# KFoundation v1.0.2

Generated by Doxygen 1.8.4

Wed Aug 26 2015 16:32:37

# **Contents**

1	Ove	rview		1
2	Mod	ule Inde	ex	5
	2.1	Module	es	5
3	Hiera	archica	I Index	7
	3.1	Class	Hierarchy	7
4	Clas	s Index	•	11
	4.1	Class	List	11
5	Mod	ule Doo	cumentation	15
	5.1	Macro	s and Typedefs	15
		5.1.1	Detailed Description	16
		5.1.2	Macro Definition Documentation	16
			5.1.2.1 IS_NULL	16
			5.1.2.2 KF_EXPORT	16
			5.1.2.3 KF_NOP	16
			5.1.2.4 NOT_NULL	16
		5.1.3	Enumeration Type Documentation	16
			5.1.3.1 kf_comparison_t	16
	5.2	Except	tions	17
		5.2.1	Detailed Description	17
	5.3	Memo	ry Management	18
		5.3.1	Detailed Description	18
		5.3.2	Macro Definition Documentation	18
			5.3.2.1 AS	18
			5.3.2.2 ISA	18
	5.4	Input/C	Output	19
		5.4.1	Detailed Description	19
	5.5	Mutithi	reading	20
		5.5.1	Detailed Description	20
		E E O	Macra Definition Decumentation	20

iv CONTENTS

			5.5.2.1 KF_SYNCHRONIZED	20
	5.6	Wrappe	ers and Containers	1!
		5.6.1	Detailed Description	21
	5.7	Utilities	3	2
		5.7.1	Detailed Description	22
	5.8	Object	Serialization and Deserialization	23
		5.8.1	Detailed Description	23
	5.9	Range	Arithmatics	24
		5.9.1	Detailed Description	24
6	Clas	s Docui	mentation 2	25
	6.1	kfound	ation::Array< T > Class Template Reference	25
		6.1.1	Detailed Description	26
		6.1.2	Constructor & Destructor Documentation	26
			6.1.2.1 Array	26
		6.1.3	Member Function Documentation	26
			6.1.3.1 at	26
			6.1.3.2 at	26
			6.1.3.3 contains	27
			6.1.3.4 indexOf	27
			6.1.3.5 indexOf	27
			6.1.3.6 insert	27
			6.1.3.7 insert	28
			6.1.3.8 pop	28
			6.1.3.9 push	28
			6.1.3.10 push	28
			6.1.3.11 remove	28
	6.2	kfound		28
		6.2.1	·	29
	6.3			29
		6.3.1	·	30
		6.3.2		30
				30
		6.3.3		30
			•	30
				30
	0.4	Later of	Ü	30
	6.4		and the process of the control of th	30
		6.4.1	Detailed Description	
		6.4.2	Constructor & Destructor Documentation	<i>,</i> I

CONTENTS

		6.4.2.1	BufferInputStream	31
	6.4.3	Member	Function Documentation	31
		6.4.3.1	isMarkSupported	31
		6.4.3.2	mark	32
		6.4.3.3	peek	32
		6.4.3.4	read	32
		6.4.3.5	read	32
		6.4.3.6	reset	32
		6.4.3.7	skip	33
6.5	kfound	ation::Buff	ferOutputStream Class Reference	33
	6.5.1	Detailed	Description	33
	6.5.2	Construc	ctor & Destructor Documentation	34
		6.5.2.1	BufferOutputStream	34
	6.5.3	Member	Function Documentation	34
		6.5.3.1	close	34
		6.5.3.2	getData	34
		6.5.3.3	write	34
		6.5.3.4	write	34
		6.5.3.5	write	34
6.6	kfound	ation::Log	ger::Channel Class Reference	35
	6.6.1	Detailed	Description	35
	6.6.2	Construc	ctor & Destructor Documentation	35
		6.6.2.1	Channel	35
		6.6.2.2	Channel	36
	6.6.3	Member	Function Documentation	36
		6.6.3.1	checkLevel	36
		6.6.3.2	checkName	36
		6.6.3.3	isSilent	36
		6.6.3.4	setFormat	36
		6.6.3.5	setLevel	37
		6.6.3.6	setSilent	37
6.7	kfound	ation::Cod	deLocation Class Reference	37
	6.7.1	Detailed	Description	38
	6.7.2	Member	Function Documentation	38
		6.7.2.1	set	38
		6.7.2.2	set	38
		6.7.2.3	setByteIndex	38
		6.7.2.4	setCharIndex	38
		6.7.2.5	setCol	39
		6.7.2.6	setLine	39

vi CONTENTS

6.8	kfounda	undation::CodeRange Class Reference			
	6.8.1	Detailed Description	40		
	6.8.2	Constructor & Destructor Documentation	40		
		6.8.2.1 CodeRange	40		
6.9	kfounda	ation::CollectionToken Class Reference	40		
	6.9.1	Detailed Description	40		
6.10	kfounda	ation::Condition Class Reference	41		
	6.10.1	Detailed Description	41		
	6.10.2	Member Function Documentation	41		
		6.10.2.1 block	41		
		6.10.2.2 release	41		
6.11	kfounda	ation::Direction Class Reference	41		
	6.11.1	Detailed Description	42		
	6.11.2	Member Enumeration Documentation	42		
		6.11.2.1 component_t	42		
	6.11.3	Member Function Documentation	42		
		6.11.3.1 set	42		
6.12	kfounda	ation::DirectionIterator Class Reference	43		
	6.12.1	Detailed Description	43		
	6.12.2	Constructor & Destructor Documentation	43		
		6.12.2.1 DirectionIterator	43		
6.13	kfounda	ation::Double Class Reference	44		
	6.13.1	Detailed Description	44		
	6.13.2	Constructor & Destructor Documentation	44		
		6.13.2.1 Double	44		
	6.13.3	Member Function Documentation	44		
		6.13.3.1 get	44		
		6.13.3.2 parse	44		
		6.13.3.3 set	45		
6.14	kfounda	ation::EndCollectionToken Class Reference	45		
	6.14.1	Detailed Description	45		
6.15	kfounda	ation::EndObjectToken Class Reference	45		
	6.15.1	Detailed Description	46		
6.16	kfounda	ation::FileInputStream Class Reference	46		
	6.16.1	Detailed Description	47		
	6.16.2	Constructor & Destructor Documentation	47		
		6.16.2.1 FileInputStream	47		
		6.16.2.2 FileInputStream	47		
	6.16.3	Member Function Documentation	47		
		6.16.3.1 isMarkSupported	47		

CONTENTS vii

		6.16.3.2 mark	48
		6.16.3.3 peek	48
		6.16.3.4 read	48
		6.16.3.5 read	48
		6.16.3.6 reset	48
		6.16.3.7 skip	49
6.17	kfounda	ation::FileOutputStream Class Reference	49
	6.17.1	Detailed Description	50
	6.17.2	Constructor & Destructor Documentation	50
		6.17.2.1 FileOutputStream	50
	6.17.3	Member Function Documentation	50
		6.17.3.1 close	50
		6.17.3.2 isLocked	50
		6.17.3.3 lock	50
		6.17.3.4 truncate	51
		6.17.3.5 unlock	51
		6.17.3.6 write	51
		6.17.3.7 write	51
		6.17.3.8 write	51
6.18	kfounda	ation::IndexOutOfBoundException Class Reference	52
	6.18.1	Detailed Description	52
	6.18.2	Constructor & Destructor Documentation	52
		6.18.2.1 IndexOutOfBoundException	52
6.19	kfounda	ation::InputStream Class Reference	52
	6.19.1	Detailed Description	53
	6.19.2	Member Function Documentation	53
		6.19.2.1 isMarkSupported	53
		6.19.2.2 mark	53
		6.19.2.3 peek	53
		6.19.2.4 read	54
		6.19.2.5 read	54
		6.19.2.6 reset	54
		6.19.2.7 skip	54
6.20		ation::Int Class Reference	55
		Detailed Description	55
	6.20.2		55
		6.20.2.1 Int	55
		6.20.2.2 Int	56
	6.20.3	Member Function Documentation	56
		6.20.3.1 get	56

viii CONTENTS

		6.20.3.2 parse	56
		6.20.3.3 set	56
		6.20.3.4 toHexString	56
6.21	kfounda	ation::InternetAddress Class Reference	56
	6.21.1	Detailed Description	57
	6.21.2	Member Enumeration Documentation	57
		6.21.2.1 class_t	57
	6.21.3	Constructor & Destructor Documentation	58
		6.21.3.1 InternetAddress	58
		6.21.3.2 InternetAddress	58
		6.21.3.3 InternetAddress	58
	6.21.4	Member Function Documentation	58
		6.21.4.1 copyWithPort	58
6.22	kfounda	ation::InternetInputStream Class Reference	58
	6.22.1	Detailed Description	59
	6.22.2	Constructor & Destructor Documentation	60
		6.22.2.1 InternetInputStream	60
	6.22.3	Member Function Documentation	60
		6.22.3.1 bind	60
		6.22.3.2 isBound	60
		6.22.3.3 isMarkSupported	60
		6.22.3.4 listen	61
		6.22.3.5 mark	62
		6.22.3.6 peek	62
		6.22.3.7 read	62
		6.22.3.8 read	62
			63
		6.22.3.10 skip	63
		6.22.3.11 unbind	63
6.23	kfounda	ation::InternetOutputStream Class Reference	63
	6.23.1	Detailed Description	64
	6.23.2	Constructor & Destructor Documentation	64
		6.23.2.1 InternetOutputStream	64
	6.23.3	Member Function Documentation	64
		6.23.3.1 close	64
			64
		6.23.3.3 write	65
		6.23.3.4 write	65
			65
6.24	kfounda	ation::InvalidFormatException Class Reference	65

CONTENTS

	6.24.1	Detailed Description	65
	6.24.2	Constructor & Destructor Documentation	66
		6.24.2.1 InvalidFormatException	66
6.25	kfounda	ation::InvalidPointerException Class Reference	66
	6.25.1	Detailed Description	66
6.26	kfounda	ation::IOException Class Reference	66
	6.26.1	Detailed Description	66
	6.26.2	Constructor & Destructor Documentation	67
		6.26.2.1 IOException	67
6.27	kfounda	ation::KFException Class Reference	67
	6.27.1	Detailed Description	67
	6.27.2	Constructor & Destructor Documentation	68
		6.27.2.1 KFException	68
		6.27.2.2 KFException	68
	6.27.3	Member Function Documentation	68
		6.27.3.1 getMessage	68
		6.27.3.2 serialize	68
		6.27.3.3 setName	68
6.28	kfounda	ation::Logger Class Reference	68
	6.28.1	Detailed Description	69
	6.28.2	Member Enumeration Documentation	71
		6.28.2.1 level_t	71
	6.28.3	Member Function Documentation	71
		6.28.3.1 addChannel	71
		6.28.3.2 addChannel	71
		6.28.3.3 getChannelByName	71
		6.28.3.4 log	71
		6.28.3.5 log	71
		6.28.3.6 setLevel	72
6.29	kfounda	ation::LongInt Class Reference	72
	6.29.1	Detailed Description	73
	6.29.2	Member Enumeration Documentation	73
		6.29.2.1 encoding_t	73
	6.29.3	Constructor & Destructor Documentation	73
		6.29.3.1 LongInt	73
		6.29.3.2 LongInt	73
	6.29.4	Member Function Documentation	73
		6.29.4.1 get	73
		6.29.4.2 parse	73
		6.29.4.3 set	74

CONTENTS

6.30	kfounda	ation::ManagedArray< T > Class Template Reference	74
	6.30.1	Detailed Description	75
	6.30.2	Constructor & Destructor Documentation	75
		6.30.2.1 ManagedArray	75
		6.30.2.2 ~ManagedArray	75
	6.30.3	Member Function Documentation	75
		6.30.3.1 at	75
		6.30.3.2 indexOf	76
		6.30.3.3 indexOf	76
		6.30.3.4 insert	76
		6.30.3.5 pop	76
		6.30.3.6 push	77
		6.30.3.7 remove	77
		6.30.3.8 setSize	77
6.31	kfounda	ation::ManagedObject Class Reference	77
	6.31.1	Detailed Description	78
	6.31.2	Constructor & Destructor Documentation	78
		6.31.2.1 ~ManagedObject	78
6.32	kfounda	ation::MasterMemoryManager Class Reference	78
	6.32.1	Detailed Description	79
	6.32.2	Member Function Documentation	79
		6.32.2.1 registerManager	79
		6.32.2.2 unregisterManager	79
		6.32.2.3 updataTable	79
6.33	kfounda	ation::MemoryException Class Reference	79
	6.33.1	Detailed Description	30
6.34	kfounda	ation::MemoryManager Class Reference	30
	6.34.1	Detailed Description	30
	6.34.2	Member Function Documentation	30
		6.34.2.1 getTable	30
		6.34.2.2 getTableSize	31
		6.34.2.3 release	31
		6.34.2.4 remove	31
6.35	kfounda	ation::Mutex Class Reference	31
	6.35.1	Detailed Description	31
	6.35.2	Constructor & Destructor Documentation	32
		6.35.2.1 Mutex	32
6.36	kfounda	ation::NullPointerException Class Reference	33
	6.36.1	Detailed Description	33
6.37	kfounda	ation::NumericVector $<$ T $>$ Class Template Reference	33

CONTENTS xi

	6.37.1	Detailed Description	84
	6.37.2	Constructor & Destructor Documentation	84
		6.37.2.1 NumericVector	84
	6.37.3	Member Function Documentation	84
		6.37.3.1 add	84
		6.37.3.2 mul	84
		6.37.3.3 parseInt	85
		6.37.3.4 sub	85
6.38	kfounda	ation::ObjectDumpBuilderException Class Reference	85
	6.38.1	Detailed Description	85
	6.38.2	Constructor & Destructor Documentation	85
		6.38.2.1 ObjectDumpBuilderException	85
6.39	kfounda	ation::ObjectPoolMemoryManager< T > Class Template Reference	86
	6.39.1	Detailed Description	86
	6.39.2	Constructor & Destructor Documentation	86
		6.39.2.1 ObjectPoolMemoryManager	86
		6.39.2.2 ~ObjectPoolMemoryManager	87
	6.39.3	Member Function Documentation	87
		6.39.3.1 getTable	87
		6.39.3.2 getTableSize	87
		6.39.3.3 release	87
		6.39.3.4 remove	87
6.40	kfounda	ation::ObjectRecord Struct Reference	88
	6.40.1	Detailed Description	88
6.41	kfounda	ation::ObjectSerializer Class Reference	88
	6.41.1	Detailed Description	89
	6.41.2	Member Enumeration Documentation	90
		6.41.2.1 output_type_t	90
	6.41.3	Constructor & Destructor Documentation	90
		6.41.3.1 ObjectSerializer	90
		6.41.3.2 ObjectSerializer	90
	6.41.4	Member Function Documentation	90
		6.41.4.1 collection	90
		6.41.4.2 endCollection	91
		6.41.4.3 endObject	91
		6.41.4.4 member	91
		6.41.4.5 null	91
		6.41.4.6 object	91
		6.41.4.7 object	91
6.42	kfounda	ation::ObjectStreamReader Class Reference	92

xii CONTENTS

	6.42.1	Detailed Description	92
6.43	kfounda	ation::ObjectToken Class Reference	92
	6.43.1	Detailed Description	93
	6.43.2	Member Function Documentation	93
		6.43.2.1 validateClass	93
6.44	kfounda	ation::OutOfMemoryException Class Reference	93
	6.44.1	Detailed Description	94
	6.44.2	Constructor & Destructor Documentation	94
		6.44.2.1 OutOfMemoryException	94
6.45	kfounda	ation::OutputStream Class Reference	94
	6.45.1	Detailed Description	94
	6.45.2	Member Function Documentation	94
		6.45.2.1 close	94
		6.45.2.2 write	95
		6.45.2.3 write	96
		6.45.2.4 write	96
6.46	kfounda	ation::ParseException Class Reference	96
	6.46.1	Detailed Description	97
	6.46.2	Constructor & Destructor Documentation	97
		6.46.2.1 ParseException	97
		6.46.2.2 ParseException	97
		6.46.2.3 ParseException	97
	6.46.3	Member Function Documentation	97
		6.46.3.1 getBegin	97
		6.46.3.2 getEnd	97
6.47	kfounda	ation::Path Class Reference	98
	6.47.1	Detailed Description	99
	6.47.2	Constructor & Destructor Documentation	99
		6.47.2.1 Path	99
	6.47.3	Member Function Documentation	99
		6.47.3.1 changeExtension	99
		6.47.3.2 hasExtention	99
		6.47.3.3 isAbsolute	99
		6.47.3.4 parent	99
		6.47.3.5 printToStream	99
6.48	kfounda	ation::PoolObject Class Reference	100
	6.48.1	Detailed Description	100
	6.48.2	Member Function Documentation	100
		6.48.2.1 finalize	100
6.49	kfounda	ation::PPtr< T > Class Template Reference	100

