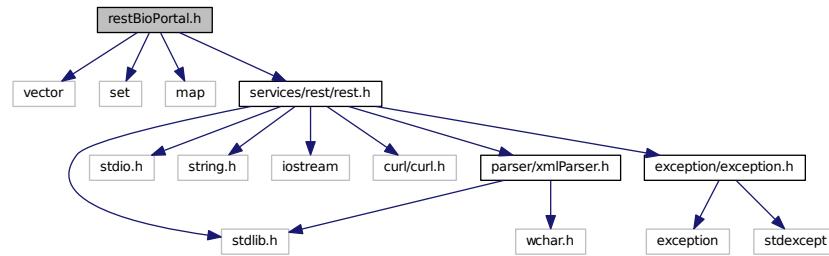
10.14.1.1 #define BIOMART_VERSION "0.8"

10.15 restBioPortal.h File Reference

BioPortal's REST service library.

```
#include <vector>
#include <set>
#include <map>
#include "services/rest/rest.h"
```

Include dependency graph for restBioPortal.h:



Classes

- class [unisys::RestBioPortalResult](#)
General class for taking care the response from BioPortal's REST service.
- class [unisys::RestBioPortalSearchResult](#)
General class for taking care the response from BioPortal's search command of REST service.
- class [unisys::ClassBean](#)
This class is for taking care of the specific structure of response data call classBean.
- struct [unisys::ClassBean::relationDataStructure](#)
- class [unisys::RestBioPortalTermResult](#)
This class is for parsing the response of term command.
- class [unisys::RestBioPortal](#)
The main class of BioPortal REST service library.

Namespaces

- namespace [unisys](#)
This namespace.

10.15.1 Detailed Description

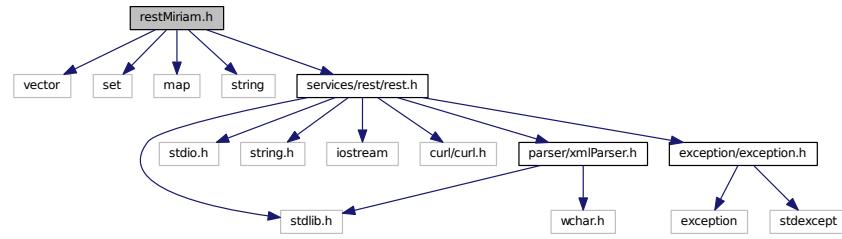
BioPortal's REST service library. This library contains search and term command. The serach command is used for retrieving the ontology that related with the query word from specific or non-specific ontology class. The term command is difference. This command is used for retrieving the information from BioPortal of exact ID from the specific ontology class

10.16 restMiriam.h File Reference

Miriam's REST service library.

```
#include <vector>
#include <set>
#include <map>
#include <string>
#include "services/rest/rest.h"
```

Include dependency graph for restMiriam.h:



Classes

- class [unisys::RestMiriam](#)

General class for taking care the response from BioPortal's REST service.

Namespaces

- namespace [unisys](#)

This namespace.

10.16.1 Detailed Description

Miriam's REST service library. This library contains interfaces of Miriam's REST service

10.17 unisysdb.h File Reference

Namespaces

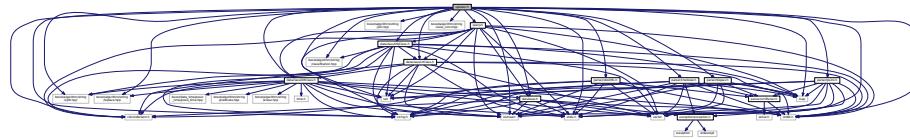
- namespace [unisys](#)

This namespace.

10.18 updater.h File Reference

It contain KGMLPlus and its subclass Graphic.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <iostream>
#include <vector>
#include <set>
#include <map>
#include <boost/algorithm/string/join.hpp>
#include <boost/algorithm/string/split.hpp>
#include <boost/algorithm/string/replace.hpp>
#include <boost/algorithm/string/case_conv.hpp>
#include <boost/algorithm/string/classification.hpp>
#include "client/dbclient.h"
#include "database.h"
#include "query.h"
#include "dataclass/LitClass.h"
#include "dataclass/ObjClass.h"
#include "dataclass/DBClass.h"
#include "parser/chebiowl.h"
#include "parser/oboXML.h"
#include "parser/biopax.h"
#include "parser/psimi.h"
Include dependency graph for updater.h:
```



Classes

- class [unisys::Updater](#)
- class [unisys::BatchInsert](#)

Namespaces

- namespace [unisys](#)

This namespace.

Macros

- [#define _UNISYSDB_UPDATER_H_](#)

Functions

- std::string [unisys::setMiriamURN](#) (std::string db, [XMLNode](#) node)

10.18.1 Detailed Description

It contain KGMLPlus and its subclass Graphic.

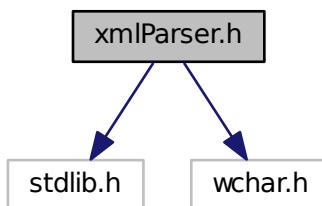
10.18.2 Macro Definition Documentation

10.18.2.1 #define _UNISYSDB_UPDATER_H_

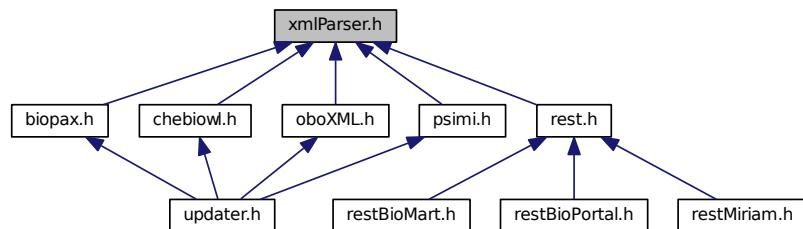
10.19 xmlParser.h File Reference