CONTENTS xiii

	6.49.1	Detailed Description
	6.49.2	Constructor & Destructor Documentation
		6.49.2.1 PPtr
		6.49.2.2 PPtr
		6.49.2.3 PPtr
		6.49.2.4 PPtr
		6.49.2.5 PPtr
6.50	kfounda	tion::PredictiveParserBase Class Reference
	6.50.1	Detailed Description
	6.50.2	Constructor & Destructor Documentation
		6.50.2.1 PredictiveParserBase
	6.50.3	Member Function Documentation
		6.50.3.1 isSpace
		6.50.3.2 isValidIdentifierBeginChar
		6.50.3.3 isValidIdentifierChar
		6.50.3.4 readAllAlphabet
		6.50.3.5 readAllAlphanumeric
		6.50.3.6 readAllBeforeChar
		6.50.3.7 readAllBeforeCharSkipEscaped
		6.50.3.8 readAllBeforeNewLine
		6.50.3.9 readAllBeforeSequence
		6.50.3.10 readAllBeforeSpace
		6.50.3.11 readAllBeforeSpaceOrNewLine
		6.50.3.12 readAllNumeric
		6.50.3.13 readAlphabet
		6.50.3.14 readAlphanumeric
		6.50.3.15 readAny
		6.50.3.16 readChar
		6.50.3.17 readChar
		6.50.3.18 readIdentifier
		6.50.3.19 readIdentifierBeginChar
		6.50.3.20 readIdentifierChar
		6.50.3.21 readNewLine
		6.50.3.22 readNumber
		6.50.3.23 readNumber
		6.50.3.24 readNumber
		6.50.3.25 readNumeric
		6.50.3.26 readNumeric
		6.50.3.27 readSequence
		6.50.3.28 readSpace

XIV

		6.50.3.29 skipSpaces	112
		6.50.3.30 skipSpacesAndNewLines	112
		6.50.3.31 testChar	112
		6.50.3.32 testChar	112
		6.50.3.33 testSequence	112
		6.50.3.34 testSpace	113
6.51	kfounda	ation::ProximityIterator Class Reference	113
	6.51.1	Detailed Description	113
	6.51.2	Constructor & Destructor Documentation	113
		6.51.2.1 ProximityIterator	113
		6.51.2.2 ProximityIterator	113
	6.51.3	Member Function Documentation	114
		6.51.3.1 next	114
6.52	kfounda	ation::Ptr< T > Class Template Reference	114
	6.52.1	Detailed Description	115
	6.52.2	Constructor & Destructor Documentation	116
		6.52.2.1 Ptr	116
		6.52.2.2 Ptr	116
		6.52.2.3 ~Ptr	116
	6.52.3	Member Function Documentation	117
		6.52.3.1 cast	117
		6.52.3.2 del	117
		6.52.3.3 getRetainCount	117
		6.52.3.4 isa	17
		6.52.3.5 isValid	117
		6.52.3.6 operator*	117
		6.52.3.7 operator->	117
		6.52.3.8 operator=	18
		6.52.3.9 operator=	18
		6.52.3.10 release	118
		6.52.3.11 replace	118
		6.52.3.12 replace	118
		6.52.3.13 retain	119
		6.52.3.14 trace	119
		6.52.3.15 untrace	119
6.53	kfounda	ation::Range Class Reference	119
	6.53.1	Detailed Description	121
	6.53.2	Constructor & Destructor Documentation	121
		6.53.2.1 Range	121
	6.53.3	Member Function Documentation	121

CONTENTS xv

		6.53.3.1 border	121							
		6.53.3.2 isEmpty	121							
6.54	kfounda	ation::RangeIterator Class Reference	121							
	6.54.1	Detailed Description	122							
6.55	kfounda	ation::RefCountMemoryManager Class Reference	122							
	6.55.1	Detailed Description	123							
	6.55.2	Member Function Documentation	123							
		6.55.2.1 getTable	123							
		6.55.2.2 getTableSize	123							
		6.55.2.3 release	123							
		6.55.2.4 remove	123							
6.56	kfounda	ation::SerializingStreamer Class Reference	124							
	6.56.1	Detailed Description	124							
	6.56.2	Member Function Documentation	124							
		6.56.2.1 printToJsonStream	124							
		6.56.2.2 printToStream	124							
		6.56.2.3 printToXmlStream	124							
6.57	kfounda	ation::SPtr < T > Class Template Reference	125							
	6.57.1	Detailed Description	125							
	6.57.2	Constructor & Destructor Documentation	125							
		6.57.2.1 SPtr	125							
		6.57.2.2 SPtr	125							
6.58	kfounda	ation::StandardInputStreamAdapter Class Reference	126							
	6.58.1	Detailed Description								
	6.58.2	Member Function Documentation								
		6.58.2.1 isMarkSupported	126							
		6.58.2.2 mark	127							
		6.58.2.3 peek	127							
		6.58.2.4 read	127							
		6.58.2.5 read	127							
		6.58.2.6 reset	127							
		6.58.2.7 skip	128							
6.59	kfounda	ation::StandardOutputStreamAdapter Class Reference	128							
	6.59.1	Detailed Description	128							
	6.59.2	Member Function Documentation	129							
		6.59.2.1 close	129							
		6.59.2.2 write	129							
		6.59.2.3 write	129							
		6.59.2.4 write	129							
6.60	kfounda	ation::Logger::Stream Class Reference	129							

xvi CONTENTS

	6.60.1	Detailed Description
	6.60.2	Constructor & Destructor Documentation
		6.60.2.1 Stream
6.61	kfounda	ation::StreamDeserializer Class Reference
	6.61.1	Detailed Description
	6.61.2	Member Function Documentation
		6.61.2.1 readFromXmlStream
6.62	kfounda	ation::Streamer Class Reference
	6.62.1	Detailed Description
6.63	kfounda	ation::StringInputStream Class Reference
	6.63.1	Detailed Description
	6.63.2	Constructor & Destructor Documentation
		6.63.2.1 StringInputStream
	6.63.3	Member Function Documentation
		6.63.3.1 isMarkSupported
		6.63.3.2 mark
		6.63.3.3 peek
		6.63.3.4 read
		6.63.3.5 read
		6.63.3.6 reset
		6.63.3.7 skip
6.64	kfounda	ation::System Class Reference
	6.64.1	Detailed Description
	6.64.2	Member Enumeration Documentation
		6.64.2.1 operating_system_t
	6.64.3	Member Function Documentation
		6.64.3.1 exec
		6.64.3.2 getCurrentTimeInMiliseconds
		6.64.3.3 getExePath
		6.64.3.4 sleep
6.65	kfounda	ation::TextToken Class Reference
	6.65.1	Detailed Description
6.66	kfounda	ation::Thread Class Reference
	6.66.1	Detailed Description
	6.66.2	Constructor & Destructor Documentation
		6.66.2.1 Thread
		6.66.2.2 Thread
	6.66.3	Member Function Documentation
		6.66.3.1 isRunning
		6.66.3.2 isTheCurrentThread

CONTENTS xvii

		6.66.3.3 run	38
		6.66.3.4 start	38
6.67	kfounda	ation::Timer Class Reference	38
	6.67.1	Detailed Description	38
	6.67.2	Member Function Documentation	39
		6.67.2.1 getCpuTime	39
6.68	kfounda	ation::Token Class Reference	39
	6.68.1	Detailed Description	40
	6.68.2	Constructor & Destructor Documentation	41
		6.68.2.1 Token	41
	6.68.3	Member Function Documentation	41
		6.68.3.1 asAttribute	41
		6.68.3.2 asCollection	41
		6.68.3.3 asEndCollection	41
		6.68.3.4 asEndObject	41
		6.68.3.5 asObject	42
	6.68.4	Member Data Documentation	42
		6.68.4.1 codeRange	42
6.69	kfounda	ation::Tuple Class Reference	42
	6.69.1	Detailed Description	43
	6.69.2	Member Function Documentation	44
		6.69.2.1 negate	44
		6.69.2.2 operator%	44
		6.69.2.3 operator*	44
		6.69.2.4 operator*	44
		6.69.2.5 operator+	44
		6.69.2.6 operator+	45
		6.69.2.7 operator	45
		6.69.2.8 operator	45
		6.69.2.9 operator	45
		6.69.2.10 operator/	45
		6.69.2.11 operator/	46
6.70	kfounda	ation::Tuple1D Class Reference	46
	6.70.1	Detailed Description	47
	6.70.2	Constructor & Destructor Documentation	47
		6.70.2.1 Tuple1D	47
6.71	kfounda	ation::Tuple2D Class Reference	47
	6.71.1	Detailed Description	48
	6.71.2	Constructor & Destructor Documentation	48
		6.71.2.1 Tuple2D	48

xviii CONTENTS

		6.71.3.1 set
6.72	kfounda	ation::Tuple3D Class Reference
	6.72.1	Detailed Description
	6.72.2	Constructor & Destructor Documentation
		6.72.2.1 Tuple3D
	6.72.3	Member Function Documentation
		6.72.3.1 set
6.73	kfounda	ation::UniChar Class Reference
	6.73.1	Detailed Description
	6.73.2	Constructor & Destructor Documentation
		6.73.2.1 UniChar
		6.73.2.2 UniChar
	6.73.3	Member Function Documentation
		6.73.3.1 get
		6.73.3.2 readUtf8
		6.73.3.3 readUtf8
		6.73.3.4 readUtf8
		6.73.3.5 readUtf8
		6.73.3.6 set
		6.73.3.7 toLowerCase
		6.73.3.8 toUpperCase
		6.73.3.9 writeUtf8
		6.73.3.10 writeUtf8
6.74	kfounda	ation::XmlObjectStreamReader Class Reference
	6.74.1	Detailed Description
	6.74.2	Constructor & Destructor Documentation
		6.74.2.1 XmlObjectStreamReader

**Index** 

156

# **Chapter 1**

## **Overview**

KFoundation is a collection of C++ APIs that are essential to every programmer. Some of them like Logger have been missing in C++ standard libraries, and some of the like managed pointers (Ptr) replace existing C++ standard APIs like auto\_ptr, etc. with better fucntionality.

Most classes in KFoundation implement kfoundation::Streamer interface. This interface provides a toString() funtion and also overloads << operator for both ostream and logger stream. For example, if myObject is a Streamer, you can do the following:

```
cout << myObject << endl;
LOG << myObject << EL;
string str = myObject.toString();
```

The KFoundation APIs can be divided into the following categories (modules):

- · Types and Macros
- · Memory Management
- · Wrappers and Containers
- I/O
- · Object Serialization and Deserialization
- Utilities
- · Range Arithmatics
- · Exceptions

The following breifly explains the most essential APIs in each module.

## **Types and Macros**

KFoundation defines a set of portable types. These are, kf\_octet\_t, kf\_int8\_t, kf\_int16\_t, kf\_int32\_t, and kf\_int64\_t.

There are a serries of macros helping to detect the operating system. Thesre are, KF\_LINUX, KF\_MAC, KF\_MA-CH, KF\_SOLARIS, and KF\_FREE\_BSD. If the target machine or operating system is supported by KFoundation, KF\_SUPPORTED will be defined.

Use IS\_NULL(X) and NOT\_NULL(X) macros to make your code more readable. There are other macros like ISA(X), and AS(X) that are related to memory management module.

See all APIs here.

2 Overview

### **Memory Management**

KFoundation provides an automatic memory management solution with several advantages. Every and only subclasses of ManagedObject can take advantage of this feature. Thus ManagedObject is the root class in KFoundation inheritance hierarchy. To create a pointer to a managed object use Ptr<T>, PPtr<T>, or SPtr<T> instead a standard one. See documentation for kfoundation::Ptr for more details.

KFoundation memory management protects you from getting segmentation fault, and instread throws NullPointer-Exception or InvalidPointerException as appropriate with complete stack trace that helps you debug your program or just keep it running without a crash.

If ever needed, you may interact with running memory managers by invoking System::getMasterMemoryManager(). You may create and register your own managers as well. A memory manager should be a subclass of Memory-Manager.

See all APIs here.

## **Wrappers and Containers**

Languages like Java offer type-wrapper classes corresponding to primitive types. Thesre are good when you need to use a variable of primitive type like an object. KFoundation offers Bool, Int, LongInt, Double, and UniChar for the same purpose. Specially UniChar contains a set of very useful functions to deal with Unicode and UTF-8 encoding.

At the moment, there are only two container classes, namely Array<T> and ManagedArray<T>. ManagedArray<T> is a container for ManagedObjects, and Array<T> is a container for everything else. However Array<T> is a ManagedObject itself. NummericVector<T> is a subclass of Array<T> that implements primary mathematical operations and implements Streamer interface i.e. it has a toString() method.

See all APIs here.

### 1/0

KFoundation offers InputStream and OutputStream classes that are minimalist equivalant of ostream and istream. Thanks to their minimalist design, an extensive set of stream types could by provided by KFoundation. Thesre are, BufferInputStream, BufferOutputStream, FileInputStream, FileOutputStream, InternetInputStream, InternetOutputStream, and StringInputStream. If you need to use a standard istream or ostream object withing KFoundation StandardInputStreamAdapter and StandardOutputStreamAdapter are provided for that purpose.

See all APIs here.

### **Object Serialization and Deserialization**

KFoundation offers a powerful and intuitive way for objects to implement serialization and deserialization capabilities. Objects that can be serialized are needed to implement SerializingStreamer interface. And those that can be deserialized should implement StreamDeserializer interface. When implemented, these interfaces allow those object to be serialized/deserialized to and from every format that is supported by KFoundation, including XML, JSON, and KFOR.

To learn more about serialization read documentation for ObjectSerializer. And for deserialization check the documentation for Token class.

See all APIs here.

### **Utilites**

- Logger is a multi-channel multi-level logging utility. You often use it indirectly via LOG, LOG\_XXX, and DLO-G XXX macros.
- System class provides a set of cross-platform APIs to access system features.
- Timer is used to measure performance of a code fragment.
- PredictiveParserBase is a utility to write parsers. It is used internally to implement obejct deserialization in KFoundation.

See all APIs here.

## **Range Arithmatics**

If you usually work with multidimensional arrays and stencil copmutation, and specially if you do so in a distributed environment, these classes make your life much easier.

Tuple represents an element in n-dimentional array. It implements a large range of mathematical operations. Range is a range of indexes. This class can divide your range into desired pieces, analyse borders, detect overlapping and adjecent ranges and so on.

Rangelterator is an exciting feature that summarizes all your for loops into one small one, and is suplemented by ProximityIterator to make stensil computation with C++ as easy as it had never been.

See all APIs here.

### **Exceptions**

KFoundation exceptions provide serializable (also printable) stack trace. The root class for all exceptions is KF-Exception. You may define your own exception. Remember to always call setName() method once in constructor if you do so.

See all APIs here.

Overview

# Chapter 2

# **Module Index**

## 2.1 Modules

Here	ie	а	liet	Λf	all	modules

cros and Typedefs	. 15
eptions	. 17
mory Management	. 18
ut/Output	. 19
ithreading	. 20
appers and Containers	. 21
ties	. 22
ect Serialization and Deserialization	. 23
nge Arithmatics	. 24

6 **Module Index** 

# **Chapter 3**

# **Hierarchical Index**

# 3.1 Class Hierarchy

This	inheritance	list is	sorted	roughly,	but not	completely,	alphabet	tically:

8 Hierarchical Index

kfoundation::OutputStream	94
kfoundation::BufferOutputStream	33
kfoundation::FileOutputStream	
kfoundation::InternetOutputStream	
kfoundation::StandardOutputStreamAdapter	28
kfoundation::Path	98
kfoundation::PoolObject	
kfoundation::PredictiveParserBase	02
kfoundation::Thread	36
kfoundation::Timer	
kfoundation::UniChar	
kfoundation::XmlObjectStreamReader	
kfoundation::MasterMemoryManager	78
kfoundation::MemoryManager	80
kfoundation::ObjectPoolMemoryManager< T >	86
kfoundation::RefCountMemoryManager	22
kfoundation::Mutex	81
kfoundation::ObjectRecord	
kfoundation::Logger::Stream	
kfoundation::StreamDeserializer	
kfoundation::Streamer	
kfoundation::Bool	
kfoundation::Direction	
kfoundation::DirectionIterator	
kfoundation::Double	
kfoundation::Int	
kfoundation::InternetAddress	
kfoundation::LongInt	
kfoundation::NumericVector< T >	
kfoundation::Range	
kfoundation::SerializingStreamer	
kfoundation::CodeLocation	
kfoundation::CodeRange	
kfoundation::InternetInputStream	
kfoundation::InternetOutputStream	
kfoundation::KFException	
kfoundation::ManagedArray< T >	
kfoundation::ObjectPoolMemoryManager< T >	
kfoundation::Path	
kfoundation::Ptr $<$ T $>$	
kfoundation::PPtr $<$ T $>$	
kfoundation::SPtr $<$ T $>$	
kfoundation::PPtr< kfoundation::XmlObjectStreamReader >	
kfoundation::RefCountMemoryManager	
kfoundation::Ptr< Insertion >	
kfoundation::Ptr < kfoundation::Array < int > >	
kfoundation::Ptr < kfoundation::InputStream >	
kfoundation::Ptr< kfoundation::ManagedArray< Insertion >>	
kfoundation::Ptr< kfoundation::ManagedArray< Insertion >>	
kfoundation::Ptr < kfoundation::ManagedObject >	
kfoundation::Ptr < kfoundation::Path >	
kfoundation::Ptr< kfoundation::PredictiveParserBase >	
kfoundation::Ptr < kfoundation::Token >	
kfoundation::SPtr < kfoundation::Token >	
kfoundation::Ptr< UniString >	
kfoundation::Ptr< UniString >	
kfoundation::Timer	<b>ರ</b> ರ

3.1 Class Hierarchy 9

	kfoundation::Tuple
	kfoundation::ProximityIterator
	kfoundation::RangeIterator
	kfoundation::Tuple1D
	kfoundation::Tuple2D
	kfoundation::Tuple3D
	kfoundation::UniChar
kfo	nundation: System

10 **Hierarchical Index** 

# **Chapter 4**

# **Class Index**

## 4.1 Class List

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

kfoundation::Array< T >	
A resizable, one-dimensional indexed container	Ę
kfoundation::AttributeToken	
Represents an attribute in the parsed stream	3
kfoundation::Bool	
Wrapper class for bool type	9
kfoundation::BufferInputStream	
Input stream to read from a buffer in memory	C
kfoundation::BufferOutputStream	
Output stream used to write to a buffer in memory	3
kfoundation::Logger::Channel	
Logger channel	5
kfoundation::CodeLocation	
Encodes location of a character in a text file	1
kfoundation::CodeRange	
Encodes the location of the begining and end of a text file fragment	9
kfoundation::CollectionToken	
Represents begining of a collection in the parsed stream	(
kfoundation::Condition	
Condition variable, used to control a thread from another	H
kfoundation::Direction	
Represents directions in n-dimensional space	H
kfoundation::DirectionIterator	
Used to iterate all possible directions in an space of given dimensions	3
kfoundation::Double	
Wrapper class for double type 4	4
kfoundation::EndCollectionToken	
Represents end of a collection in the parsed stream	Ę
kfoundation::EndObjectToken	
Represents end of an object in the parsed stream	Ę
kfoundation::FileInputStream	
Input stream to read from file	e
kfoundation::FileOutputStream	
Output stream used to write data on file	ę
kfoundation::IndexOutOfBoundException	
Thrown to signal access to a nonexisting element in an array	52
kfoundation::InputStream	
Abstract interface for all input streams	52

12 Class Index

kfoundation::Int	
Wrapper for int type	55
kfoundation::InternetAddress	
Encodes an IP address and port number	56
kfoundation::InternetInputStream	
Input stream used to read from a TCP/IP port	58
kfoundation::InternetOutputStream	
Input stream used to write to TCP/IP socket	63
kfoundation::InvalidFormatException	
Thrown when an input with an invalid format is encountered	65
kfoundation::InvalidPointerException	
Thrown on attempt to access an invalid pointer	66
kfoundation::IOException	
Thrown to signal an IO-related exception	66
kfoundation::KFException	
Superclass for all exceptions in KFoundation	67
kfoundation::Logger	
Multi-channel, muti-level logger utility	68
kfoundation::LongInt	
Wrapper class for 'long int' type	72
kfoundation::ManagedArray< T >	
One-dimentional indexed collection of ManagedObjects	74
kfoundation::ManagedObject	
The root class for all classes using KFoundation framework	77
kfoundation::MasterMemoryManager	
Manages all the memory managers used in a process	78
kfoundation::MemoryException	
Used to throw exeptions related to memory	79
kfoundation::MemoryManager	
Abstract interface to be implemented by all memory managers	80
kfoundation::Mutex	
Mutex, used to prevent multiple threads to enter a critical region	81
kfoundation::NullPointerException	
Thrown on attempt to access to a null pointer	83
kfoundation::NumericVector< T >	
A subclass of Array, adds numeric operations	83
kfoundation::ObjectDumpBuilderException	
Thrown when ObjectSerializer is used in an invalid way	85
kfoundation::ObjectPoolMemoryManager< T >	
Reuses the objects in a preallocated pool whenever a new instance is needed	86
kfoundation::ObjectRecord	
Structure of memory manager's table records	88
kfoundation::ObjectSerializer	
Provides APIs to serialize an object	88
kfoundation::ObjectStreamReader	
Generic interface for utility object used to read objets from an stream of a given format	92
kfoundation::ObjectToken	
Represents begining of an object in the parsed stream	92
kfoundation::OutOfMemoryException	
Thrown on out of memory	93
kfoundation::OutputStream	
Abstract inferface for all output streams	94
kfoundation::ParseException	
Thrown when a parsing error happens	96
kfoundation::Path	
Used to represent and manipulate file and directory pathnames	98
kfoundation::PoolObject	
Superclass for all objects that are meant to be allocated by a pool memory manager	100

4.1 Class List

kfoundation::PPtr< T >	100
Passive pointer, will not release and retain automatically	100
Packs ample of basic functionalities to implement any predictive parser	102
kfoundation::ProximityIterator	
Iterates the proximity of a desired point	113
kfoundation::Ptr< T >	
Managed pointer to a class of given template type	114
kfoundation::Range	
Represents a range in n-dimensional space	119
kfoundation::RangeIterator	
Used to iterate over all points in a given range	121
kfoundation::RefCountMemoryManager	100
Reference counting memory manager	122
Objects implementing this class can be serialized into any format allowed by ObjectSerializer .	124
kfoundation::SPtr< T >	124
Static pointer, makes the pointed object immortal	125
kfoundation::StandardInputStreamAdapter	
Wraps around the given istream (C++ standard libraries) to be read as a a KFoundation input	
stream	126
kfoundation::StandardOutputStreamAdapter	
KFoundation wrapper for C++ ostream	128
kfoundation::Logger::Stream	
Log stream	129
kfoundation::StreamDeserializer	
Interface to be implemented by any class that can be deserialized from stream	130
kfoundation::Streamer	
Base class for all classes that can print information about themeselves to a std::ostream .	131
kfoundation::StringInputStream	101
Input stream to read from string	131
kfoundation::System  Provides a cross-platform way to access sytsem features	134
kfoundation::TextToken	134
Represents a text body (CDATA) in the parsed stream	136
kfoundation::Thread	
An object-oriented, cross-platform abstraction for thread	136
kfoundation::Timer	
Utility class to measure execution time of a code fragment	138
kfoundation::Token	
Represents a token in a stream	139
kfoundation::Tuple	
Represents a point in n-dimensional space	142
kfoundation::Tuple1D	
1-dimensional specialization of Tuple	146
kfoundation::Tuple2D	4.4-
2-dimensional specialization of Tuple	147
kfoundation::Tuple3D	140
3-dimensional specialization of Tuple	148
Wrapper class for unicode character	149
kfoundation::XmlObjectStreamReader	, 10
ObjectStreamReader to deserialize XML streams	155

14 Class Index

# **Chapter 5**

# **Module Documentation**

## 5.1 Macros and Typedefs

#### **Macros**

```
• #define KF_LINUX
```

is defined if the target platform is linux.

• #define KF\_SUPPORTED

is defined when the target platform is supported.

#define KF\_MAC

is defined when the target platform is Mac OS.

• #define KF\_MACH

is defined when the target platform is based on Mach kernel.

#define KF\_SOLARIS

is defined when the target platform is Solaris.

• #define KF\_FREE\_BSD

is defined when the target platform is FreeBSD.

#define NOT\_NULL(X) (X)

Tests if the argument is not null.

#define IS\_NULL(X) !(X)

Tests if the argument is null.

#define KF\_EXPORT \_\_attribute\_\_((visibility("default")))

Used to mark exported symbols.

#define KF\_NOP while(false){}

Used for consuming semicolon in other macros.

### **Typedefs**

```
• typedef unsigned char kfoundation::kf_octet_t
```

8-bit unsigned numeric type.

typedef char kfoundation::kf\_int8\_t

8-bit signed numeric type.

typedef short int kfoundation::kf\_int16\_t

16-bit signed numeric type.

• typedef int kfoundation::kf\_int32\_t

32-bit signed numeric type.

• typedef long int kfoundation::kf\_int64\_t

64-bit signed numeric type.

16 Module Documentation

### **Enumerations**

 enum kfoundation::kf\_comparison\_t { kfoundation::SMALLER = -1, kfoundation::EQUAL = 0, kfoundation::G-REATER = 1 }

Encodes the comparison result between two values.

### 5.1.1 Detailed Description

#### 5.1.2 Macro Definition Documentation

```
5.1.2.1 #define IS_NULL( X ) !(X)
```

Tests if the argument is null.

Recommended to be used to improve code readability.

```
5.1.2.2 #define KF_EXPORT __attribute__((visibility("default")))
```

Used to mark exported symbols.

Recommended to be used in place of compiler-specific method.

```
5.1.2.3 #define KF_NOP while(false){}
```

Used for consuming semicolon in other macros.

Performs no operation.

```
5.1.2.4 #define NOT_NULL( X ) (X)
```

Tests if the argument is not null.

Recommended to be used to improve code readability.

### 5.1.3 Enumeration Type Documentation

### 5.1.3.1 enum kfoundation::kf\_comparison\_t

Encodes the comparison result between two values.

#### Enumerator

SMALLER Smaller.

EQUAL Eqla.

GREATER Greater.

5.2 Exceptions 17

### 5.2 Exceptions

### Classes

class kfoundation::IndexOutOfBoundException

Thrown to signal access to a nonexisting element in an array.

• class kfoundation::InvalidFormatException

Thrown when an input with an invalid format is encountered.

• class kfoundation::InvalidPointerException

Thrown on attempt to access an invalid pointer.

· class kfoundation::IOException

Thrown to signal an IO-related exception.

• class kfoundation::KFException

Superclass for all exceptions in KFoundation.

• class kfoundation::MemoryException

Used to throw exeptions related to memory.

• class kfoundation::NullPointerException

Thrown on attempt to access to a null pointer.

• class kfoundation::ObjectDumpBuilderException

Thrown when ObjectSerializer is used in an invalid way.

· class kfoundation::OutOfMemoryException

Thrown on out of memory.

• class kfoundation::ParseException

Thrown when a parsing error happens.

### 5.2.1 Detailed Description

18 Module Documentation

### 5.3 Memory Management

#### Classes

class kfoundation::InvalidPointerException

Thrown on attempt to access an invalid pointer.

class kfoundation::ManagedArray< T >

One-dimentional indexed collection of  ${\it ManagedObjects}$ .

· class kfoundation::MasterMemoryManager

Manages all the memory managers used in a process.

• class kfoundation::MemoryException

Used to throw exeptions related to memory.

· class kfoundation::MemoryManager

Abstract interface to be implemented by all memory managers.

class kfoundation::NullPointerException

Thrown on attempt to access to a null pointer.

class kfoundation::ObjectPoolMemoryManager< T >

Reuses the objects in a preallocated pool whenever a new instance is needed.

class kfoundation::OutOfMemoryException

Thrown on out of memory.

class kfoundation::Ptr< T >

Managed pointer to a class of given template type.

class kfoundation::SPtr< T >

Static pointer, makes the pointed object immortal.

class kfoundation::PPtr< T >

Passive pointer, will not release and retain automatically.

class kfoundation::RefCountMemoryManager

Reference counting memory manager.

#### **Macros**

#define ISA(X) isa<X>()

Operates like a member function on a Ptr<T> and returns a boolean.

#define AS(X) cast<X>()

Operates like a member function on a Ptr<T> and casts it to another type.

### 5.3.1 Detailed Description

#### 5.3.2 Macro Definition Documentation

```
5.3.2.1 #define AS( X ) cast<X>()
```

Operates like a member function on a Ptr<T> and casts it to another type.

Usage:

```
Ptr<MySuperClass> myCastedObject = myObject.AS(MySuperClass);
```

#### 5.3.2.2 #define ISA( X ) isa<X>()

Operates like a member function on a Ptr < T > and returns a boolean.

If myObject.ISA (MyClass) returns true, then myObject is an instance of MyClass.

5.4 Input/Output 19

# 5.4 Input/Output

#### Classes

· class kfoundation::BufferInputStream

Input stream to read from a buffer in memory.

• class kfoundation::BufferOutputStream

Output stream used to write to a buffer in memory.

class kfoundation::CodeLocation

Encodes location of a character in a text file.

· class kfoundation::CodeRange

Encodes the location of the begining and end of a text file fragment.

class kfoundation::FileInputStream

Input stream to read from file.

class kfoundation::FileOutputStream

Output stream used to write data on file.

· class kfoundation::InputStream

Abstract interface for all input streams.

· class kfoundation::InternetAddress

Encodes an IP address and port number.

class kfoundation::InternetInputStream

Input stream used to read from a TCP/IP port.

class kfoundation::InternetOutputStream

Input stream used to write to TCP/IP socket.

class kfoundation::IOException

Thrown to signal an IO-related exception.

class kfoundation::ObjectDumpBuilderException

Thrown when ObjectSerializer is used in an invalid way.

· class kfoundation::OutputStream

Abstract inferface for all output streams.

• class kfoundation::ParseException

Thrown when a parsing error happens.

· class kfoundation::Path

Used to represent and manipulate file and directory pathnames.

· class kfoundation::PredictiveParserBase

Packs ample of basic functionalities to implement any predictive parser.

class kfoundation::StandardInputStreamAdapter

Wraps around the given istream (C++ standard libraries) to be read as a a KFoundation input stream.

class kfoundation::StandardOutputStreamAdapter

KFoundation wrapper for C++ ostream.

class kfoundation::StreamDeserializer

Interface to be implemented by any class that can be deserialized from stream.