```
#include <stdlib.h>
#include <wchar.h>
```

Include dependency graph for xmlParser.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct [XMLResults](#)
Structure used to obtain error details if the parse fails.
- struct [XMLClear](#)
Structure for XML clear (unformatted) node (usually comments)
- struct [XMLAttribute](#)
Structure for XML attribute.
- struct [XMLNode](#)
Main Class representing a XML node.
- struct [XMLNode::XMLNodeDataTag](#)
- struct [XMLNodeContents](#)
This structure is given by the function `XMLNode::enumContents`.
- struct [ToXMLStringTool](#)

- **struct XMLParserBase64Tool**
Helper class to include binary data inside XML strings using "Base64 encoding".

Macros

- `#define XMLDLLENTRY`
- `#define XMLDLLENTRY`
- `#define _CXML(c) c`
- `#define XMLCSTR const char *`
- `#define XMLSTR char *`
- `#define XMLCHAR char`
- `#define FALSE 0`
- `#define TRUE 1`

Typedefs

- **typedef enum XMLError XMLError**
Enumeration for XML parse errors.
- **typedef enum XMLElementType XMLElementType**
Enumeration used to manage type of data. Use in conjunction with structure [XMLNodeContents](#).
- **typedef struct XMLResults XMLResults**
Structure used to obtain error details if the parse fails.
- **typedef struct XMLClear XMLClear**
Structure for XML clear (unformatted) node (usually comments)
- **typedef struct XMLAttribute XMLAttribute**
Structure for XML attribute.
- **typedef int XMLElementPosition**
XMLElementPosition are not interchangeable with simple indexes.
- **typedef struct XMLDLLENTRY XMLNode XMLNode**
Main Class representing a XML node.
- **typedef struct XMLNodeContents XMLNodeContents**
This structure is given by the function [XMLNode::enumContents](#).
- **typedef struct XMLDLLENTRY ToXMLStringTool ToXMLStringTool**
Helper class to create XML files using "printf", "fprintf", "cout",... functions.
- **typedef struct XMLDLLENTRY XMLParserBase64Tool XMLParserBase64Tool**
Helper class to include binary data inside XML strings using "Base64 encoding".

Enumerations

- **enum XMLError {**
`eXMLErrorNone = 0, eXMLErrorMissingEndTag, eXMLErrorNoXMLTagFound, eXMLErrorEmpty,`
`eXMLErrorMissingTagName, eXMLErrorMissingEndTagName, eXMLErrorUnmatchedEndTag, eXMLError-`
`UnmatchedEndClearTag,`
`eXMLErrorUnexpectedToken, eXMLErrorNoElements, eXMLErrorFileNotFoundException, eXMLErrorFirstTagNot-`
`Found,`
`eXMLErrorUnknownCharacterEntity, eXMLErrorCharacterCodeAbove255, eXMLErrorCharConversionError,`
`eXMLErrorCannotOpenWriteFile,`
`eXMLErrorCannotWriteFile, eXMLErrorBase64DataSizesNotMultipleOf4, eXMLErrorBase64DecodeIllegal-`
`Character, eXMLErrorBase64DecodeTruncatedData,`
`eXMLErrorBase64DecodeBufferTooSmall }`

Enumeration for XML parse errors.

- enum **XMLElementType** {

eNodeChild = 0, eNodeAttribute = 1, eNodeText = 2, eNodeClear = 3,

eNodeNULL = 4 }

Enumeration used to manage type of data. Use in conjunction with structure [XMLNodeContents](#).

Functions

- **XMDLLENTRY XMLSTR stringDup** (XMLCSTR source, int cbData=-1)

Duplicate (copy in a new allocated buffer) the source string.
- **XMDLLENTRY void freeXMLString** (XMLSTR t)

to free the string allocated inside the "stringDup" function or the "createXMLString" function.
- **XMDLLENTRY char xmltob** (XMLCSTR xmlString, char defaultValue=0)
- **XMDLLENTRY int xmltoi** (XMLCSTR xmlString, int defaultValue=0)
- **XMDLLENTRY long xmltol** (XMLCSTR xmlString, long defaultValue=0)
- **XMDLLENTRY double xmltof** (XMLCSTR xmlString, double defaultValue=.0)
- **XMDLLENTRY XMLCSTR xmltoa** (XMLCSTR xmlString, XMLCSTR defaultValue=_CXML(""))

XMDLLENTRY XMLCHAR xmltoc (XMLCSTR xmlString, const XMLCHAR defaultValue=_CXML('\0'))

10.19.1 Macro Definition Documentation

10.19.1.1 #define _CXML(c) c

10.19.1.2 #define FALSE 0

10.19.1.3 #define TRUE 1

10.19.1.4 #define XMLCHAR char

10.19.1.5 #define XMLCSTR const char *

10.19.1.6 #define XMDLLENTRY

10.19.1.7 #define XMLDLLEntry

10.19.1.8 #define XMLSTR char *

10.19.2 Typedef Documentation

10.19.2.1 **typedef struct XMLAttribute XMLAttribute**

Structure for XML attribute.

10.19.2.2 **typedef struct XMLClear XMLClear**

Structure for XML clear (unformatted) node (usually comments)

10.19.2.3 **typedef int XMLElementPosition**

XMLElementPosition are not interchangeable with simple indexes.

10.19.2.4 **typedef enum XMLElementType XMLElementType**

Enumeration used to manage type of data. Use in conjunction with structure [XMLNodeContents](#).

10.19.2.5 **typedef enum XMLError XMLError**

Enumeration for XML parse errors.

10.19.2.6 **typedef struct XMLDLLENTRY XMLNode XMLNode**

Main Class representing a XML node.

All operations are performed using this class.

Note

The constructors of the [XMLNode](#) class are protected, so use instead one of these four methods to get your first instance of [XMLNode](#):

- [XMLNode::parseString](#)
- [XMLNode::parseFile](#)
- [XMLNode::openFileHelper](#)
- [XMLNode::createXMLTopNode](#) (or [XMLNode::createXMLTopNode_WOSD](#))

10.19.2.7 **typedef struct XMLNodeContents XMLNodeContents**

This structure is given by the function [XMLNode::enumContents](#).

10.19.2.8 **typedef struct XMLResults XMLResults**

Structure used to obtain error details if the parse fails.

10.19.3 **Enumeration Type Documentation****10.19.3.1 **enum XMLElementType****

Enumeration used to manage type of data. Use in conjunction with structure [XMLNodeContents](#).

Enumerator:

- eNodeChild**
eNodeAttribute
eNodeText
eNodeClear
eNodeNULL

10.19.3.2 **enum XMLError**

Enumeration for XML parse errors.

Enumerator:

- eXMLErrorNone**

eXMLErrorMissingEndTag
eXMLErrorNoXML TagFound
eXMLErrorEmpty
eXMLErrorMissingTagName
eXMLErrorMissingEndTagName
eXMLErrorUnmatchedEndTag
eXMLErrorUnmatchedEndClearTag
eXMLErrorUnexpectedToken
eXMLErrorNoElements
eXMLErrorNotFound
eXMLErrorFirstTagNotFound
eXMLErrorUnknownCharacterEntity
eXMLErrorCharacterCodeAbove255
eXMLErrorCharConversionError
eXMLErrorCannotOpenWriteFile
eXMLErrorCannotWriteFile
eXMLErrorBase64DataSizeIsNotMultipleOf4
eXMLErrorBase64DecodeIllegalCharacter
eXMLErrorBase64DecodeTruncatedData
eXMLErrorBase64DecodeBufferTooSmall

Index

~REST
 unisys::REST, 196

~ToXMLStringTool
 ToXMLStringTool, 239

~UniSysError