· class kfoundation::Streamer

Base class for all classes that can print information about themeselves to a std::ostream.

· class kfoundation::StringInputStream

Input stream to read from string.

class kfoundation::XmlObjectStreamReader

ObjectStreamReader to deserialize XML streams.

# 5.4.1 Detailed Description

20 Module Documentation

# 5.5 Mutithreading

#### Classes

· class kfoundation::Condition

Condition variable, used to control a thread from another.

· class kfoundation::Mutex

Mutex, used to prevent multiple threads to enter a critical region.

· class kfoundation::Thread

An object-oriented, cross-platform abstraction for thread.

#### **Macros**

#define KF\_SYNCHRONIZED(X, Y)
 This mimics the synchronized block in Java.

### 5.5.1 Detailed Description

### 5.5.2 Macro Definition Documentation

```
5.5.2.1 #define KF_SYNCHRONIZED( X, Y)
```

#### Value:

```
{X .lock();\
{Y}\
X .unlock();} KF_NOP
```

This mimics the synchronized block in Java.

For example, with m being a Mutex,

The following syntax is also valid. Use of curly braces and semicolon is optional.

# 5.6 Wrappers and Containers

### Classes

class kfoundation::Array

A resizable, one-dimensional indexed container.

· class kfoundation::Bool

Wrapper class for bool type.

· class kfoundation::Double

Wrapper class for double type.

• class kfoundation::IndexOutOfBoundException

Thrown to signal access to a nonexisting element in an array.

class kfoundation::Int

Wrapper for int type.

· class kfoundation::LongInt

Wrapper class for 'long int' type.

- class kfoundation::ManagedArray< T >

One-dimentional indexed collection of ManagedObjects.

class kfoundation::NumericVector< T >

A subclass of Array, adds numeric operations.

· class kfoundation::UniChar

Wrapper class for unicode character.

# 5.6.1 Detailed Description

22 Module Documentation

# 5.7 Utilities

#### Classes

· class kfoundation::CodeLocation

Encodes location of a character in a text file.

· class kfoundation::CodeRange

Encodes the location of the begining and end of a text file fragment.

· class kfoundation::Logger

Multi-channel, muti-level logger utility.

• class kfoundation::ParseException

Thrown when a parsing error happens.

• class kfoundation::PredictiveParserBase

Packs ample of basic functionalities to implement any predictive parser.

• class kfoundation::System

Provides a cross-platform way to access sytsem features.

· class kfoundation::Timer

Utility class to measure execution time of a code fragment.

# 5.7.1 Detailed Description

# 5.8 Object Serialization and Deserialization

#### Classes

· class kfoundation::ObjectSerializer

Provides APIs to serialize an object.

· class kfoundation::ObjectStreamReader

Generic interface for utility object used to read objets from an stream of a given format.

· class kfoundation::Token

Represents a token in a stream.

· class kfoundation::ObjectToken

Represents begining of an object in the parsed stream.

class kfoundation::EndObjectToken

Represents end of an object in the parsed stream.

· class kfoundation::AttributeToken

Represents an attribute in the parsed stream.

class kfoundation::TextToken

Represents a text body (CDATA) in the parsed stream.

• class kfoundation::CollectionToken

Represents begining of a collection in the parsed stream.

• class kfoundation::EndCollectionToken

Represents end of a collection in the parsed stream.

· class kfoundation::SerializingStreamer

Objects implementing this class can be serialized into any format allowed by ObjectSerializer.

# 5.8.1 Detailed Description

24 Module Documentation

# 5.9 Range Arithmatics

### Classes

· class kfoundation::Direction

Represents directions in n-dimensional space.

· class kfoundation::DirectionIterator

Used to iterate all possible directions in an space of given dimensions.

· class kfoundation::ProximityIterator

Iterates the proximity of a desired point.

· class kfoundation::Range

Represents a range in n-dimensional space.

• class kfoundation::RangeIterator

Used to iterate over all points in a given range.

• class kfoundation::Tuple

Represents a point in n-dimensional space.

· class kfoundation::Tuple1D

1-dimensional specialization of Tuple.

· class kfoundation::Tuple2D

2-dimensional specialization of Tuple.

• class kfoundation::Tuple3D

3-dimensional specialization of Tuple.

# 5.9.1 Detailed Description

# **Chapter 6**

# **Class Documentation**

# 6.1 kfoundation::Array < T > Class Template Reference

A resizable, one-dimensional indexed container.

```
#include <kfoundation/Array.h>
```

#### **Public Member Functions**

• Array ()

Default constructor.

Array (T \*, kf\_int32\_t size)

Constructs a new Array copying the indicated number of items from the given C-style array.

• ∼Array ()

Deconstructor.

void remove (const kf\_int32\_t index)

Removes the item at the given index.

void push (const T &value)

Expands the array by one and sets the newly added item to the given value.

• T & push ()

Expands the array by one and returns the reference to the newly added element.

• T pop ()

Returns the value of the last element in the array, and decreases its size by one.

T & insert (const kf\_int32\_t index)

Used to insert a new element.

void insert (const kf\_int32\_t index, const T &value)

Used to insert the given value at the given index.

• void clear ()

Resets the size of this array to zero.

• bool isEmpty () const

Checks if this array is empty.

• void setSize (kf\_int32\_t size)

Adjusts the size of array to the given value.

kf\_int32\_t getSize () const

Returns the number of elements in this array.

T & at (const kf int32 t index)

Returns a reference to the value at the given index.

const T & at (const kf\_int32\_t index) const

Returns a reference to the value at the given index.

• bool contains (const T &value) const

Checks if this array contains the given value.

kf\_int32\_t indexOf (const T &value) const

Used to search for a value in the array.

• kf\_int32\_t indexOf (const kf\_int32\_t offset, const T &value) const

Searchs for the occurance of the given value starting from the given offset.

### **Static Public Attributes**

• static const kf int32 t NOT FOUND = -1

Flag returned by search methods when the desired item is not found.

#### 6.1.1 Detailed Description

template<typename T>class kfoundation::Array< T>

A resizable, one-dimensional indexed container.

This class is not designed to contain managed objects. For managed objects, use ManagedArray.

#### See Also

ManagedArray

#### 6.1.2 Constructor & Destructor Documentation

6.1.2.1 template<typename T > kfoundation::Array< T >::Array ( T \* values, kf\_int32\_t size )

Constructs a new Array copying the indicated number of items from the given C-style array.

#### **Parameters**

values	Location of the memory containing values to be copied.
size	The number of items to be copied.

#### 6.1.3 Member Function Documentation

6.1.3.1 template < typename T > T & kfoundation::Array < T >::at ( const kf\_int32\_t index ) [inline]

Returns a reference to the value at the given index.

#### **Parameters**

index The index of the item to be accessed.
---

#### **Exceptions**

Throws	IndexOutOfBoundsException if requested index is bigger or equal the size of the
	array.

6.1.3.2 template<typename T > const T & kfoundation::Array< T >::at ( const kf int32 t index ) const [inline]

Returns a reference to the value at the given index.

#### **Parameters**

:	The index of the item to be account
index	

#### **Exceptions**

Throws	IndexOutOfBoundsException if requested index is bigger or equal the size of the
	array.

6.1.3.3 template<typename T > bool kfoundation::Array< T >::contains ( const T & value ) const

Checks if this array contains the given value.

#### **Parameters**

value The value to search for.
--------------------------------

6.1.3.4 template<typename T > kf int32 t kfoundation::Array< T >::indexOf ( const T & value ) const

Used to search for a value in the array.

#### **Parameters**

value	The value to search for.

#### Returns

If found, the index of the first occurance of the given value, otherwise, NOT\_FOUND.

6.1.3.5 template<typename T > kf\_int32\_t kfoundation::Array< T >::indexOf ( const kf\_int32\_t offset, const T & value ) const

Searchs for the occurance of the given value starting from the given offset.

#### **Parameters**

offset	The index to start the search from.
value	The value to search for.

### Returns

If found, the index of the first occurance of the given value, otherwise, NOT\_FOUND.

6.1.3.6 template<typename T > T & kfoundation::Array< T >::insert ( const kf\_int32\_t index )

Used to insert a new element.

All elements at the given index and above will be shifted one index higher. Usage:

array->insert(index) = value;

#### **Parameters**

Generated on Wed Aug 26 2015 16:32:37 for KFoundation by Doxygen

index	The index to insert at.

6.1.3.7 template<typename T > void kfoundation::Array<T >::insert ( const kf\_int32\_t index, const T & value )

Used to insert the given value at the given index.

All the existing at the index and higher will be shifted one index higher.

#### **Parameters**

index	The index to insert at.
value	The value to be inserted.

### 6.1.3.8 template<typename T > T kfoundation::Array< T >::pop ( )

Returns the value of the last element in the array, and decreases its size by one.

#### **Exceptions**

TI	lander Out Of Decoration of the common temperature
Inrows	IndexOutOfBoundException if the array is empty.
	mache at a least a lea

### 6.1.3.9 template<typename T > void kfoundation::Array< T >::push ( const T & value )

Expands the array by one and sets the newly added item to the given value.

#### **Parameters**

value	Value to be pushed.

#### 6.1.3.10 template<typename T > T & kfoundation::Array< T >::push ( )

Expands the array by one and returns the reference to the newly added element.

#### Usage:

```
array->push() = value;
```

### 6.1.3.11 template < typename T > void kfoundation::Array < T >::remove ( const kf\_int32\_t index )

Removes the item at the given index.

If there are items at higher indexes, they will be shifted down by one.

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

- · ArrayDecl.h
- · Array.h

# 6.2 kfoundation::AttributeToken Class Reference

Represents an attribute in the parsed stream.

#include <kfoundation/ObjectStreamReader.h>

#### **Public Member Functions**

• AttributeToken (const CodeRange &range)

Constructor.

• bool checkName (const string &name) const

Checks if the name of this attribute equals the given parameter.

• PPtr< AttributeToken > validateName (const string &name) const

Checks if the name of this attribute equals the given parameter, if not throws a ParseException.

· void throwInvliadName () const

Throws a ParseException explanaining the name of this attribute is invalid.

• type\_t getType () const

Returns the type of this token.

#### **Static Public Attributes**

static const type\_t TYPE = Token::ATTRIBUTE
 Type this token, that is Token::ATTRIBUTE.

#### **Additional Inherited Members**

### 6.2.1 Detailed Description

Represents an attribute in the parsed stream.

#### See Also

Token ObjectStreamReader

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

- · ObjectStreamReader.h
- · ObjectStreamReader.cpp

# 6.3 kfoundation::Bool Class Reference

Wrapper class for bool type.

#include <kfoundation/Bool.h>

### **Public Member Functions**

• Bool (bool value)

Constructor.

• bool get () const

Getter method.

void set (const bool value)

Setter method.

• void printToStream (ostream &os) const

Implements compatibility with Streamer interface.

#### **Static Public Member Functions**

static string toString (const bool value)
 Returns the string representation of the given argument.

# 6.3.1 Detailed Description

Wrapper class for bool type.

#### 6.3.2 Constructor & Destructor Documentation

6.3.2.1 kfoundation::Bool::Bool (bool value)

Constructor.

Sets the internal value to the given parameter.

### 6.3.3 Member Function Documentation

**6.3.3.1** bool kfoundation::Bool::get() const [inline]

Getter method.

Returns the internal value.

6.3.3.2 void kfoundation::Bool::set ( const bool value ) [inline]

Setter method.

Sets the internal value to the given parameter.

**6.3.3.3** string kfoundation::Bool::toString (const bool value) [static]

Returns the string representation of the given argument.

**Parameters** 

value | The value to be converted to string.

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

- Bool.h
- · Bool.cpp

# 6.4 kfoundation::BufferInputStream Class Reference

Input stream to read from a buffer in memory.

#include <kfoundation/BufferInputStream.h>

### **Public Member Functions**

BufferInputStream (const kf\_octet\_t \*const buffer, const kf\_int32\_t size, bool takover)
 Constructor.

∼BufferInputStream ()

Deconstructor.

kf\_int32\_t read (kf\_octet\_t \*buffer, kf\_int32\_t nBytes)

Reads at most the given number of octets from the given buffer.

• int read ()

Reads a single octet.

• int peek ()

Reads a single octet without advancing.

kf\_int32\_t skip (kf\_int32\_t bytes)

Skips the at most the given number of octets without reading them.

• bool isEof ()

Checks if the end of stream is reached.

bool isMarkSupported ()

If returns true, mark() and reset() methods can be used.

· void mark ()

Marks the current position of the stream.

· void reset ()

Returns the stream position to where mark() was used last time.

• bool isBigEndian ()

Checks if the stream is big-endian.

### 6.4.1 Detailed Description

Input stream to read from a buffer in memory.

### 6.4.2 Constructor & Destructor Documentation

6.4.2.1 kfoundation::BufferInputStream::BufferInputStream ( const kf\_octet\_t \*const buffer, const kf\_int32\_t size, bool takeover )

Constructor.

#### **Parameters**

buffer	The buffer to read from.
size	Size of the buffer to read.
takeover	If set true the buffer will be deleted once this object is deconstructed. The default value is
	false.

#### 6.4.3 Member Function Documentation

**6.4.3.1** bool kfoundation::BufferInputStream::isMarkSupported( ) [virtual]

If returns true, mark() and reset() methods can be used.

See Also

mark()

reset()

Implements kfoundation::InputStream.

```
6.4.3.2 void kfoundation::BufferInputStream::mark() [virtual]
```

Marks the current position of the stream.

Use reset() to return to this position later and read the data again.

See Also

```
isMarkSupported()
reset()
```

Implements kfoundation::InputStream.

```
6.4.3.3 int kfoundation::BufferInputStream::peek( ) [virtual]
```

Reads a single octet without advancing.

Returns

The value of read octet, or -1 if no data is available.

Implements kfoundation::InputStream.

```
6.4.3.4 kf_int32_t kfoundation::BufferInputStream::read ( kf_octet_t * buffer, kf_int32_t nOctets ) [virtual]
```

Reads at most the given number of octets from the given buffer.

Returns the actual number of octets read.

**Parameters** 

buffer	The octets to read.
nOctets	Maximum number of octets to read.

#### Returns

The actual number of octets read.

Implements kfoundation::InputStream.

```
6.4.3.5 int kfoundation::BufferInputStream::read() [virtual]
```

Reads a single octet.

Returns

The value of read octet, or -1 if no data is available.

Implements kfoundation::InputStream.

```
6.4.3.6 void kfoundation::BufferInputStream::reset() [virtual]
```

Returns the stream position to where mark() was used last time.

If mark() is never called, it will reset to the begining of the stream.

See Also

```
isMarkSupported()
mark()
```

Implements kfoundation::InputStream.

6.4.3.7 kf\_int32\_t kfoundation::BufferInputStream::skip(kf\_int32\_t nOctets) [virtual]

Skips the at most the given number of octets without reading them.

If there are less number of octets in the stream, all available octets will be skipped.

**Parameters** 

*nOctets* | The desired number of octets to skeep.

#### Returns

The actual number of octets skipped.

Implements kfoundation::InputStream.

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

- · BufferInputStream.h
- · BufferInputStream.cpp

# 6.5 kfoundation::BufferOutputStream Class Reference

Output stream used to write to a buffer in memory.

#include <kfoundation/BufferOutputStream.h>

### **Public Member Functions**

• BufferOutputStream (const kf\_int32\_t capacity)

Constructor, creates a new stream with an internal buffer of given capacity.

∼BufferOutputStream ()

Deconstructor.

kf\_octet\_t \* getData () const

Returns the pointer to the first byte of the internal buffer.

kf\_int32\_t getSize () const

Returns the number of octets written to this stream.

• bool isBigEndian () const

Checks if the stream is big-endian.

void write (const kf\_octet\_t \*buffer, const kf\_int32\_t nBytes)

Writes the given number of octets of the given buffer to the stream.

• void write (kf\_octet\_t byte)

Writes a single octet to the stream.

void write (PPtr< InputStream > is)

Writes the available contents from the given input stream to this output stream.

• void close ()

Closes the stream.

# 6.5.1 Detailed Description

Output stream used to write to a buffer in memory.

### 6.5.2 Constructor & Destructor Documentation

6.5.2.1 kfoundation::BufferOutputStream::BufferOutputStream ( const kf\_int32\_t capacity )

Constructor, creates a new stream with an internal buffer of given capacity.

If more octets than the given capacity is written to it, the buffer size will grow exponentially.

**Parameters** 

capacity	The initial capacity of the internal buffer.

### 6.5.3 Member Function Documentation

**6.5.3.1** void kfoundation::BufferOutputStream::close() [virtual]

Closes the stream.

It will no longer be readable.

Implements kfoundation::OutputStream.

6.5.3.2 kf octet t \* kfoundation::BufferOutputStream::getData ( ) const

Returns the pointer to the first byte of the internal buffer.

Note

The returned value might change as the internal buffer expands.

6.5.3.3 void kfoundation::BufferOutputStream::write ( const kf\_octet\_t \* buffer, const kf\_int32\_t nOctets ) [virtual]

Writes the given number of octets of the given buffer to the stream.

#### **Parameters**

buffer	The octets to write.
nOctets	Number of octets to write.

Implements kfoundation::OutputStream.

**6.5.3.4** void kfoundation::BufferOutputStream::write ( kf octet t octet ) [virtual]

Writes a single octet to the stream.

### **Parameters**

octet	The octet to write

Implements kfoundation::OutputStream.

6.5.3.5 void kfoundation::BufferOutputStream::write ( PPtr < InputStream > is ) [virtual]

Writes the available contents from the given input stream to this output stream.

#### **Parameters**

is The stream to read from.

Implements kfoundation::OutputStream.

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

- · BufferOutputStream.h
- BufferOutputStream.cpp

# 6.6 kfoundation::Logger::Channel Class Reference

#### Logger channel.

#include <kfoundation/Logger.h>

#### **Public Member Functions**

Channel (const string &name, const string &fileName)

Constructor, do not use directly.

Channel (const string &name, ostream \*os)

Constructor, do not use directly.

∼Channel ()

Deconstructor.

• Channel & setLevel (level\_t level)

Sets the filtering level.

• level\_t getLevel () const

Returns the current filtering level.

bool checkLevel (level\_t lvl) const

Checks if the current filtering level equals the given parameter.

void setSilent (bool isSilent)

If the given parameter is true, this channel will no longer produce any output.

• bool isSilent () const

Checks if this channel is set to silent.

• const string & getName () const

Returns the name of this channel.

bool checkName (const string &name) const

Checks if the name of this channel is the same as the given parameter.

• Channel & setFormat (bool printHourAndMinutes, bool printSecondAndMilis, bool printLocation)

The heading of each log entry can be customized using this function.

#### 6.6.1 Detailed Description

Logger channel.

### 6.6.2 Constructor & Destructor Documentation

6.6.2.1 kfoundation::Logger::Channel::Channel ( const string & name, const string & fileName )

Constructor, do not use directly.

Instead, use Logger::addChannel().

6.6.2.2 kfoundation::Logger::Channel::Channel ( const string & name, ostream \* os )

Constructor, do not use directly.

Instread, use Logger::addChannel().

#### 6.6.3 Member Function Documentation

6.6.3.1 bool kfoundation::Logger::Channel::checkLevel ( level\_t /v/ ) const

Checks if the current filtering level equals the given parameter.

#### **Parameters**

lvl	The level to check against.
-----	-----------------------------

#### Returns

The result of comparison.

6.6.3.2 bool kfoundation::Logger::Channel::checkName ( const string & name ) const

Checks if the name of this channel is the same as the given parameter.

#### **Parameters**

name	The name to check against.
------	----------------------------

### Returns

The result of comparison.

6.6.3.3 bool kfoundation::Logger::Channel::isSilent ( ) const

Checks if this channel is set to silent.

#### See Also

setSilent()

6.6.3.4 Logger::Channel & kfoundation::Logger::Channel::setFormat ( bool printHourAndMinutes, bool printSecondAndMilisecond, bool printLocation )

The heading of each log entry can be customized using this function.

#### **Parameters**

printHourAnd-	If set true the hour and minute will be printed in HH:MM format.
Minutes	
printSecondAnd-	If set true second and milisecond will be printed in SS.m format. If printHourAndMinutes is
Milisecond	already set to true the output will be in HH:MM:SS.m format.

printLocation | If set true the location of the code is printed in [function\_name@file\_name:line\_number] | format.

6.6.3.5 Logger::Channel & kfoundation::Logger::Channel::setLevel ( level\_t level )

Sets the filtering level.

All messages with equal or higher level than the given level will be passed and the rest will be filtered.

#### Parameters 2 4 1

level	Filtering level

6.6.3.6 void kfoundation::Logger::Channel::setSilent ( bool isSilent )

If the given parameter is true, this channel will no longer produce any output.

Otherwise, output will be produced with the given filtering level applied.

**Parameters** 

isSilent | If set true this channel will be silenced, otherwise it will resume producing output.

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

- · Logger.h
- · Logger.cpp

# 6.7 kfoundation::CodeLocation Class Reference

Encodes location of a character in a text file.

#include <CodeLocation.h>

#### **Public Member Functions**

· CodeLocation ()

Default constructor, sets all values to zero.

void set (const CodeLocation & other)

Setter, copies all the values from the given argument.

void set (long int line, long int col, long int byteIndex)

Setter, sets all values except charlndex which will be set equal to byteIndex.

void set (long int line, long int col, long int charIndex, long int byteIndex)

Setter, sets all values as given in arguments.

CodeLocation & setLine (long int line)

Setter, sets the line number.

• CodeLocation & setCol (long int col)

Setter, sets the column number.

CodeLocation & setByteIndex (long int byteIndex)

Setter, sets byte index.

CodeLocation & setCharIndex (long int charIndex)

Setter, sets character index.

• long int getLine () const

Getter, returns line number.

• long int getCol () const

Getter, returns column number.

• long int getByteIndex () const

Getter, return the byte index.

• long int getCharIndex () const

Getter, returns the character index.

• void serialize (PPtr< ObjectSerializer > builder) const

Implements compatibility with SerializingStreamer interface.

# 6.7.1 Detailed Description

Encodes location of a character in a text file.

The location is expressed as line, column. Character index and byte index from the begining of the file are also available. These two may be different in UTF encoded files.

#### 6.7.2 Member Function Documentation

6.7.2.1 void kfoundation::CodeLocation::set ( long int line, long int col, long int byteIndex )

Setter, sets all values except charindex which will be set equal to byteindex.

#### **Parameters**

line	Line number
col	Column number
byteIndex	Byte index from the begining of the file.

6.7.2.2 void kfoundation::CodeLocation::set ( long int line, long int col, long int charlndex, long int bytelndex )

Setter, sets all values as given in arguments.

#### **Parameters**

line	Line number
col	Column number
charIndex	Character index from the begining of the file.
byteIndex	Byte index from the begining of the file.

#### 6.7.2.3 CodeLocation & kfoundation::CodeLocation::setByteIndex ( long int byteIndex )

Setter, sets byte index.

#### **Parameters**

byteIndex	Byte index from the begining of the file.
Reference	to self.

#### 6.7.2.4 CodeLocation & kfoundation::CodeLocation::setCharIndex ( long int charIndex )

Setter, sets character index.

**Parameters** 

charIndex Character index from the begining of the file.

Returns

Reference to self.

6.7.2.5 CodeLocation & kfoundation::CodeLocation::setCol ( long int col )

Setter, sets the column number.

**Parameters** 

col Column number

Returns

Reference to self.

6.7.2.6 CodeLocation & kfoundation::CodeLocation::setLine ( long int line )

Setter, sets the line number.

**Parameters** 

line Line number

Returns

Reference to slef.

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

- · CodeLocation.h
- · CodeLocation.cpp

# 6.8 kfoundation::CodeRange Class Reference

Encodes the location of the begining and end of a text file fragment.

#include <kfoundation/CodeRange.h>

### **Public Member Functions**

• CodeRange (const CodeLocation &begin, const CodeLocation &end)

Constructor.

• const CodeLocation & getBegin () const

Returns the begining of the range.

const CodeLocation & getEnd () const

Returns the end of the range.

• void serialize (PPtr< ObjectSerializer > serializer) const

Implements compatibility with SerializingStreamer interface.

# 6.8.1 Detailed Description

Encodes the location of the begining and end of a text file fragment.

### 6.8.2 Constructor & Destructor Documentation

6.8.2.1 kfoundation::CodeRange::CodeRange ( const CodeLocation & begin, const CodeLocation & end )

#### Constructor.

#### **Parameters**

begin	The begining of the range (inclusive).
end	The end of the range (inclusive).

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

- · CodeRange.h
- · CodeRange.cpp

# 6.9 kfoundation::CollectionToken Class Reference

Represents begining of a collection in the parsed stream.

#include <kfoundation/ObjectStreamReader.h>

#### **Public Member Functions**

CollectionToken (const CodeRange &range)

Constructor.

• type\_t getType () const

Returns the type of this token.

### **Static Public Attributes**

static type\_t TYPE = Token::COLLECTION
 Type this token, that is Token::COLLECTION.

# **Additional Inherited Members**

### 6.9.1 Detailed Description

Represents begining of a collection in the parsed stream.

#### See Also

Token

ObjectStreamReader

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

- · ObjectStreamReader.h
- · ObjectStreamReader.cpp

### 6.10 kfoundation::Condition Class Reference

Condition variable, used to control a thread from another.

```
#include <kfoundation/Condition.h>
```

### **Public Member Functions**

void block (const kf\_int64\_t timeout)

Blocks this thread until the release() is called by another thread, or the given timeout is reached.

· void block ()

Blocks this thread unless release() is called by another thread.

void release ()

Releases the thread blocked by this condition variable.

void releaseAll ()

Releases all threads blocked by this condition.

· bool isBlocked () const

Checks if this condition variable is currently blocking any thread.

### 6.10.1 Detailed Description

Condition variable, used to control a thread from another.

#### 6.10.2 Member Function Documentation

```
\textbf{6.10.2.1} \quad \textbf{void kfoundation::} \textbf{Condition::} \textbf{block ( const kf\_int64\_t \textit{timeout })} \quad \texttt{[inline]}
```

Blocks this thread until the release() is called by another thread, or the given timeout is reached.

If it is meant to be block for a given period of time dt, use the following pattern:

```
condition.block(System::getCurrentTimeInMiliseconds() + dt);
```

#### **Parameters**

timeout Target absolute time measured in miliseconds.

```
6.10.2.2 void kfoundation::Condition::release ( ) [inline]
```

Releases the thread blocked by this condition variable.

If multiple threads are blocked, one of them will be released.

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

- · Condition.h
- · Condition.cpp

### 6.11 kfoundation::Direction Class Reference

Represents directions in n-dimensional space.

```
#include <kfoundation/Direction.h>
```

# **Public Types**

enum component\_t { BACK = -1, NEUTRAL = 0, FORTH = 1 }
 Directions alongside a line.

#### **Public Member Functions**

• Direction (kf\_int8\_t size)

Constructor, creates a neutral direction of the given size.

Direction (const Direction &copy)

Copy constructor.

• kf\_int8\_t getSize () const

Returns the number of elements.

• Direction & set (kf\_int8\_t index, const component\_t value)

Setter, sets the element at the given index to the given value.

• component\_t get (kf\_int8\_t index) const

Returns the value of element at the given index.

• Direction getOpposite () const

Returns a direction opposite to this one.

• bool isCenter () const

Checks if this is a neutral direction.

· void printToStream (ostream &stream) const

Implements compatibility with Streamer interface.

# 6.11.1 Detailed Description

Represents directions in n-dimensional space.

n can be between 0 to 4.

#### 6.11.2 Member Enumeration Documentation

6.11.2.1 enum kfoundation::Direction::component\_t

Directions alongside a line.

#### Enumerator

BACK Backward.

**NEUTRAL** Netural.

FORTH Forward.

### 6.11.3 Member Function Documentation

6.11.3.1 Direction & kfoundation::Direction::set ( kf\_int8\_t index, const component\_t value ) [inline]

Setter, sets the element at the given index to the given value.

#### Returns

Reference to self.

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

- · Direction.h
- · Direction.cpp

# 6.12 kfoundation::DirectionIterator Class Reference

Used to iterate all possible directions in an space of given dimensions.

```
#include <kfoundation/DirectionIterator.h>
```

#### **Public Member Functions**

• DirectionIterator (kf\_int8\_t nDims)

Constructor.

• void first ()

Resets the iterator.

• void next ()

Moves on to the next element.

· bool hasMore () const

Checks if there is more items to iterate.

# **Additional Inherited Members**

### 6.12.1 Detailed Description

Used to iterate all possible directions in an space of given dimensions.

Usage:

```
for(DirectionIterator di(3); di.hasMore(); di.next()) {
   Direction d = di;
   ... do something with d ...
}
```

### 6.12.2 Constructor & Destructor Documentation

6.12.2.1 kfoundation::DirectionIterator::DirectionIterator ( kf\_int8\_t nDims )

Constructor.

**Parameters** 

```
nDims Number of dimensions.
```

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

- · DirectionIterator.h
- · DirectionIterator.cpp

### 6.13 kfoundation::Double Class Reference

Wrapper class for double type.

#include <kfoundation/Double.h>

#### **Public Member Functions**

• Double (const double value)

Constructor.

double get () const

Getter method.

• void set (const double v)

Setter method.

void printToStream (ostream &os) const

Implements compatibility with Streamer interface.

#### **Static Public Member Functions**

static double parse (const string &str)

Parses the given string to the corresponding double value.

# 6.13.1 Detailed Description

Wrapper class for double type.

#### 6.13.2 Constructor & Destructor Documentation

6.13.2.1 kfoundation::Double::Double ( const double value )

Constructor.

Sets the internal value to the given parameter.

#### 6.13.3 Member Function Documentation

6.13.3.1 double kfoundation::Double::get() const [inline]

Getter method.

Returns the internal value.

**6.13.3.2** double kfoundation::Double::parse ( const string & str ) [static]

Parses the given string to the corresponding double value.

**Parameters** 

str The string to be parsed.

#### Returns

The numerical value parsed from the given string.

6.13.3.3 void kfoundation::Double::set ( const double value ) [inline]

Setter method.

Sets the internal value to the given parameter.

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

- · Double.h
- · Double.cpp

### 6.14 kfoundation::EndCollectionToken Class Reference

Represents end of a collection in the parsed stream.

```
#include <kfoundation/ObjectStreamReader.h>
```

#### **Public Member Functions**

• EndCollectionToken (const CodeRange &range)

Constructor.

type\_t getType () const

Returns the type of this token.

### **Static Public Attributes**

```
    static type_t TYPE = Token::END_COLLECTION
    Type this token, that is Token::TEXT.
```

#### **Additional Inherited Members**

### 6.14.1 Detailed Description

Represents end of a collection in the parsed stream.

See Also

Token

ObjectStreamReader

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

- · ObjectStreamReader.h
- · ObjectStreamReader.cpp

# 6.15 kfoundation::EndObjectToken Class Reference

Represents end of an object in the parsed stream.

#include <kfoundation/ObjectStreamReader.h>

#### **Public Member Functions**

EndObjectToken (const CodeRange &range)

Constructor.

· bool checkClass (const string &name) const

Checks if this token represents the end of object with the given name.

· void validateClass (const string &name) const

Checks if this token represents the end of object with the given name, if not throws a ParseException.

type\_t getType () const

Returns the type of this token.