 unisys::UniSysError, 247

~XMLNode
 XMLNode, 263

~XMLParserBase64Tool
 XMLParserBase64Tool, 270

??_WOSD functions., 32
 addAttribute_WOSD, 33
 addChild_WOSD, 33
 addClear_WOSD, 33
 addText_WOSD, 33
 createXMLTopNode_WOSD, 33
 updateAttribute_WOSD, 33, 34
 updateClear_WOSD, 34
 updateName_WOSD, 34
 updateText_WOSD, 34

_CXML
 xmlParser.h, 299

accessDate
 unisys::RestBioPortalResult, 210

accessedResource
 unisys::RestBioPortalResult, 210

addAnnotation
 unisys::BioObject, 62

addAttribute
 Creating from scratch a XMLNode structure, 26

addAttribute_WOSD
 ??_WOSD functions., 33

addAttribute_priv
 XMLNode, 263

addCellularLocation
 unisys::PhysicalEntity, 174

addChild
 Creating from scratch a XMLNode structure, 26

addChild_WOSD
 ??_WOSD functions., 33

addChild_priv
 XMLNode, 263

addClear
 Creating from scratch a XMLNode structure, 26

addClear_WOSD
 ??_WOSD functions., 33

addClear_priv
 XMLNode, 263

addComplexMember
 unisys::Complex, 88

addControlType
 unisys::Control, 93

addDNARegion
 unisys::DNA, 110

addDataXref
 unisys::BioObject, 62

addEvidence
 unisys::BioObject, 62

addExport
 unisys::BiochemicalReactionWithTransport, 58
 unisys::Transport, 245

addFieldSet
 unisys::DataObj, 105

addImport
 unisys::BiochemicalReactionWithTransport, 58
 unisys::Transport, 245

addKineticLaw
 unisys::Conversion, 97

addLeft
 unisys::BiochemicalReaction, 54
 unisys::MolecularInteraction, 150

addLeftIn
 unisys::BiochemicalReactionWithTransport, 58

addLeftOut
 unisys::BiochemicalReactionWithTransport, 58

addModificationType
 unisys::Control, 93

addOntoRelationship
 unisys::Object, 153

addPhenotype
 unisys::Control, 93

addPosition
 unisys::RNA, 225
 unisys::SubRegion, 237

addRNASource
 unisys::Protein, 177

addRelation
 unisys::Object, 153

addRight
 unisys::BiochemicalReaction, 54
 unisys::MolecularInteraction, 150

addRightIn
 unisys::BiochemicalReactionWithTransport, 59

addRightOut
 unisys::BiochemicalReactionWithTransport, 59

addSpecificKineticParameter
 unisys::Complex, 88
 unisys::Protein, 177

unisys::SmallMolecule, 231
addSubRegion
 unisys::PhysicalEntity, 174
addSynonyms
 unisys::BioObject, 62
addText
 Creating from scratch a XMLNode structure, 27
addText_WOSD
 ???_WOSD functions., 33
addText_priv
 XMLNode, 263
addToOrder
 XMLNode, 263
addTranscriptionProduct
 unisys::DNARegion, 113
addTranslationProduct
 unisys::RNA, 225
addWithCheck
 unisys::DataObj, 105
addWithOutCheck
 unisys::DataObj, 105
alloc
 XMLParserBase64Tool, 270
Annotation
 unisys::Annotation, 48
AnnotationProperties
 unisys::ChEBIOWLClass, 80
append
 unisys::DataObj, 105
appendArray
 unisys::DataObj, 105
appendAs
 unisys::DataObj, 105
appendAsNumber
 unisys::DataObj, 105
appendCode
 unisys::DataObj, 105
appendDBRef
 unisys::DataObj, 105
appendElements
 unisys::DataObj, 105
appendElementsUnique
 unisys::DataObj, 105
appendNull
 unisys::DataObj, 105
appendRegex
 unisys::DataObj, 105
appendTimeStamp
 unisys::DataObj, 105
appendTimeT
 unisys::DataObj, 105
ato? like functions, 38
 xmltoa, 38
 xmltob, 38
 xmltoc, 38
 xmltof, 38
 xmltoi, 38
 xmltol, 38
attrib
 XMLNodeContents, 266
attributeList
 unisys::PSIMI, 181
availabilityList
 unisys::PSIMI, 181
BIOMART_VERSION
 restBioMart.h, 293
BIOPAX2
 unisys::BIOPAX2, 65
BIOPAX_obj
 unisys::BIOPAX_obj, 67
BSONToSet
 unisys, 44
BSONToVector
 unisys, 44
baseURL
 unisys::RestBioPortal, 206
 unisys::RestMiriam, 219
BatchInsert
 unisys::BatchInsert, 51
begin
 unisys::BIOPAX2, 65
bioMartVersion
 unisys::RestBioMart, 200
 unisys::RestBioMartResult, 202
BioObject
 unisys::BioObject, 62
BioSource
 unisys::BioSource, 71
BiochemicalReaction
 unisys::BiochemicalReaction, 54
BiochemicalReactionWithTransport
 unisys::BiochemicalReactionWithTransport, 58
biopax
 unisys::BIOPAX2, 65
biopax.h, 277
biopaxObjMap
 unisys::BIOPAX2, 65
buf
 ToXMLStringTool, 239
 XMLParserBase64Tool, 271
buffer
 unisys::REST, 197
buflen
 ToXMLStringTool, 239
 XMLParserBase64Tool, 271
callInteractionKey
 unisys::Interaction, 127
callback
 unisys::REST, 196
CellularLocation
 unisys::CellularLocation, 75
ChEBIOWL
 unisys::ChEBIOWL, 77
ChEBIOWLClass
 unisys::ChEBIOWLClass, 80

char_encoding_Big5
 XMLNode, 263
 char_encoding_GB2312
 XMLNode, 263
 char_encoding_GBK
 XMLNode, 263
 char_encoding_ShiftJIS
 XMLNode, 263
 char_encoding_UTF8
 XMLNode, 263
 char_encoding_error
 XMLNode, 263
 char_encoding_legacy
 XMLNode, 263
 chebIld
 unisys::ChEBIOWLClass, 80
 chebiowl.h, 278
 checkIdPair
 unisys::Updater, 251
 child
 XMLNodeContents, 266
 ClassBean
 unisys::ClassBean, 84
 classBean
 unisys::RestBioPortalTermResult, 216
 classList
 unisys::ChEBIOWL, 77
 cleanString
 unisys::REST, 196
 clear
 XMLNodeContents, 266
 Complex
 unisys::Complex, 88
 connect
 unisys::Database, 99
 Control
 unisys::Control, 93
 Conversion
 unisys::Conversion, 97
 count
 unisys::ClassBean::relationDataStructure, 192
 Create or Update the XMLNode structure, 25
 createIdPair
 unisys::BioObject, 62
 createRelationInsert
 unisys::Object, 153
 createRelationRemove
 unisys::Object, 153
 createTrack
 unisys::Object, 153
 createXMLString
 Parsing XML files/strings to an XMLNode structure
 and Rendering XMLNode's to files/string., 17
 CreateXMLStringR
 XMLNode, 263
 createXMLTopNode
 Creating from scratch a XMLNode structure, 27
 createXMLTopNode_WOSD
 ???_WOSD functions., 33
 Creating from scratch a XMLNode structure, 26
 addAttribute, 26
 addChild, 26
 addClear, 26
 addText, 27
 createXMLTopNode, 27
 curlHandle
 unisys::REST, 197
 curlResult
 unisys::REST, 197
 d
 XMLNode, 264
 DB_CANNOT_CONNECT
 unisys, 44
 DB_CANNOT_INSERT
 unisys, 44
 DB_OK
 unisys, 44
 DBClass.h, 281
 DNA
 unisys::DNA, 110
 DNARegion
 unisys::DNARegion, 113
 data
 unisys::DataObj, 107
 DataError
 unisys::DataError, 101
 DataObj
 unisys::DataObj, 105
 Database
 unisys::Database, 99
 database.h, 279
 databaseHandle
 unisys::Query, 184
 unisys::Updater, 254
 dbStatus
 unisys, 44
 dbname
 unisys::Database, 99
 decode
 XMLParserBase64Tool, 270
 decodeSize
 XMLParserBase64Tool, 270
 deepCopy
 Navigate the XMLNode structure, 22
 definitions
 unisys::ClassBean, 84
 deleteAttribute
 Deleting Nodes or Attributes, 30
 deleteClear
 Deleting Nodes or Attributes, 31
 deleteNodeContent
 Deleting Nodes or Attributes, 31
 deleteText
 Deleting Nodes or Attributes, 31
 Deleting Nodes or Attributes, 30
 deleteAttribute, 30

deleteClear, 31
deleteNodeContent, 31
deleteText, 31
detachFromParent
 XMLNode, 264

eNodeAttribute
 xmlParser.h, 300
eNodeChild
 xmlParser.h, 300
eNodeClear
 xmlParser.h, 300
eNodeNULL
 xmlParser.h, 300
eNodeText
 xmlParser.h, 300
eXMLErrorBase64DataSizelsNotMultipleOf4
 xmlParser.h, 301
eXMLErrorBase64DecodeBufferTooSmall
 xmlParser.h, 301
eXMLErrorBase64DecodelllegalCharacter
 xmlParser.h, 301
eXMLErrorBase64DecodeTruncatedData
 xmlParser.h, 301
eXMLErrorCannotOpenWriteFile
 xmlParser.h, 301
eXMLErrorCannotWriteFile
 xmlParser.h, 301
eXMLErrorCharConversionError
 xmlParser.h, 301
eXMLErrorCharacterCodeAbove255
 xmlParser.h, 301
eXMLErrorEmpty
 xmlParser.h, 301
eXMLErrorNotFound
 xmlParser.h, 301
eXMLErrorFirstTagNotFound
 xmlParser.h, 301
eXMLErrorMissingEndTag
 xmlParser.h, 300
eXMLErrorMissingEndTagName
 xmlParser.h, 301
eXMLErrorMissingTagName
 xmlParser.h, 301
eXMLErrorNoElements
 xmlParser.h, 301
eXMLErrorNoXMLTagFound
 xmlParser.h, 301
eXMLErrorNone
 xmlParser.h, 300
eXMLErrorUnexpectedToken
 xmlParser.h, 301
eXMLErrorUnknownCharacterEntity
 xmlParser.h, 301
eXMLErrorUnmatchedEndClearTag
 xmlParser.h, 301
eXMLErrorUnmatchedEndTag
 xmlParser.h, 301
emptyNode
 Navigate the XMLNode structure, 22
emptyTheNode
 XMLNode, 264
emptyXMLAttribute
 XMLNode, 264
emptyXMLClear
 XMLNode, 264
emptyXMLNode
 XMLNode, 264
encode
 XMLParserBase64Tool, 270
encodeLength
 XMLParserBase64Tool, 271
end
 unisys::BIOPAX2, 65
enumContents
 Navigate the XMLNode structure, 22
error
 XMLResults, 272
errorCode
 unisys::RestBioPortalResult, 210
errorLongMessage
 unisys::RestBioPortalResult, 210
errorMsg
 unisys::UniSysError, 247
etype
 XMLNodeContents, 266
Evidence
 unisys::Evidence, 116
exactMemory
 XMLNode, 264
exception.h, 283
experimentList
 unisys::PSIMI, 181
explain
 unisys::Query, 184

FALSE
 xmlParser.h, 299
fetchDBRef
 unisys::Database, 99
fieldSet
 unisys::DataObj, 107
findPosition
 XMLNode, 264
freeBuffer
 ToXMLStringTool, 239
 XMLParserBase64Tool, 271
freeXMLString
 String Allocation/Free functions, 37
fullId
 unisys::ClassBean, 84

genOID
 unisys::DataObj, 105
GeneticInteraction
 unisys::GeneticInteraction, 120
getAccessDate
 unisys::RestBioPortalResult, 209

getAccessedResource
 unisys::RestBioPortalResult, 209

getAnnotationProperties
 unisys::ChEBIOWLClass, 80

getAttribute
 Navigate the XMLNode structure, 22
 unisys::BIOPAX_obj, 67

getAttributeName
 Navigate the XMLNode structure, 23
 unisys::BIOPAX_obj, 67

getAttributeValue
 Navigate the XMLNode structure, 23
 unisys::BIOPAX_obj, 67

getChildNode
 Navigate the XMLNode structure, 23
 unisys::BIOPAX_obj, 67

getChildNodeByPath
 Navigate the XMLNode structure, 23

getChildNodeByPathNonConst
 Navigate the XMLNode structure, 23

getChildNodeWithAttribute
 Navigate the XMLNode structure, 23

getClassBean
 unisys::RestBioPortalTermResult, 216

getClassList
 unisys::ChEBIOWL, 77

getClear
 Navigate the XMLNode structure, 23

getDBName
 unisys::Database, 99

getDefinition
 unisys::Stanza, 234

getDefinitions
 unisys::ClassBean, 84

getError
 Parsing XML files/strings to an XMLNode structure
 and Rendering XMLNode's to files/string., 18
 unisys::RestBioPortalResult, 209

getField
 unisys::DataObj, 106

getFileAnnotation
 unisys::ChEBIOWL, 77

getFormula
 unisys::ChEBIOWLClass, 80

getFullId
 unisys::ClassBean, 84

getId
 unisys::BIOPAX_obj, 67
 unisys::ChEBIOWLClass, 80
 unisys::ClassBean, 84
 unisys::IdRef, 123
 unisys::Stanza, 234

getIdList
 unisys::ClassBean::relationDataStructure, 192

getInChi
 unisys::ChEBIOWLClass, 80

getInChiKey
 unisys::ChEBIOWLClass, 80

getLabel
 unisys::ChEBIOWLClass, 80
 unisys::ClassBean, 84

getNS
 unisys::IdRef, 123
 unisys::Stanza, 234

getName
 Navigate the XMLNode structure, 23
 unisys::Stanza, 234

getNodeAttributeList
 unisys::PSIMI, 180

getNodeAvailabilityList
 unisys::PSIMI, 180

getNodeExperimentList
 unisys::PSIMI, 180

getNodeInteractionList
 unisys::PSIMI, 181

getNodeInteractorList
 unisys::PSIMI, 181

getNodeSource
 unisys::PSIMI, 181

getOWLNode
 unisys::BIOPAX2, 65

getOntologyHitList
 unisys::RestBioPortalSearchResult, 213

getParentNode
 Navigate the XMLNode structure, 23

getParticipant
 unisys::Interaction, 127

getPropertyList
 unisys::ChEBIOWLClass, 80

getPropertyValue
 unisys::BIOPAX_obj, 67

getRelation
 unisys::Object, 153

getRelations
 unisys::ClassBean, 84

getRelationship
 unisys::Stanza, 234

getResult
 unisys::REST, 196

getResultInXML
 unisys::REST, 196

getSMILES
 unisys::ChEBIOWLClass, 80

getSearchResultList
 unisys::RestBioPortalSearchResult, 213

getServiceURL
 unisys::RestBioMart, 200

getSpAnnotationProperty
 unisys::ChEBIOWLClass, 80

getSpSubClassOf
 unisys::ChEBIOWLClass, 80

getSpecificOntology
 unisys::RestBioPortal, 205

getSubClassOf
 unisys::ChEBIOWLClass, 80

getSynonyms

unisys::ClassBean, 84
getText
 Navigate the XMLNode structure, 24
getType
 unisys::BIOPAX_obj, 67
 unisys::ClassBean, 84
getVersion
 unisys::ChEBIOWL, 77
 XMLNode, 264
globalClean
 unisys::REST, 196
globalInit
 unisys::REST, 196
guessCharEncoding
 Parsing XML files/strings to an XMLNode structure
 and Rendering XMLNode's to files/string., 18
hasField
 unisys::DataObj, 106
header
 unisys::OBOXML, 156
 unisys::RestBioMartResult, 202
Helper class to create XML files using "printf", "fprintf",
 "cout",... functions., 39
 ToXMLStringTool, 39
Helper class to include binary data inside XML strings
 using "Base64 encoding", 40
 XMLParserBase64Tool, 40
id
 unisys::ClassBean, 84
 unisys::Miriam, 147
idConverter
 unisys::Updater, 252
IdRef
 unisys::IdRef, 123
inchi
 unisys::ChEBIOWLClass, 80
inchiKey
 unisys::ChEBIOWLClass, 80
increaseVer
 unisys::Miriam, 146
indexClear
 XMLNode, 264
indexText
 XMLNode, 264
initField
 unisys::Annotation, 48
 unisys::BiochemicalReaction, 54
 unisys::BiochemicalReactionWithTransport, 59
 unisys::BioObject, 62
 unisys::BioSource, 71
 unisys::CellularLocation, 75
 unisys::Complex, 88
 unisys::Control, 93
 unisys::Conversion, 97
 unisys::DataObj, 106
 unisys::DNA, 110
 unisys::DNARegion, 113
unisys::Evidence, 116
unisys::GeneticInteraction, 120
unisys::IdRef, 123