### **Static Public Attributes**

static const type\_t TYPE = Token::END\_OBJECT
 Type this token, that is Token::END\_OBJECT.

#### **Additional Inherited Members**

### 6.15.1 Detailed Description

Represents end of an object in the parsed stream.

See Also

Token ObjectStreamReader

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

- ObjectStreamReader.h
- ObjectStreamReader.cpp

# 6.16 kfoundation::FileInputStream Class Reference

Input stream to read from file.

```
#include <kfoundation/FileInputStream.h>
```

#### **Public Member Functions**

FileInputStream (PPtr< Path > path)

Constructor, opens the file pointed by the given Path object.

• FileInputStream (const string &fileName)

Constructor, opens file pointed by the given string to read.

virtual ∼FileInputStream ()

Deconstructor.

kf\_int64\_t getSize () const

Returns the size of the openned file.

• bool isOpen () const

Checks if the file is open.

• void close ()

Closes the file.

• kf\_int32\_t read (kf\_octet\_t \*buffer, const kf\_int32\_t nBytes)

Reads at most the given number of octets from the given buffer.

• int read ()

Reads a single octet.

• int peek ()

Reads a single octet without advancing.

• kf int32 t skip (kf int32 t bytes)

Skips the at most the given number of octets without reading them.

bool isEof ()

Checks if the end of stream is reached.

bool isMarkSupported ()

If returns true, mark() and reset() methods can be used.

void mark ()

Marks the current position of the stream.

· void reset ()

Returns the stream position to where mark() was used last time.

• bool isBigEndian ()

Checks if the stream is big-endian.

# 6.16.1 Detailed Description

Input stream to read from file.

#### 6.16.2 Constructor & Destructor Documentation

6.16.2.1 kfoundation::FileInputStream::FileInputStream ( PPtr< Path > path )

Constructor, opens the file pointed by the given Path object.

**Parameters** 

path The path to the file to open.

6.16.2.2 kfoundation::FileInputStream::FileInputStream ( const string & fileName )

Constructor, opens file pointed by the given string to read.

**Parameters** 

fileName The path to the file to open.

### 6.16.3 Member Function Documentation

**6.16.3.1** bool kfoundation::FileInputStream::isMarkSupported() [virtual]

If returns true, mark() and reset() methods can be used.

See Also

mark()

reset()

Implements kfoundation::InputStream.

```
6.16.3.2 void kfoundation::FileInputStream::mark() [virtual]
```

Marks the current position of the stream.

Use reset() to return to this position later and read the data again.

See Also

```
isMarkSupported()
reset()
```

Implements kfoundation::InputStream.

```
6.16.3.3 int kfoundation::FileInputStream::peek( ) [virtual]
```

Reads a single octet without advancing.

Returns

The value of read octet, or -1 if no data is available.

Implements kfoundation::InputStream.

```
6.16.3.4 kf_int32_t kfoundation::FileInputStream::read ( kf_octet_t * buffer, const kf_int32_t nOctets ) [virtual]
```

Reads at most the given number of octets from the given buffer.

Returns the actual number of octets read.

**Parameters** 

buffer	The octets to read.
nOctets	Maximum number of octets to read.

#### Returns

The actual number of octets read.

Implements kfoundation::InputStream.

```
6.16.3.5 int kfoundation::FileInputStream::read() [virtual]
```

Reads a single octet.

Returns

The value of read octet, or -1 if no data is available.

Implements kfoundation::InputStream.

```
6.16.3.6 void kfoundation::FileInputStream::reset() [virtual]
```

Returns the stream position to where mark() was used last time.

If mark() is never called, it will reset to the begining of the stream.

See Also

```
isMarkSupported()
mark()
```

Implements kfoundation::InputStream.

6.16.3.7 kf\_int32\_t kfoundation::FileInputStream::skip(kf\_int32\_t nOctets) [virtual]

Skips the at most the given number of octets without reading them.

If there are less number of octets in the stream, all available octets will be skipped.

**Parameters** 

*nOctets* | The desired number of octets to skeep.

#### Returns

The actual number of octets skipped.

Implements kfoundation::InputStream.

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

- · FileInputStream.h
- FileInputStream.cpp

# 6.17 kfoundation::FileOutputStream Class Reference

Output stream used to write data on file.

#include <kfoundation/FileOutputStream.h>

#### **Public Member Functions**

FileOutputStream (PPtr< Path > path)

Constructor, opens the file pointed by the given path object to write.

•  $\sim$ FileOutputStream ()

Deconstructor.

• bool isLocked () const

Checks if a lock is placed on the file to be read by this stream.

· void lock () const

Places a lock on the file to be read by this stream.

• void unlock () const

Removes the lock placed on this file.

• void close ()

Closes the stream.

PPtr< Path > getPath () const

Returns the path of the file being written by this stream.

bool isBigEndian () const

Checks if the stream is big-endian.

• void truncate () const

Earases the file contents and resets the stream position to the begining of the file.

void write (const kf\_octet\_t \*buffer, const kf\_int32\_t nBytes)

Writes the given number of octets of the given buffer to the stream.

void write (kf\_octet\_t byte)

Writes a single octet to the stream.

void write (PPtr< InputStream > is)

Writes the available contents from the given input stream to this output stream.

# 6.17.1 Detailed Description

Output stream used to write data on file.

### 6.17.2 Constructor & Destructor Documentation

6.17.2.1 kfoundation::FileOutputStream::FileOutputStream ( PPtr< Path > path )

Constructor, opens the file pointed by the given path object to write.

If the file already exists, the writting begins from the end of the file. If it is desired to earase the existing contents of the file, use truncate() method.

**Parameters** 

path | Path to the file to be opened.

See Also

truncate()

#### 6.17.3 Member Function Documentation

**6.17.3.1 void kfoundation::FileOutputStream::close()** [virtual]

Closes the stream.

It will no longer be readable.

Implements kfoundation::OutputStream.

6.17.3.2 bool kfoundation::FileOutputStream::isLocked ( ) const

Checks if a lock is placed on the file to be read by this stream.

Locking mechanism is used to prevent write by more than one process.

See Also

lock() unlock()

**Exceptions** 

Throws | IOException if lock status could not be obtained.

6.17.3.3 void kfoundation::FileOutputStream::lock ( ) const

Places a lock on the file to be read by this stream.

Locking mechanism is used to prevent write by more than one process.

See Also

isLocked() unlock()

#### **Exceptions**

Throws	IOException if lock could not be placed.
--------	--

6.17.3.4 void kfoundation::FileOutputStream::truncate ( ) const

Earases the file contents and resets the stream position to the begining of the file.

**Exceptions** 

Throws	IOException if the operation failed.

6.17.3.5 void kfoundation::FileOutputStream::unlock ( ) const

Removes the lock placed on this file.

Locking mechanism is used to prevent write by more than one process.

See Also

isLocked()
lock()

#### **Exceptions**

Throw	IOException if lock could not be removed.

6.17.3.6 void kfoundation::FileOutputStream::write ( const kf\_octet\_t \* buffer, const kf\_int32\_t nOctets ) [virtual]

Writes the given number of octets of the given buffer to the stream.

#### **Parameters**

buffer	The octets to write.
nOctets	Number of octets to write.

Implements kfoundation::OutputStream.

**6.17.3.7 void kfoundation::FileOutputStream::write ( kf\_octet\_t** *octet* ) [virtual]

Writes a single octet to the stream.

# Parameters

octet	The octet to write

 $Implements\ k foundation :: Output Stream.$ 

**6.17.3.8** void kfoundation::FileOutputStream::write ( PPtr < InputStream > is ) [virtual]

Writes the available contents from the given input stream to this output stream.

#### **Parameters**

is The stream to read from.

Implements kfoundation::OutputStream.

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

- · FileOutputStream.h
- FileOutputStream.cpp

# 6.18 kfoundation::IndexOutOfBoundException Class Reference

Thrown to signal access to a nonexisting element in an array.

#include <kfoundation/IndexOutOfBoundException.h>

#### **Public Member Functions**

IndexOutOfBoundException (string message)

Constructor.

#### **Additional Inherited Members**

# 6.18.1 Detailed Description

Thrown to signal access to a nonexisting element in an array.

### 6.18.2 Constructor & Destructor Documentation

6.18.2.1 kfoundation::IndexOutOfBoundException::IndexOutOfBoundException ( string message )

Constructor.

**Parameters** 

message A message describing the cause of the exception.

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

- · IndexOutOfBoundException.h
- IndexOutOfBoundException.cpp

# 6.19 kfoundation::InputStream Class Reference

Abstract interface for all input streams.

#include <kfoundation/InputStream.h>

### **Public Member Functions**

- virtual kf\_int32\_t read (kf\_octet\_t \*buffer, const kf\_int32\_t nOctets)=0
   Reads at most the given number of octets from the given buffer.
- virtual int read ()=0

Reads a single octet.

virtual int peek ()=0

Reads a single octet without advancing.

virtual kf\_int32\_t skip (const kf\_int32\_t nOctets)=0

Skips the at most the given number of octets without reading them.

virtual bool isEof ()=0

Checks if the end of stream is reached.

virtual bool isMarkSupported ()=0

If returns true, mark() and reset() methods can be used.

virtual void mark ()=0

Marks the current position of the stream.

virtual void reset ()=0

Returns the stream position to where mark() was used last time.

• virtual bool isBigEndian ()=0

Checks if the stream is big-endian.

### 6.19.1 Detailed Description

Abstract interface for all input streams.

#### 6.19.2 Member Function Documentation

```
6.19.2.1 virtual bool kfoundation::InputStream::isMarkSupported() [pure virtual]
```

If returns true, mark() and reset() methods can be used.

See Also

mark()

reset()

Implemented in kfoundation::InternetInputStream, kfoundation::BufferInputStream, kfoundation::FileInputStream, kfoundation::StringInputStream.

```
6.19.2.2 virtual void kfoundation::InputStream::mark() [pure virtual]
```

Marks the current position of the stream.

Use reset() to return to this position later and read the data again.

See Also

```
isMarkSupported()
reset()
```

Implemented in kfoundation::InternetInputStream, kfoundation::BufferInputStream, kfoundation::FileInputStream, kfoundation::StringInputStream.

```
6.19.2.3 virtual int kfoundation::InputStream::peek( ) [pure virtual]
```

Reads a single octet without advancing.

Returns

The value of read octet, or -1 if no data is available.

Implemented in kfoundation::InternetInputStream, kfoundation::BufferInputStream, kfoundation::FileInputStream, kfoundation::StringInputStream.

```
6.19.2.4 virtual kf_int32_t kfoundation::InputStream::read ( kf_octet_t *  buffer, const kf_int32_t nOctets ) [pure virtual]
```

Reads at most the given number of octets from the given buffer.

Returns the actual number of octets read.

**Parameters** 

buffer	The octets to read.
nOctets	Maximum number of octets to read.

#### Returns

The actual number of octets read.

Implemented in kfoundation::InternetInputStream, kfoundation::BufferInputStream, kfoundation::FileInputStream, kfoundation::StringInputStream.

```
6.19.2.5 virtual int kfoundation::InputStream::read() [pure virtual]
```

Reads a single octet.

#### Returns

The value of read octet, or -1 if no data is available.

Implemented in kfoundation::InternetInputStream, kfoundation::BufferInputStream, kfoundation::FileInputStream, kfoundation::StringInputStream.

```
6.19.2.6 virtual void kfoundation::InputStream::reset() [pure virtual]
```

Returns the stream position to where mark() was used last time.

If mark() is never called, it will reset to the begining of the stream.

#### See Also

```
isMarkSupported()
mark()
```

Implemented in kfoundation::InternetInputStream, kfoundation::BufferInputStream, kfoundation::FileInputStream, kfoundation::StringInputStream.

```
6.19.2.7 virtual kf int32 t kfoundation::InputStream::skip ( const kf int32 t nOctets ) [pure virtual]
```

Skips the at most the given number of octets without reading them.

If there are less number of octets in the stream, all available octets will be skipped.

#### Parameters

nOctets	The desired number of octets to skeep.
	The desired harmon or estate to encope

#### Returns

The actual number of octets skipped.

Implemented in kfoundation::InternetInputStream, kfoundation::BufferInputStream, kfoundation::FileInputStream, kfoundation::StringInputStream.

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

· InputStream.h

# 6.20 kfoundation::Int Class Reference

```
Wrapper for int type.
```

#include <kfoundation/Int.h>

#### **Public Member Functions**

• Int (int \_value)

Constructor.

• Int (const string &str)

Constructor.

• int get () const

Getter method.

• void set (const int value)

Setter method.

void printToStream (ostream &os) const

Implements compatibility with Streamer interface.

• string toString () const

Converts the result of invocation of printToStream(ostream&) to a std::string object.

#### **Static Public Member Functions**

• static int parse (const string &str)

Parses the given string to corresponding integer value.

• static string to HexString (const int v)

Returns the hexadecimal representation of the given value as a string.

• static string toString (const int v)

Converts the given integer value to string.

# 6.20.1 Detailed Description

Wrapper for int type.

### 6.20.2 Constructor & Destructor Documentation

6.20.2.1 kfoundation::Int::Int ( int value )

Constructor.

Sets the internal value to the given parameter.

6.20.2.2 kfoundation::Int::Int ( const string & str )

Constructor.

Parses the given string and sets the internal value accordingly.

#### 6.20.3 Member Function Documentation

```
6.20.3.1 int kfoundation::Int::get() const [inline]
```

Getter method.

Returns the internal value.

**6.20.3.2** int kfoundation::Int::parse (const string & str) [static]

Parses the given string to corresponding integer value.

**Parameters** 

```
str The string to be parsed.
```

#### Returns

The numeric value extracted from the given string.

```
6.20.3.3 void kfoundation::Int::set ( const int value ) [inline]
```

Setter method.

Sets the internal value to the given parameter.

```
6.20.3.4 string kfoundation::Int::toHexString (const int v) [static]
```

Returns the hexadecimal representation of the given value as a string.

**Parameters** 

V	The value to convert to hexadecimal.

# Returns

The hexadecimal representation of the given value.

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

- · Int.h
- Int.cpp

# 6.21 kfoundation::InternetAddress Class Reference

Encodes an IP address and port number.

#include <kfoundation/InternetAddress.h>

# **Public Types**

```
enum class_t {
    A, B, C, D,
    E }
```

Represents IPv4 address classes.

#### **Public Member Functions**

· InternetAddress ()

Constructs and address with value of 0.0.0.0:0.

InternetAddress (const kf\_octet\_t \*ip, kf\_int32\_t port)

Constructos a new instance with the given IP address and port number.

InternetAddress (const string &str)

Constructs a new instance parsing the address from the given string.

InternetAddress (const string &str, kf int32 t port)

Constructs a new instance parsing the address from the given string, and assigning port number from the given value.

InternetAddress (const InternetAddress &other)

Copy constructor.

const kf\_octet\_t \* getlp () const

Returns the IP address as an array of kf\_octet\_t.

kf\_int32\_t getPort () const

Returns the port number.

• InternetAddress copyWithPort (kf\_int32\_t port) const

Creates a new instance changing the port number to the one specified.

• bool equals (const InternetAddress &other) const

Checks if two instance of this class are equal.

class\_t getClass () const

Returns the class of this address.

InternetAddress getBroadcastAddress () const

Returns the address used to broad cast to the corresponding subnet.

InternetAddress getSubnetMask () const

Returns the corresponding subnet mask.

void printToStream (ostream &stream) const

Implements compatibility with Streamer interface.

### 6.21.1 Detailed Description

Encodes an IP address and port number.

This is an immutable object.

### 6.21.2 Member Enumeration Documentation

#### 6.21.2.1 enum kfoundation::InternetAddress::class t

Represents IPv4 address classes.

#### **Enumerator**

- A Class A.
- B Class B.

- C Class C.
- D Class D, reserved for multicast.
- *E* Class E, experimental, used for research.