unisys::Interaction, 127
unisys::KineticParameter, 133
unisys::Literal, 136
unisys::MathML, 140
unisys::Metadata, 143
unisys::MolecularInteraction, 150
unisys::Object, 153
unisys::Ontology, 162
unisys::OntoRelationship, 165
unisys::PhysicalEntity, 174
unisys::Protein, 177
unisys::Relation, 190
unisys::RNA, 225
unisys::Score, 228
unisys::SmallMolecule, 231
unisys::SubRegion, 237
unisys::Tracking, 242
unisys::Transport, 245
unisys::Xref, 275
initMember
 unisys::IdRef, 123
 unisys::IntIdRef, 130
 unisys::OntoldRef, 159
 unisys::PEIdRef, 171
insert
 unisys::BatchInsert, 51
 unisys::Updater, 252
insertPublicationSupport
 unisys::Evidence, 116
IntIdRef
 unisys::IntIdRef, 130
Interaction
 unisys::Interaction, 127
interactionList
 unisys::PSIMI, 181
interactorList
 unisys::PSIMI, 181
isAttributeSet
 Navigate the XMLNode structure, 24
isDeclaration
 Navigate the XMLNode structure, 24
 XMLNode::XMLNodeDataTag, 268
isEmpty
 Navigate the XMLNode structure, 24
isError
 unisys::RestBioPortalResult, 209
isObsolete
 unisys::Stanza, 234
isOwned
 unisys::DataObj, 106
isResponseError
 unisys::REST, 196
isValid
 unisys::DataObj, 106
 unisys::IdRef, 124

unisys::Miriam, 146
 unisys::Relation, 190
KineticParameter
 unisys::KineticParameter, 133
label
 unisys::ChEBIOWLClass, 81
 unisys::ClassBean, 85
lengthXMLString
 ToXMLStringTool, 239
list
 unisys::ClassBean::relationDataStructure, 192
listAllLastestOntologies
 unisys::RestBioPortal, 205
LitClass.h, 284
Literal
 unisys::Literal, 136
LiteralBSON
 unisys::LiteralBSON, 137
lpszCloseTag
 XMLClear, 256
lpszName
 XMLAttribute, 255
 XMLNode::XMLNodeDataTag, 268
lpszOpenTag
 XMLClear, 256
lpszValue
 XMLAttribute, 255
 XMLClear, 256
mapOfStringString
 unisys::PSIMI, 180
mapOfXMLNode
 unisys::PSIMI, 180
MathML
 unisys::MathML, 140
maybeAddTxT
 XMLNode, 264
md5
 unisys::DataObj, 106
Metadata
 unisys::Metadata, 143
Miriam
 unisys::Miriam, 145
MolecularInteraction
 unisys::MolecularInteraction, 150
nAttribute
 Navigate the XMLNode structure, 24
 unisys::BIOPAX_obj, 68
 XMLNode::XMLNodeDataTag, 268
nChild
 XMLNode::XMLNodeDataTag, 268
nChildNode
 Navigate the XMLNode structure, 24
 unisys::BIOPAX_obj, 68
nClear
 Navigate the XMLNode structure, 24
XMLNode::XMLNodeDataTag, 268
nColumn
 XMLResults, 272
nElement
 Navigate the XMLNode structure, 24
nLine
 XMLResults, 272
nSearchResults
 unisys::RestBioPortalSearchResult, 213
nText
 Navigate the XMLNode structure, 24
 XMLNode::XMLNodeDataTag, 268
 Navigate the XMLNode structure, 21
 deepCopy, 22
 emptyNode, 22
 enumContents, 22
 getAttribute, 22
 getAttributeName, 23
 getAttributeValue, 23
 getChildNode, 23
 getChildNodeByPath, 23
 getChildNodeByPathNonConst, 23
 getChildNodeWithAttribute, 23
 getClear, 23
 getName, 23
 getParentNode, 23
 getText, 24
 isAttributeSet, 24
 isDeclaration, 24
 isEmpty, 24
 nAttribute, 24
 nChildNode, 24
 nClear, 24
 nElement, 24
 nText, 24
ns
 unisys::Miriam, 147
OBOXML
 unisys::OBOXML, 156
ObjClass.h, 286
objElement.h, 288
Object
 unisys::Object, 153
object
 unisys::BIOPAX_obj, 68
objects
 unisys::BIOPAX2, 65
oboXML.h, 288
OntoldRef
 unisys::OntoldRef, 159
OntoRelationship
 unisys::OntoRelationship, 165
Ontology
 unisys::Ontology, 162
ontologyHitList
 unisys::RestBioPortalSearchResult, 213
openFileHelper

Parsing XML files/strings to an XMLNode structure
and Rendering XMLNode's to files/string., 18
operator<
 unisys::DataObj, 106
 unisys::KineticParameter, 133
 unisys::Miriam, 146
 unisys::Xref, 275
operator<=
 unisys::DataObj, 106
operator>
 unisys::DataObj, 106
operator>=
 unisys::DataObj, 106
operator++
 unisys::Miriam, 146
operator=
 XMLNode, 264
operator==
 unisys::DataObj, 106
 unisys::Miriam, 146
owlOntology
 unisys::ChEBIOWL, 77

pAttribute
 XMLNode::XMLNodeDataTag, 268
pChild
 XMLNode::XMLNodeDataTag, 268
pClear
 XMLNode::XMLNodeDataTag, 268
PEIdRef
 unisys::PEIdRef, 171
pOrder
 XMLNode::XMLNodeDataTag, 268
pParent
 XMLNode::XMLNodeDataTag, 268
PSIMI
 unisys::PSIMI, 180
pText
 XMLNode::XMLNodeDataTag, 268
parse
 unisys::ChEBIOWL, 77
 unisys::ChEBIOWLClass, 80
 unisys::PSIMI, 181
parseAnnotation
 unisys::ChEBIOWLClass, 80
parseClearTag
 XMLNode, 264
parseError
 unisys::RestBioPortalResult, 209
parseFile
 Parsing XML files/strings to an XMLNode structure
 and Rendering XMLNode's to files/string., 18
parseString
 Parsing XML files/strings to an XMLNode structure
 and Rendering XMLNode's to files/string., 19
parseSubClassOf
 unisys::ChEBIOWLClass, 80
ParseXMLElement
 XMLNode, 264
parser.h, 289
Parsing XML files/strings to an XMLNode structure and
Rendering XMLNode's to files/string., 17
 createXMLString, 17
 getError, 18
 guessCharEncoding, 18
 openFileHelper, 18
 parseFile, 18
 parseString, 19
 setGlobalOptions, 19
 writeToFile, 20
ParsingError
 unisys::ParsingError, 167
perform
 unisys::Query, 184
 unisys::REST, 196
performHTTPGET
 unisys::REST, 196
performOne
 unisys::Query, 184
performPOST
 unisys::REST, 196
PhysicalEntity
 unisys::PhysicalEntity, 174
Position helper functions (use in conjunction with the up-
date&add functions, 35
 positionOfChildNode, 35
 positionOfClear, 35
 positionOfText, 35
positionOfChildNode
 Position helper functions (use in conjunction with
 the update&add functions, 35)
positionOfClear
 Position helper functions (use in conjunction with
 the update&add functions, 35)
positionOfText
 Position helper functions (use in conjunction with
 the update&add functions, 35)
print
 unisys::REST, 196
Protein
 unisys::Protein, 177
psimi.h, 289
Query
 unisys::Query, 184
query
 unisys::Query, 184
 unisys::RestBioMart, 200
query.h, 290
queryById
 unisys::Query, 184
QueryError
 unisys::QueryError, 186
REST
 unisys::REST, 196
RESTServiceError
 unisys::RESTServiceError, 221

RNA
 unisys::RNA, 225

ref_count
 XMLNode::XMLNodeDataTag, 268

Relation
 unisys::Relation, 190

relationMap
 unisys::Object, 154

relationStruct
 unisys::ClassBean, 83

relations
 unisys::ClassBean, 85

remove
 unisys::Updater, 252, 253

removeField
 unisys::DataObj, 106

removeOrderElement
 XMLNode, 264

requireField
 unisys::DataObj, 107

resolve
 unisys::RestMiriam, 219

responseXML
 unisys::RestBioMartResult, 202

rest.h, 291

RestBioMart
 unisys::RestBioMart, 200
 unisys::RestBioMartResult, 202

restBioMart.h, 293