### 6.21.3 Constructor & Destructor Documentation

6.21.3.1 kfoundation::InternetAddress::InternetAddress ( const kf\_octet\_t \* ip, kf\_int32\_t port )

Constructos a new instance with the given IP address and port number.

#### **Parameters**

ip	An array of kf_octet_t of size 4.
port	A port number.

### 6.21.3.2 kfoundation::InternetAddress::InternetAddress ( const string & str )

Constructs a new instance parsing the address from the given string.

#### **Parameters**

str	A string in X.Y.Z.T:P format.	':P' part is optional.	If not present, the port number will be
	assumed 0.		

#### 6.21.3.3 kfoundation::InternetAddress::InternetAddress ( const string & str, kf\_int32\_t port )

Constructs a new instance parsing the address from the given string, and assigning port number from the given value.

# Parameters

str	A string in X.Y.Z.T:P format. ':P' part is optional. It will be ignored even if specified.
port	A port number.

### 6.21.4 Member Function Documentation

### 6.21.4.1 InternetAddress kfoundation::InternetAddress::copyWithPort ( kf\_int32\_t port ) const

Creates a new instance changing the port number to the one specified.

### Parameters

port	The port number for the new object.

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

- · InternetAddress.h
- InternetAddress.cpp

# 6.22 kfoundation::InternetInputStream Class Reference

Input stream used to read from a TCP/IP port.

#include <kfoundation/InternetInputStream.h>

### **Public Member Functions**

• InternetInputStream ()

Constructor, creates a closed input stream.

∼InternetInputStream ()

Deconstructor.

const InternetAddress & getAddress () const

Returns the address that this stream is assigned to,.

void bind (const InternetAddress &address) throw (IOException)

Binds this stream to an Internet address.

• void unbind ()

Closes and unbinds this stream.

• bool isBound () const

Checks if this stream is bound to an address.

void listen ()

Blocks the current thread until an incomming connection is stablished.

• bool isOpen () const

Checks if the port is open.

• void close ()

Closes the stream if it is open.

• kf\_int32\_t getNReceivedOctets () const

Returns the number of octets received since the connection is stablished.

kf\_int32\_t read (kf\_octet\_t \*buffer, const kf\_int32\_t nBytes)

Reads at most the given number of octets from the given buffer.

• int read ()

Reads a single octet.

• int peek ()

Reads a single octet without advancing.

kf\_int32\_t skip (kf\_int32\_t bytes)

Skips the at most the given number of octets without reading them.

• bool isEof ()

Checks if the end of stream is reached.

• bool isMarkSupported ()

If returns true, mark() and reset() methods can be used.

· void mark ()

Marks the current position of the stream.

• void reset ()

Returns the stream position to where mark() was used last time.

• bool isBigEndian ()

Checks if the stream is big-endian.

• void serialize (PPtr< ObjectSerializer > serializer) const

Implements compatibility with SerializingStreamer interface.

# 6.22.1 Detailed Description

Input stream used to read from a TCP/IP port.

### 6.22.2 Constructor & Destructor Documentation

6.22.2.1 kfoundation::InternetInputStream::InternetInputStream ( )

Constructor, creates a closed input stream.

To use, first, bind this stream to an address using bind(const InternetAddress&) method, then invoke listen() method. listen() is a blocking method. It will be unblocked once a connection from a remote host is established. The following is the common pattern to use this class:

```
Ptr<InternetInputStream> iis = new InternetInputStream();
iis.bind(InternetAddress("1.2.3.4:5678"));
iis.listen();
while(!iis.isEof()) {
    iis.read();
}
```

#### 6.22.3 Member Function Documentation

6.22.3.1 void kfoundation::InternetInputStream::bind ( const InternetAddress & address ) throw IOException)

Binds this stream to an Internet address.

If the stream is already open and bound to another address, it will be closed and unbound. If the process does not have enough privilages to bind to the given address and port, or the port is already in use, this function will throw and exception.

**Parameters** 

address	The address to bind this stream to.
---------	-------------------------------------

#### **Exceptions**

```
Throws | IOException if binding fails.
```

# See Also

```
unbind() isBound()
```

6.22.3.2 bool kfoundation::InternetInputStream::isBound ( ) const

Checks if this stream is bound to an address.

```
See Also
```

```
bind()
unbind()
```

**6.22.3.3** bool kfoundation::InternetInputStream::isMarkSupported() [virtual]

If returns true, mark() and reset() methods can be used.

#### See Also

mark()
reset()

Implements kfoundation::InputStream.

6.22.3.4 void kfoundation::InternetInputStream::listen ( )

Blocks the current thread until an incomming connection is stablished.

#### **Exceptions**

Throws	IOException if listening could not be initiated.
--------	--

```
6.22.3.5 void kfoundation::InternetInputStream::mark() [virtual]
```

Marks the current position of the stream.

Use reset() to return to this position later and read the data again.

# See Also

```
isMarkSupported()
reset()
```

Implements kfoundation::InputStream.

```
6.22.3.6 int kfoundation::InternetInputStream::peek() [virtual]
```

Reads a single octet without advancing.

#### Returns

The value of read octet, or -1 if no data is available.

Implements kfoundation::InputStream.

```
6.22.3.7 kf_int32_t kfoundation::InternetInputStream::read ( kf_octet_t * buffer, const kf_int32_t nOctets )

[virtual]
```

Reads at most the given number of octets from the given buffer.

Returns the actual number of octets read.

#### **Parameters**

buffer	The octets to read.
nOctets	Maximum number of octets to read.

# Returns

The actual number of octets read.

Implements kfoundation::InputStream.

```
6.22.3.8 int kfoundation::InternetInputStream::read() [virtual]
```

Reads a single octet.

#### Returns

The value of read octet, or -1 if no data is available.

Implements kfoundation::InputStream.

```
6.22.3.9 void kfoundation::InternetInputStream::reset() [virtual]
```

Returns the stream position to where mark() was used last time.

If mark() is never called, it will reset to the begining of the stream.

#### See Also

```
isMarkSupported()
mark()
```

Implements kfoundation::InputStream.

```
6.22.3.10 kf_int32_t kfoundation::InternetInputStream::skip(kf_int32_t nOctets) [virtual]
```

Skips the at most the given number of octets without reading them.

If there are less number of octets in the stream, all available octets will be skipped.

#### **Parameters**

```
nOctets The desired number of octets to skeep.
```

#### Returns

The actual number of octets skipped.

Implements kfoundation::InputStream.

6.22.3.11 void kfoundation::InternetInputStream::unbind ( )

Closes and unbinds this stream.

#### See Also

bind()
isBound()

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

- InternetInputStream.h
- · InternetInputStream.cpp

# 6.23 kfoundation::InternetOutputStream Class Reference

Input stream used to write to TCP/IP socket.

```
#include <kfoundation/InternetOutputStream.h>
```

#### **Public Member Functions**

InternetOutputStream (const InternetAddress &address)

Constructor, the object will dedicated to read from the given address.

∼InternetOutputStream ()

Deconstructor.

· const InternetAddress & getAddress () const

Returns the address this stream is assigend to.

void connect () throw (IOException)

Connects to the given remote address.

• bool isOpen () const

Checks if the connection is open.

• kf\_int32\_t getNSentOctets () const

Get the number of octets written since the connection is established.

• bool isBigEndian () const

Checks if the stream is big-endian.

• void write (const kf\_octet\_t \*buffer, const kf\_int32\_t nBytes)

Writes the given number of octets of the given buffer to the stream.

void write (kf\_octet\_t byte)

Writes a single octet to the stream.

void write (PPtr< InputStream > os)

Writes the available contents from the given input stream to this output stream.

· void close ()

Closes the stream.

void serialize (PPtr< ObjectSerializer > serializer) const

Implements compatibility with SerializingStreamer interface.

# 6.23.1 Detailed Description

Input stream used to write to TCP/IP socket.

#### 6.23.2 Constructor & Destructor Documentation

6.23.2.1 kfoundation::InternetOutputStream::InternetOutputStream ( const InternetAddress & address )

Constructor, the object will dedicated to read from the given address.

To use, invoke connect() first.

#### 6.23.3 Member Function Documentation

**6.23.3.1** void kfoundation::InternetOutputStream::close() [virtual]

Closes the stream.

It will no longer be readable.

Implements kfoundation::OutputStream.

6.23.3.2 void kfoundation::InternetOutputStream::connect ( ) throw IOException)

Connects to the given remote address.

Blocks the current thread until the connection is established.

**Exceptions** 

Generated on Wed Aug 26 2015 16:32:37 for KFoundation by Doxygen

Throws | IOException if the connection could not be established.

6.23.3.3 void kfoundation::InternetOutputStream::write ( const kf\_octet\_t \* buffer, const kf\_int32\_t nOctets )

[virtual]

Writes the given number of octets of the given buffer to the stream.

#### **Parameters**

buffer	The octets to write.
nOctets	Number of octets to write.

Implements kfoundation::OutputStream.

**6.23.3.4** void kfoundation::InternetOutputStream::write ( kf\_octet\_t octet ) [virtual]

Writes a single octet to the stream.

#### **Parameters**

octet	The octet to write

Implements kfoundation::OutputStream.

**6.23.3.5** void kfoundation::InternetOutputStream::write ( PPtr < InputStream > is ) [virtual]

Writes the available contents from the given input stream to this output stream.

#### **Parameters**

is The stream to read from.
-----------------------------

Implements kfoundation::OutputStream.

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

- · InternetOutputStream.h
- · InternetOutputStream.cpp

# 6.24 kfoundation::InvalidFormatException Class Reference

Thrown when an input with an invalid format is encountered.

#include <kfoundation/InvalidFormatException.h>

#### **Public Member Functions**

• InvalidFormatException (string message)

Constrcutor.

### **Additional Inherited Members**

## 6.24.1 Detailed Description

Thrown when an input with an invalid format is encountered.

# 6.24.2 Constructor & Destructor Documentation

6.24.2.1 kfoundation::InvalidFormatException::InvalidFormatException ( string message )

Constrcutor.

**Parameters** 

message | A message describing the error.

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

- · InvalidFormatException.h
- InvalidFormatException.cpp

# 6.25 kfoundation::InvalidPointerException Class Reference

Thrown on attempt to access an invalid pointer.

#include <kfoundation/InvalidPointerException.h>

#### **Public Member Functions**

• InvalidPointerException (string message)

Constructor.

#### **Additional Inherited Members**

# 6.25.1 Detailed Description

Thrown on attempt to access an invalid pointer.

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

- · InvalidPointerException.h
- · InvalidPointerException.cpp

# 6.26 kfoundation::IOException Class Reference

Thrown to signal an IO-related exception.

```
#include <kfoundation/IOException.h>
```

#### **Public Member Functions**

IOException (const string &message)

Constructor.

### **Additional Inherited Members**

# 6.26.1 Detailed Description

Thrown to signal an IO-related exception.

#### 6.26.2 Constructor & Destructor Documentation

6.26.2.1 kfoundation::IOException::IOException ( const string & message )

Constructor.

**Parameters** 

```
message A message explaining the cause of exception.
```

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

- · IOException.h
- IOException.cpp

# 6.27 kfoundation::KFException Class Reference

Superclass for all exceptions in KFoundation.

```
#include <kfoundation/KFException.h>
```

# **Public Member Functions**

• KFException (string message)

Primary constructor.

KFException (const KFException & other)

Copy constructor.

∼KFException () throw ()

Deconstructor.

- const string & getMessage () const
- void serialize (PPtr< ObjectSerializer > os) const

Serializing method.

• virtual const char \* what () const throw ()

Overriding the standard C++ behaviour, returns the message and stack trace in C-string format.

### **Protected Member Functions**

• void setName (string name)

This method should be called in the constructor of every subclass.

### 6.27.1 Detailed Description

Superclass for all exceptions in KFoundation.

It maintain a record of stack trace at the time it is thrown. Since this class is a SerializingStreamer, it can be fed directly to logger.

```
try {
    ... something ...
} catch(KFException& e) {
  LOG << e << EL;
}</pre>
```

#### 6.27.2 Constructor & Destructor Documentation

6.27.2.1 kfoundation::KFException::KFException ( string message )

Primary constructor.

Once invoked, creates and stores the current stack trace.

**Parameters** 

message The error message describing the exception.

6.27.2.2 kfoundation::KFException::KFException ( const KFException & other )

Copy constructor.

**Parameters** 

other The object to be copied.

#### 6.27.3 Member Function Documentation

6.27.3.1 const string & kfoundation::KFException::getMessage ( ) const

Returns

The message assigned to this exception.

**6.27.3.2** void kfoundation::KFException::serialize ( PPtr < ObjectSerializer > builder ) const [virtual]

Serializing method.

Parameters

builder The ObjectSerializer used to build the output.

Implements kfoundation::SerializingStreamer.

**6.27.3.3 void kfoundation::KFException::setName ( string**  *name* **)** [protected]

This method should be called in the constructor of every subclass.

It sets the name of the exception which would normally be the same as the name of the class. At the same time, it keeps track of the number of exception class constructors of superclasses called within each other, in order to remove them from the printed stack trace.

**Parameters** 

name The name of the exception class.

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

- · KFException.h
- KFException.cpp

# 6.28 kfoundation::Logger Class Reference

Multi-channel, muti-level logger utility.

```
#include <kfoundation/Logger.h>
```

#### **Classes**

· class Channel

Logger channel.

· class Stream

Log stream.

# **Public Types**

```
    enum level_t {
        ERR = 0, WRN = 1, L1 = 2, L2 = 3,
        L3 = 4 }
        Log level.
```

#### **Public Member Functions**

· Logger ()

Default constructor.

∼Logger ()

Deconstructor.

Channel & addChannel (const string &name, ostream \*os)

Adds a channel that outputs to the given ostream object.

Channel & addChannel (const string &name, const string &fileName)

Adds a channel that outputs to the given file.

Channel & getChannelByName (const string &name) const

Returns refernce to the channel with the given name.

• void removeAllChannels ()

Removes all channels.

Stream & log (level\_t level, const char fileName[], int lineNumebr, const char functionName[])

Not for direct use.

Stream & log (level\_t level)

Creates and initiates a new Logger:Stream with the given level.

void setLevel (level\_t level)

Set the filtering level of all channels to the given value.

• void mute ()

Sets all channels to silent.

• void unmute ()

Removes silence flag from all channels.

# 6.28.1 Detailed Description

Multi-channel, muti-level logger utility.

Normally, you want to use the default logger already provided via System::getLogger(). LOG and LOG\_XXX macros expand to the default logger. So,

```
{	t LOG} << {	t "Hello"} << {	t EL;}
```

is equivalant to

```
System::getLogger().log(Logger::L3) << "Hello" << EL;</pre>
```

Calling log() method starts a log stream. The stream should end with EL in order to be flushed into the designated channels. Inbetween, any of the following types can be used:

- · string
- · char\*
- wchar t\*
- bool
- · char
- int
- · long int
- · long long int
- · float
- double
- · long double
- · Streamer and any of its subclasses.

The default logger has one channel that outputs to standard error console. However it is possible add and remove channels manually via removeAllChannels() and addChannel() methods.

Log level determines the importance of the message being logged. Ordered by importance from lowest to highest, these levels are

- L3
- L2
- L1
- WRN
- ERR

A channel will filter any message with a level lower than its designated level. For example a channel designated with WRN level, will pass WRN and ERR but filters L3, L2, and L1.

It is possible to reduce the amount of output produced by logger by setting its level or using mute() method or Logger::Channel::setSilent().

Another way that helps the performance of your released program is to use DLOG\_XXX macros. Usage:

```
DLOG_L3("myCounter: " << myCounter);</pre>
```

And to disable logging define DLOG\_L3 as nothing in the beginning of your file. For example:

```
#ifndef DEBUG
#undef DLOG_L3
#define DLOG_L3
#endif
```

This will remove the log part all together from your compiled code.

To customize the header preceeding each log, use Channel::setFormat() method.

# 6.28.2 Member Enumeration Documentation

#### 6.28.2.1 enum kfoundation::Logger::level\_t

Log level.