 BIOMART_VERSION, 293

RestBioMartResult
 unisys::RestBioMartResult, 202

RestBioPortal
 unisys::RestBioPortal, 205

restBioPortal.h, 293

RestBioPortalResult
 unisys::RestBioPortalResult, 209

RestBioPortalSearchResult
 unisys::RestBioPortalSearchResult, 213

RestBioPortalTermResult
 unisys::RestBioPortalTermResult, 216

RestMiriam
 unisys::RestMiriam, 219

restMiriam.h, 294

result
 unisys::REST, 197

Score
 unisys::Score, 228

search
 unisys::RestBioPortal, 205

searchResultList
 unisys::RestBioPortalSearchResult, 213

serviceURL
 unisys::RestBioMart, 200

set
 unisys::DataObj, 106
 unisys::Miriam, 146

setAnnotation

unisys::SubRegion, 237

setBiologicalSource
 unisys::Evidence, 116

setCellState
 unisys::BioSource, 71, 72

setCellType
 unisys::BioSource, 72

setCoefficient
 unisys::Relation, 190

setComment
 unisys::BioObject, 62
 unisys::Literal, 136

setConfidence
 unisys::Evidence, 116

setControlled
 unisys::Control, 93

setController
 unisys::Control, 94

setConversionDirection
 unisys::BiochemicalReaction, 55

setDBName
 unisys::Database, 99

setDNARegionSource
 unisys::RNA, 225

setDNADataSource
 unisys::DNARegion, 113

setDataPrimarySource
 unisys::BioObject, 63

setDbHandle
 unisys::Query, 184
 unisys::Updater, 253

setDefinition
 unisys::Annotation, 48
 unisys::Ontology, 162

setDetail
 unisys::Metadata, 143
 unisys::Xref, 276

setEvidence
 unisys::Annotation, 48
 unisys::CellularLocation, 75
 unisys::KineticParameter, 133
 unisys::MathML, 140
 unisys::OntoRelationship, 165
 unisys::Relation, 190

setEvidenceSource
 unisys::Evidence, 117

setExperimentMethod
 unisys::Evidence, 117

setFormula
 unisys::SmallMolecule, 231

setFunctional
 unisys::Conversion, 97

setGlobalOptions
 Parsing XML files/strings to an XMLNode structure
 and Rendering XMLNode's to files/string., 19

setId
 unisys::IntIdRef, 130
 unisys::Object, 154

unisys::OntoldRef, 159
unisys::PEIdRef, 171
unisys::Xref, 276
setInChi
 unisys::SmallMolecule, 231
setInChiKey
 unisys::SmallMolecule, 231
setInteractionKey
 unisys::BiochemicalReactionWithTransport, 59
 unisys::Interaction, 127
setInteractionType
 unisys::GeneticInteraction, 120
setKineticLaw
 unisys::MolecularInteraction, 150
setLength
 unisys::DNA, 110
setLocation
 unisys::CellularLocation, 75
setMainName
 unisys::BioObject, 63
setMathMLDetail
 unisys::MathML, 140
setMiriamURN
 unisys, 44
setNS
 unisys::Ontology, 162
setOntoRelationship
 unisys::OntoRelationship, 165
setParticipant
 unisys::Interaction, 127
setPhenotype
 unisys::GeneticInteraction, 120
setPosition
 unisys::DNARegion, 113
setQuery
 unisys::Query, 184
setRelationType
 unisys::OntoRelationship, 165
setRelationWith
 unisys::OntoRelationship, 165
 unisys::Relation, 190
setSMILES
 unisys::SmallMolecule, 232
setSciName
 unisys::BioSource, 72
setScoreType
 unisys::Score, 228
setServiceURL
 unisys::RestBioMart, 200
setSource
 unisys::DNA, 110
setSpontaneous
 unisys::Conversion, 97
setStain
 unisys::BioSource, 72
setStrand
 unisys::SubRegion, 237
setTaxonomicId
 unisys::DataObj, 106
 unisys::BioSource, 72
setTerm
 unisys::KineticParameter, 134
 unisys::Ontology, 162
setTissueType
 unisys::BioSource, 72
setType
 unisys::BioObject, 63
 unisys::Relation, 190
 unisys::Xref, 276
setUnit
 unisys::KineticParameter, 134
 unisys::Score, 228
setUri
 unisys::Miriam, 146
setValue
 unisys::KineticParameter, 134
 unisys::Score, 228
SmallMolecule
 unisys::SmallMolecule, 231
smiles
 unisys::ChEBIOWLClass, 81
sort
 unisys::DataObj, 106
 unisys::Query, 184
source
 unisys::OBOXML, 156
 unisys::PSIMI, 181
Stanza
 unisys::Stanza, 234
stanzas
 unisys::OBOXML, 156
stanzasBegin
 unisys::OBOXML, 156
stanzasEnd
 unisys::OBOXML, 156
String Allocation/Free functions, 37
 freeXMLString, 37
 stringDup, 37
stringDup
 String Allocation/Free functions, 37
subClassOf
 unisys::ChEBIOWLClass, 81
SubRegion
 unisys::SubRegion, 237
synonyms
 unisys::ClassBean, 85
TRUE
 xmlParser.h, 299
term
 unisys::RestBioPortal, 206
text
 XMLNodeContents, 266
The XML parser, 15
toBSONArray
 unisys, 44
toBSONObj
 unisys::DataObj, 106

unisys::Interaction, 127
 toDBId
 unisys::Miriam, 146
 toDBIdWithVer
 unisys::Miriam, 146
 told
 unisys::Miriam, 146
 toRXNString
 unisys::BiochemicalReaction, 55
 unisys::BiochemicalReactionWithTransport, 59
 unisys::Relation, 190
 unisys::Transport, 245
 toString
 unisys::DataObj, 106
 unisys::Query, 184
 unisys::Tracking, 242
 toURI
 unisys::Miriam, 146
 toURIWithVer
 unisys::Miriam, 146
 toXML
 ToXMLStringTool, 239
 unisys::Stanza, 234
 ToXMLStringTool, 238
 ~ToXMLStringTool, 239
 buf, 239
 buflen, 239
 freeBuffer, 239
 Helper class to create XML files using "printf",
 "fprintf", "cout",... functions., 39
 lengthXMLString, 239
 toXML, 239
 ToXMLStringTool, 239
 toXMLUnSafe, 239
 ToXMLStringTool, 239
 toXMLUnSafe
 ToXMLStringTool, 239
 Tracking
 unisys::Tracking, 242
 Transport
 unisys::Transport, 245
 type
 unisys::ClassBean, 85
 unisys::ClassBean::relationDataStructure, 192
 unisys, 41
 BSOntoSet, 44
 BSOntoVector, 44
 DB_CANNOT_CONNECT, 44
 DB_CANNOT_INSERT, 44
 DB_OK, 44
 dbStatus, 44
 setMiriamURN, 44
 toBSONArray, 44
 unisys::Annotation, 45
 Annotation, 48
 initField, 48
 setDefinition, 48
 setEvidence, 48
 unisys::BIOPAX2, 63
 BIOPAX2, 65
 begin, 65
 biopax, 65
 biopaxObjMap, 65
 end, 65
 getOWLNode, 65
 objects, 65
 unisys::BIOPAX_obj, 66
 BIOPAX_obj, 67
 getAttribute, 67
 getAttributeName, 67
 getAttributeValue, 67
 getChildNode, 67
 getId, 67
 getPropertyValue, 67
 getType, 67
 nAttribute, 68
 nChildNode, 68
 object, 68
 unisys::BatchInsert, 49
 BatchInsert, 51
 insert, 51
 unisys::BioObject, 59
 addAnnotation, 62
 addDataXref, 62
 addEvidence, 62
 addSynonyms, 62
 BioObject, 62
 createIdPair, 62
 initField, 62
 setComment, 62
 setDataPrimarySource, 63
 setMainName, 63
 setType, 63
 unisys::BioSource, 68
 BioSource, 71
 initField, 71
 setCellState, 71, 72
 setCellType, 72
 setSciName, 72
 setStain, 72
 setTaxonomicId, 72
 setTissueType, 72
 unisys::BiochemicalReaction, 51
 addLeft, 54
 addRight, 54
 BiochemicalReaction, 54
 initField, 54
 setConversionDirection, 55
 toRXNString, 55
 unisys::BiochemicalReactionWithTransport, 55
 addExport, 58
 addImport, 58
 addLeftIn, 58
 addLeftOut, 58
 addRightIn, 59
 addRightOut, 59

BiochemicalReactionWithTransport, 58
initField, 59
setInteractionKey, 59
toRXNString, 59
unisys::CellularLocation, 72
 CellularLocation, 75
 initField, 75
 setEvidence, 75
 setLocation, 75
unisys::ChEBIOWL, 76
 ChEBIOWL, 77
 classList, 77
 getClassList, 77
 getFileAnnotation, 77
 getVersion, 77
 owlOntology, 77
 parse, 77
unisys::ChEBIOWLClass, 78
 AnnotationProperties, 80
 ChEBIOWLClass, 80
 chebild, 80
 getAnnotationProperties, 80
 getFormula, 80
 getId, 80
 getInChi, 80
 getInChiKey, 80
 getLabel, 80
 getPropertyList, 80
 getSMILES, 80
 getSpAnnotationProperty, 80
 getSpSubClassOf, 80
 getSubClassOf, 80
 inchi, 80
 inchiKey, 80
 label, 81
 parse, 80
 parseAnnotation, 80
 parseSubClassOf, 80
 smiles, 81
 subClassOf, 81
unisys::ClassBean, 81
 ClassBean, 84
 definitions, 84
 fullId, 84
 getDefinitions, 84
 getFullId, 84
 getId, 84
 getLabel, 84
 getRelations, 84
 getSynonyms, 84
 getType, 84
 id, 84
 label, 85
 relationStruct, 83
 relations, 85
 synonyms, 85
 type, 85
unisys::ClassBean::relationDataStructure, 191
 count, 192
 getIdList, 192
 list, 192
 type, 192
unisys::Complex, 85
 addComplexMember, 88
 addSpecificKineticParameter, 88
 Complex, 88
 initField, 88
unisys::ConnectionError, 88
 what, 90
unisys::Control, 91
 addControlType, 93
 addModificationType, 93
 addPhenotype, 93
 Control, 93
 initField, 93
 setControlled, 93
 setController, 94
unisys::Conversion, 94
 addKineticLaw, 97
 Conversion, 97
 initField, 97
 setFunctional, 97
 setSpontaneous, 97
unisys::DNA, 107
 addDNARegion, 110
 DNA, 110
 initField, 110
 setLength, 110
 setSource, 110
unisys::DNARegion, 111
 addTranscriptionProduct, 113
 DNARegion, 113
 initField, 113
 setDNASource, 113
 setPosition, 113
unisys::DataError, 100
 DataError, 101
 what, 101
unisys::DataObj, 102
 addFieldSet, 105
 addWithCheck, 105
 addWithOutCheck, 105
 append, 105
 appendArray, 105
 appendAs, 105
 appendAsNumber, 105
 appendCode, 105
 appendDBRef, 105
 appendElements, 105
 appendElementsUnique, 105
 appendNull, 105
 appendRegex, 105
 appendTimeStamp, 105
 appendTimeT, 105
 data, 107
 DataObj, 105

fieldSet, 107
 genOID, 105
 getField, 106
 hasField, 106
 initField, 106
 isOwned, 106
 isValid, 106
 md5, 106
 operator<, 106
 operator<=, 106
 operator>, 106
 operator>=, 106
 operator==, 106
 removeField, 106
 requireField, 107
 set, 106
 sort, 106
 toBSONObj, 106
 toString, 106
 valid, 107
unisys::Database, 97
 connect, 99
 Database, 99
 dbname, 99
 fetchDBRef, 99
 getdbname, 99
 setdbname, 99
 Updater, 99
unisys::Evidence, 114
 Evidence, 116
 initField, 116
 insertPublicationSupport, 116
 setBiologicalSource, 116
 setConfidence, 116
 setEvidenceSource, 117
 setExperimentMethod, 117
unisys::GeneticInteraction, 117
 GeneticInteraction, 120
 initField, 120
 setInteractionType, 120
 setPhenotype, 120
unisys::IdRef, 121
 getId, 123
 getNS, 123
 IdRef, 123
 initField, 123
 initMember, 123
 isValid, 124
unisys::IntIdRef, 128
 initMember, 130
 IntIdRef, 130
 setId, 130
unisys::Interaction, 124
 callInteractionKey, 127
 getParticipant, 127
 initField, 127
 Interaction, 127
 setInteractionKey, 127
 setParticipant, 127
 toBSONObj, 127
unisys::KineticParameter, 130
 initField, 133
 KineticParameter, 133
 operator<, 133
 setEvidence, 133
 setTerm, 134
 setUnit, 134
 setValue, 134
unisys::Literal, 134
 initField, 136
 Literal, 136
 setComment, 136
unisys::LiteralBSON, 136
 LiteralBSON, 137
unisys::MathML, 137
 initField, 140
 MathML, 140
 setEvidence, 140
 setMathMLDetail, 140
unisys::Metadata, 140
 initField, 143
 Metadata, 143
 setDetail, 143
unisys::Miriam, 143
 id, 147
 increaseVer, 146
 isValid, 146
 Miriam, 145
 ns, 147
 operator<, 146
 operator++, 146
 operator==, 146
 set, 146
 setUri, 146
 toDBId, 146
 toDBIdWithVer, 146
 told, 146
 toURI, 146
 toURIWithVer, 146
 version, 147
unisys::MolecularInteraction, 147
 addLeft, 150
 addRight, 150
 initField, 150
 MolecularInteraction, 150
 setKineticLaw, 150
unisys::OBOXML, 154
 header, 156
 OBOXML, 156
 source, 156
 stanzas, 156
 stanzasBegin, 156
 stanzasEnd, 156
unisys::Object, 151
 addOntoRelationship, 153
 addRelation, 153

createRelationInsert, 153
createRelationRemove, 153
createTrack, 153
getRelation, 153
initField, 153
Object, 153
relationMap, 154
setId, 154
unisys::OntoldRef, 156
initMember, 159
OntoldRef, 159
setId, 159
unisys::OntoRelationship, 162
initField, 165
OntoRelationship, 165
setEvidence, 165
setOntoRelationship, 165
setRelationType, 165
setRelationWith, 165
unisys::Ontology, 159
initField, 162
Ontology, 162
setDefinition, 162
setNS, 162
setTerm, 162
unisys::PEIdRef, 168
initMember, 171
PEIdRef, 171
setId, 171
unisys::PSIMI, 177
attributeList, 181
availabilityList, 181
experimentList, 181
getNodeAttributeList, 180
getNodeAvailabilityList, 180
getNodeExperimentList, 180
getNodeInteractionList, 181
getNodeInteractorList, 181
getNodeSource, 181
interactionList, 181
interactorList, 181
mapOfStringString, 180
mapOfXMLNode, 180
PSIMI, 180
parse, 181
source, 181
unisys::ParsingError, 166
ParsingError, 167
what, 167
unisys::PhysicalEntity, 171
addCellularLocation, 174
addSubRegion, 174
initField, 174
PhysicalEntity, 174
unisys::Protein, 174
addRNASource, 177
addSpecificKineticParameter, 177
initField, 177
Protein, 177
unisys::Query, 182
databaseHandle, 184
explain, 184
perform, 184
performOne, 184
Query, 184
query, 184
queryById, 184
setDbHandle, 184
setQuery, 184
sort, 184
toString, 184
where, 184
unisys::QueryError, 185
QueryError, 186
what, 186
unisys::REST, 192
~REST, 196
buffer, 197
callback, 196
cleanString, 196
curlHandle, 197
curlResult, 197
getResult, 196
getResultInXML, 196
globalClean, 196
globallInit, 196
isResponseError, 196
perform, 196
performHTTPGET, 196
performPOST, 196
print, 196
REST, 196
result, 197
xmlResult, 197
unisys::RESTServiceError, 219
RESTServiceError, 221
what, 221
unisys::RNA, 222
addPosition, 225
addTranslationProduct, 225
initField, 225
RNA, 225
setDNARegionSource, 225
unisys::Relation, 187
initField, 190
isValid, 190
Relation, 190
setCoefficient, 190
setEvidence, 190
setRelationWith, 190
setType, 190
toRXNString, 190
unisys::RestBioMart, 197
bioMartVersion, 200
getServiceURL, 200
query, 200

RestBioMart, 200
 serviceURL, 200
 setServiceURL, 200
 unisys::RestBioMartResult, 200
 bioMartVersion, 202
 header, 202
 responseXML, 202
 RestBioMart, 202
 RestBioMartResult, 202
 unisys::RestBioPortal, 202
 baseURL, 206
 getSpecificOntology, 205
 listAllLastestOntologies, 205
 RestBioPortal, 205
 search, 205
 term, 206
 unisys::RestBioPortalResult, 206
 accessDate, 210
 accessedResource, 210
 errorCode, 210
 errorLongMessage, 210
 getAccessDate, 209
 getAccessedResource, 209
 getError, 209
 isError, 209
 parseError, 209
 RestBioPortalResult, 209
 unisys::RestBioPortalSearchResult, 210
 getOntologyHitList, 213
 getSearchResultList, 213
 nSearchResults, 213
 ontologyHitList, 213
 RestBioPortalSearchResult, 213
 searchResultList, 213
 unisys::RestBioPortalTermResult, 214
 classBean, 216
 getClassBean, 216
 RestBioPortalTermResult, 216
 unisys::RestMiriam, 216
 baseURL, 219
 resolve, 219
 RestMiriam, 219
 unisys::Score, 226
 initField, 228
 Score, 228
 setScoreType, 228
 setUnit, 228
 setValue, 228
 unisys::SmallMolecule, 229
 addSpecificKineticParameter, 231
 initField, 231
 setFormula, 231
 setInChi, 231
 setInChiKey, 231
 setSMILES, 232
 SmallMolecule, 231
 unisys::Stanza, 232
 getDefinition, 234
 getId, 234
 getNS, 234
 getName, 234
 getRelationship, 234
 isObsolete, 234
 Stanza, 234
 toXML, 234
 xml, 234
 unisys::SubRegion, 234
 addPosition, 237
 initField, 237
 setAnnotation, 237
 setStrand, 237
 SubRegion, 237
 unisys::Tracking, 239
 initField, 242
 toString, 242
 Tracking, 242
 unisys::Transport, 242
 addExport, 245
 addImport, 245
 initField, 245
 toRXNString, 245
 Transport, 245
 unisys::UniSysError, 246
 ~UniSysError, 247
 errorMsg, 247
 unisys::UpdateError, 247
 UpdateError, 248
 what, 248
 unisys::Updater, 249
 checkIdPair, 251
 databaseHandle, 254
 idConverter, 252
 insert, 252
 remove, 252, 253
 setDbHandle, 253
 update, 253
 updateIdPair, 254
 updateRelationAdd, 254
 updateRelationDel, 254
 Updater, 251
 unisys::Xref, 272
 initField, 275
 operator<, 275
 setDetail, 276
 setId, 276
 setType, 276
 Xref, 275
 unisysdb.h, 295
 update
 unisys::Updater, 253
 updateAttribute
 Updating Nodes, 28
 updateAttribute_WOSD
 ???_WOSD functions., 33, 34
 updateClear
 Updating Nodes, 29

updateClear_WOSD
 ???_WOSD functions., 34

UpdateError
 unisys::UpdateError, 248

updateIdPair
 unisys::Updater, 254

updateName
 Updating Nodes, 29

updateName_WOSD
 ???_WOSD functions., 34

updateRelationAdd
 unisys::Updater, 254

updateRelationDel
 unisys::Updater, 254

updateText
 Updating Nodes, 29

updateText_WOSD
 ???_WOSD functions., 34

Updater
 unisys::Database, 99
 unisys::Updater, 251

updater.h, 295

Updating Nodes, 28
 updateAttribute, 28
 updateClear, 29
 updateName, 29
 updateText, 29

valid
 unisys::DataObj, 107

version
 unisys::Miriam, 147

what
 unisys::ConnectionError, 90
 unisys::DataError, 101
 unisys::ParsingError, 167
 unisys::QueryError, 186
 unisys::RESTServiceError, 221
 unisys::UpdateError, 248

where
 unisys::Query, 184

writeToFile
 Parsing XML files/strings to an XMLNode structure
 and Rendering XMLNode's to files/string., 20

XMLNode
 char_encoding_Big5, 263
 char_encoding_GB2312, 263
 char_encoding_GBK, 263
 char_encoding_ShiftJIS, 263
 char_encoding_UTF8, 263
 char_encoding_error, 263
 char_encoding_legacy, 263

XMLAttribute, 255
 lpszName, 255
 lpszValue, 255
 xmlParser.h, 299

XMLCHAR
 xmlParser.h, 299

 XMLCSTR
 xmlParser.h, 299

 XMLCharEncoding
 XMLNode, 262, 263

 XMLClear, 255
 lpszCloseTag, 256
 lpszOpenTag, 256
 lpszValue, 256
 xmlParser.h, 299

 XMLDLLENTRY
 xmlParser.h, 299

 XMLElementPosition
 xmlParser.h, 299

 XMLElementType
 xmlParser.h, 299, 300

 XMLError
 xmlParser.h, 300

 XMLNode, 256
 ~XMLNode, 263
 addAttribute_priv, 263
 addChild_priv, 263
 addClear_priv, 263
 addText_priv, 263
 addToOrder, 263
 CreateXMLStringR, 263
 d, 264
 detachFromParent, 264
 emptyTheNode, 264
 emptyXMLAttribute, 264
 emptyXMLClear, 264
 emptyXMLNode, 264
 exactMemory, 264
 findPosition, 264
 getVersion, 264
 indexClear, 264
 indexText, 264
 maybeAddTxt, 264
 operator=, 264
 parseClearTag, 264
 ParseXMLElement, 264
 removeOrderElement, 264
 XMLCharEncoding, 262, 263
 XMLNode, 263
 XMLNodeData, 262
 XMLNode, 263
 xmlParser.h, 300

 XMLNode::XMLNodeDataTag, 267
 isDeclaration, 268
 lpszName, 268
 nAttribute, 268
 nChild, 268
 nClear, 268
 nText, 268
 pAttribute, 268
 pChild, 268
 pClear, 268
 pOrder, 268

pParent, 268
 pText, 268
 ref_count, 268
XMLNodeContents, 264
 attrib, 266
 child, 266
 clear, 266
 etype, 266
 text, 266
 xmlParser.h, 300
XMLNodeData
 XMLNode, 262
XMLParserBase64Tool, 268
 ~XMLParserBase64Tool, 270
 alloc, 270
 buf, 271
 buflen, 271
 decode, 270
 decodeSize, 270
 encode, 270
 encodeLength, 271
 freeBuffer, 271
 Helper class to include binary data inside XML strings using "Base64 encoding", 40
XMLParserBase64Tool, 270
XMLParserBase64Tool, 270
XMLResults, 271
 error, 272
 nColumn, 272
 nLine, 272
 xmlParser.h, 300
XMLSTR
 xmlParser.h, 299
xml
 unisys::Stanza, 234
xmlParser.h
 eNodeAttribute, 300
 eNodeChild, 300
 eNodeClear, 300
 eNodeNULL, 300
 eNodeText, 300
 eXMLErrorBase64DataSizelsNotMultipleOf4, 301
 eXMLErrorBase64DecodeBufferTooSmall, 301
 eXMLErrorBase64DecodeIllegalCharacter, 301
 eXMLErrorBase64DecodeTruncatedData, 301
 eXMLErrorCannotOpenWriteFile, 301
 eXMLErrorCannotWriteFile, 301
 eXMLErrorCharConversionError, 301
 eXMLErrorCharacterCharCodeAbove255, 301
 eXMLErrorEmpty, 301
 eXMLErrorNotFound, 301
 eXMLErrorFirstTagNotFound, 301
 eXMLErrorMissingEndTag, 300
 eXMLErrorMissingEndTagName, 301
 eXMLErrorMissingTagName, 301
 eXMLErrorNoElements, 301
 eXMLErrorNoXMLTagFound, 301
 eXMLErrorNone, 300
 eXMLErrorUnexpectedToken, 301
 eXMLErrorUnknownCharacterEntity, 301
 eXMLErrorUnmatchedEndClearTag, 301
 eXMLErrorUnmatchedEndTag, 301
xmlParser.h, 297
 _CXML, 299
 FALSE, 299
 TRUE, 299
 XMLAttribute, 299
 XMLCHAR, 299
 XMLCSTR, 299
 XMLClear, 299
 XMLDLLENTRY, 299
 XMLElementPosition, 299
 XMLElementType, 299, 300
 XMLError, 300
 XMLNode, 300
 XMLNodeContents, 300
 XMLResults, 300
 XMLSTR, 299
xmlResult
 unisys::REST, 197
xmltoa
 ato? like functions, 38
xmltob
 ato? like functions, 38
xmltoc
 ato? like functions, 38
xmltof
 ato? like functions, 38
xmltoi
 ato? like functions, 38
xmltol
 ato? like functions, 38
Xref
 unisys::Xref, 275