#### **Enumerator**

**ERR** Error severity level.

WRN Warning severity level.

- L1 Severity lower than WRN but higher than L2.
- L2 Severity lower than L1 but higher than L3.
- L3 The lowest severity level.

#### 6.28.3 Member Function Documentation

### 6.28.3.1 Logger::Channel & kfoundation::Logger::addChannel ( const string & name, ostream \* os )

Adds a channel that outputs to the given ostream object.

Since ostream is not a ManagedObejct it needs to be deleted by user if necessary.

#### **Parameters**

name	A name for the new channel.
os	The output stream to print to.

### 6.28.3.2 Logger::Channel & kfoundation::Logger::addChannel ( const string & name, const string & fileName )

Adds a channel that outputs to the given file.

The new logs will be appended to the existing contents of the file.

#### **Parameters**

name	A name for the new channel.
fileName	Path to the file to write to.

### 6.28.3.3 Logger::Channel & kfoundation::Logger::getChannelByName ( const string & name ) const

Returns refernce to the channel with the given name.

### **Exceptions**

An	exception if there is no channel with the given name.
----	---

# 6.28.3.4 Logger::Stream & kfoundation::Logger::log ( level\_t level, const char fileName[], int lineNumber, const char functionName[] )

Not for direct use.

Used by LOG, LOG\_XXX and DLOG\_XXX macros to create a new Logger::Stream.

### 6.28.3.5 Logger::Stream & kfoundation::Logger::log ( level\_t level )

Creates and initiates a new Logger:Stream with the given level.

#### **Parameters**

level	Level of the log item.
-------	------------------------

#### Returns

Reference to a new Logger::Stream.

6.28.3.6 void kfoundation::Logger::setLevel ( level\_t level )

Set the filtering level of all channels to the given value.

#### **Parameters**

```
The level to set.
```

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

- · Logger.h
- · Logger.cpp

# 6.29 kfoundation::LongInt Class Reference

Wrapper class for 'long int' type.

#include <kfoundation/LongInt.h>

# **Public Types**

enum encoding\_t { DECIMAL, HEXADECIMAL }
 Numeral Base.

### **Public Member Functions**

· LongInt (const long int value)

Constructor.

• LongInt (const string &str, const encoding\_t &encoding=DECIMAL)

Constructor.

· long int get () const

Getter method.

void set (const long int v)

Setter method.

void printToStream (ostream &os) const

Implements compatibility with Streamer interface.

#### **Static Public Member Functions**

static long int parse (const string &str, const encoding\_t encoding=DECIMAL)

Parses the given string according the given encoding and returns the extracted value.

static string toHexString (const long int v)

Converts the given value to hexadecimal string representation.

• static string to HexString (void \*ptr)

Converts the value of the given pointer (not the value it is pointing at) to hexadecimal string.

static string toString (const long int v)

Converts the given value to string.

### 6.29.1 Detailed Description

Wrapper class for 'long int' type.

#### 6.29.2 Member Enumeration Documentation

6.29.2.1 enum kfoundation::LongInt::encoding\_t

Numeral Base.

**Enumerator** 

DECIMAL Base 10.

HEXADECIMAL Base 16.

#### 6.29.3 Constructor & Destructor Documentation

6.29.3.1 kfoundation::LongInt::LongInt ( const long int value )

Constructor.

Assigns the internal value to the given paramaeter.

6.29.3.2 kfoundation::LongInt::LongInt ( const string & str, const encoding\_t & encoding = DECIMAL )

Constructor.

Parses the given string according to the given encoding and sets the internal value accordingly.

**Parameters** 

str	The string to parse.
encoding	The encoding of the given string. Default value is DECIMAL.

#### 6.29.4 Member Function Documentation

**6.29.4.1** long int kfoundation::LongInt::get ( ) const [inline]

Getter method.

Returns the internal value.

6.29.4.2 long int kfoundation::LongInt::parse ( const string & str, const encoding\_t encoding = DECIMAL ) [static]

Parses the given string according the given encoding and returns the extracted value.

**Parameters** 

str	The string to parse
encoding	The encoding of the given string. Default value is DECIMAL.

6.29.4.3 void kfoundation::LongInt::set ( const long int v ) [inline]

Setter method.

Sets the internal value to the given argument.

**Parameters** 

V	The value to be set.

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

- · LongInt.h
- · LongInt.cpp

# 6.30 kfoundation::ManagedArray< T > Class Template Reference

One-dimentional indexed collection of ManagedObjects.

```
#include <kfoundation/ManagedArray.h>
```

#### **Public Member Functions**

ManagedArray (kf\_int32\_t initialCapacity)

Constructor, creates an empty new array with the given initial capacity.

ManagedArray ()

Default constructor, creates an empty new array with default initial capacity.

∼ManagedArray ()

Deconstructor.

void remove (const kf\_int32\_t index)

Removes and releases the element at the given index.

void push (PPtr< T > value)

Adds the given pointer to the end of the array and retains it.

• Ptr< T > pop ()

Returns the pointer at highest index of the array, and decreases its size by one.

void insert (const kf\_int32\_t index, Ptr< T > value)

Inserts the given pointer to at the given index.

• void clear ()

Resets the size of the array to zero and releases all existing elements.

• bool isEmpty () const

Checks if the array is empty.

void setSize (kf\_int32\_t size)

Sets the size of the array as specified.

kf\_int32\_t getSize () const

Returns the size of the array.

Ptr< T > & at (const kf\_int32\_t index)

Returns reference to the pointer at the given index of the array.

-  $kf_{int32_t indexOf}$  (PPtr< T > value) const

Searches for the first occurance of the given pointer in the array.

kf\_int32\_t indexOf (kf\_int32\_t offset, PPtr< T > value) const

Searches for the occurance of the first occurance of the given pointer after the given offset in the array.

virtual void serialize (PPtr< ObjectSerializer > builder) const

Implements compatibility with SerializingStreamer interface.

#### Static Public Attributes

static const kf\_int32\_t NOT\_FOUND = -1

Flag returned by search methods when the desired item is not found.

### 6.30.1 Detailed Description

template<typename T>class kfoundation::ManagedArray<T>

One-dimentional indexed collection of ManagedObjects.

This class only maintains pointers to the given objects. In contrast the Array class maintains actual values. Items will be retained once added and released once removed. All objects will be released open deconstruction of this class.

See Also

Array

#### 6.30.2 Constructor & Destructor Documentation

6.30.2.1 template < typename T > kfoundation::ManagedArray < T >::ManagedArray ( kf\_int32\_t initialCapacity )

Constructor, creates an empty new array with the given initial capacity.

The capacity will grow exponentially as needed.

**Parameters** 

initialCapacity Initial capacity.

6.30.2.2 template<typename T > kfoundation::ManagedArray< T >::~ManagedArray ( )

Deconstructor.

All elements will be released upon deconstruction.

# 6.30.3 Member Function Documentation

6.30.3.1 template  $T > Ptr < T > & kfoundation::ManagedArray < T >::at ( const kf_int32_t index ) 
[inline]$ 

Returns reference to the pointer at the given index of the array.

**Parameters** 

index The index of the element to be accessed.

#### **Exceptions**

Throws	IndexOutOfBoundException if the requested index is bigger or equal the size of
	the array.

6.30.3.2 template < typename T > kf\_int32\_t kfoundation::ManagedArray < T >::indexOf ( PPtr < T > value ) const

Searches for the first occurance of the given pointer in the array.

Note

This method only compares pointers not the pointed values.

#### **Parameters**

_		
	value	The pointer to search for.

#### Returns

The index of the first occurance of the given pointer, or NOT\_FOUND.

6.30.3.3 template<typename T > kf\_int32\_t kfoundation::ManagedArray< T >::indexOf ( kf\_int32\_t offset, PPtr< T > value ) const

Searches for the occurance of the first occurance of the given pointer after the given offset in the array.

Note

This method only compares pointers not the pointed values.

#### **Parameters**

value	The pointer to search for.

#### Returns

The index of the desired element, or NOT FOUND.

6.30.3.4 template < typename T > void kfoundation::ManagedArray < T >::insert ( const kf\_int32\_t index, Ptr < T > value )

Inserts the given pointer to at the given index.

Previous values at the given index and above will be shifted to higher indexes.

#### **Parameters**

index	The index to be inserted at.
value	The pointer to be inserted.

6.30.3.5 template < typename T > Ptr< T > kfoundation::ManagedArray < T >::pop ( )

Returns the pointer at highest index of the array, and decreases its size by one.

The pointer will not be released internally, it will be released by the Ptr object returned.

Returns

The popped pointer.

#### **Exceptions**

Throws	IndexOutOfBoundException if the array is empty.
--------	---

6.30.3.6 template < typename T > void kfoundation::ManagedArray < T >::push ( PPtr < T > value )

Adds the given pointer to the end of the array and retains it.

#### **Parameters**

value The pointer to be pushed.
---------------------------------

6.30.3.7 template < typename T > void kfoundation::ManagedArray < T >::remove ( const kf\_int32\_t index )

Removes and releases the element at the given index.

#### **Parameters**

index	The index of the element to be removed.

6.30.3.8 template < typename T > void kfoundation::ManagedArray < T >::setSize ( kf\_int32\_t size )

Sets the size of the array as specified.

If the new size is larger than previous one the added elements are set to NULL. If the new size is smaller the removed elements are released. If the request size is larger than the capacity, the capacity will grow as needed.

### **Parameters**

size	The array's new size.
------	-----------------------

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

- · ManagedArrayDecl.h
- ManagedArray.h

# 6.31 kfoundation::ManagedObject Class Reference

The root class for all classes using KFoundation framework.

#include <kfoundation/ManagedObject.h>

### **Public Member Functions**

• ManagedObject ()

Default constructor.

virtual ∼ManagedObject ()

Deconstructor.

PPtr< ManagedObject > getPtr () const

Returns a managed poitner to this object.

# 6.31.1 Detailed Description

The root class for all classes using KFoundation framework.

Only subclasses of ManagedObject can be accessed via managed pointers.

#### 6.31.2 Constructor & Destructor Documentation

```
6.31.2.1 kfoundation::ManagedObject::~ManagedObject() [virtual]
```

Deconstructor.

Upon distruction makes sure this object is undregistered from its corresponding memory manager.

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

- · ManagedObject.h
- · ManagedObject.cpp

# 6.32 kfoundation::MasterMemoryManager Class Reference

Manages all the memory managers used in a process.

```
#include <kfoundation/MasterMemoryManager.h>
```

#### **Public Member Functions**

• MasterMemoryManager ()

Default constructor.

•  $\sim$ MasterMemoryManager ()

Deconstructor.

• const ObjectRecord & registerObject (ManagedObject \*ptr)

Registeres a new object to the default manager.

• int registerManager (MemoryManager \*manager)

Registers a new memory manager and assignes it with a unique ID.

• void unregisterManager (int index)

Unregisters a memory manager given its index.

• kf\_octet\_t getNManagers ()

Returns the number of registered managers.

MemoryManager \* getManagerAtIndex (int index)

Returns the manager at the given index.

• void updataTable (int index)

This should be called whenever a manager reallocates its table.

• void dump () const

Prints a list of all managed objects.

· void printStats () const

Prints a report of memory usage.

### 6.32.1 Detailed Description

Manages all the memory managers used in a process.

Only once instance of this class exists per process. This instance can be obtained via System::getMasterMemory-Manager().

This object always owns an instance of RefCountMemoryManager as its default manager. To access use get-ManagerAtIndex(0).

### 6.32.2 Member Function Documentation

6.32.2.1 int kfoundation::MasterMemoryManager::registerManager ( MemoryManager \* manager )

Registers a new memory manager and assignes it with a unique ID.

#### **Parameters**

manager	The manager to be registered.	

#### Returns

The ID assigned to the given manager.

6.32.2.2 void kfoundation::MasterMemoryManager::unregisterManager ( int index )

Unregisters a memory manager given its index.

**Parameters** 

index	The index of the memory manager to be unregistered.

6.32.2.3 void kfoundation::MasterMemoryManager::updataTable ( int index )

This should be called whenever a manager reallocates its table.

MasterMemoryManager keeps a separate record of the all memory manager tables. If its location is changed for any reason, this function should be called to so that master updates its internal reference.

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

- · MasterMemoryManager.h
- · MasterMemoryManager.cpp

# 6.33 kfoundation::MemoryException Class Reference

Used to throw exeptions related to memory.

#include <kfoundation/MemoryException.h>

#### **Public Member Functions**

MemoryException (string message)

Constructor.

#### **Additional Inherited Members**

### 6.33.1 Detailed Description

Used to throw exeptions related to memory.

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

- · MemoryException.h
- · MemoryException.cpp

# 6.34 kfoundation::MemoryManager Class Reference

Abstract interface to be implemented by all memory managers.

```
#include <kfoundation/MemoryManager.h>
```

#### **Public Member Functions**

virtual ∼MemoryManager ()

Deconstructor.

• virtual const ObjectRecord & registerObject (ManagedObject \*obj)=0

Creates a new entry for the object at the given memory location.

• virtual void retain (kf\_int32\_t index, kf\_int16\_t key)=0

Retains the object associated with the given index if key maches, otherwise throws InvalidPointerException.

• virtual void release (kf\_int32\_t index, kf\_int16\_t key)=0

Releases the object associated with the given index if the key matches, otherwise throws InvalidPointerException.

virtual void remove (kf\_int32\_t index, kf\_int16\_t key)=0

Unmanages the object at the given index if the key matches, otherwise, throws InvalidPointerException.

virtual ObjectRecord \* getTable ()=0

Returns the memory locatoin of the begining of this manager's object table.

virtual kf int32 t getTableSize () const =0

Returns the number of available table records.

# 6.34.1 Detailed Description

Abstract interface to be implemented by all memory managers.

#### 6.34.2 Member Function Documentation

```
6.34.2.1 kfoundation::MemoryManager::getTable() [pure virtual]
```

Returns the memory locatoin of the begining of this manager's object table.

The object table is a one dimentional array of ObjectRecord.

See Also

getTableSize()

Implemented in kfoundation::ObjectPoolMemoryManager< T >, and kfoundation::RefCountMemoryManager.

```
6.34.2.2 kfoundation::MemoryManager::getTableSize()const [pure virtual]
```

Returns the number of available table records.

Since some records can be unused, this is usually not equal but larger than the the number of objects managed by this manager.

See Also

getTable()

Implemented in kfoundation::ObjectPoolMemoryManager< T >, and kfoundation::RefCountMemoryManager.

```
6.34.2.3 kfoundation::MemoryManager::release ( kf_int32_t index, kf_int16_t key ) [pure virtual]
```

Releases the object associated with the given index if the key matches, otherwise throws InvalidPointerException.

If the retain count drops to zero the object will be deleted.

Implemented in kfoundation::ObjectPoolMemoryManager< T >, and kfoundation::RefCountMemoryManager.

```
6.34.2.4 kfoundation::MemoryManager::remove(kf_int32_t index, kf_int16_t key) [pure virtual]
```

Unmanages the object at the given index if the key matches, otherwise, throws InvalidPointerException.

The index will be reused in the future.

Implemented in kfoundation::ObjectPoolMemoryManager< T >, and kfoundation::RefCountMemoryManager.

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

- · MemoryManager.h
- · MemoryManager.cpp

### 6.35 kfoundation::Mutex Class Reference

Mutex, used to prevent multiple threads to enter a critical region.

```
#include <kfoundation/Mutex.h>
```

#### **Public Member Functions**

Mutex (bool isShared=false)

Constructor.

• ∼Mutex ()

Deconstructor.

• void lock ()

Locks the mutex.

void unlock ()

Unlocks the mutex.

# 6.35.1 Detailed Description

Mutex, used to prevent multiple threads to enter a critical region.

# 6.35.2 Constructor & Destructor Documentation

**6.35.2.1 kfoundation::Mutex::Mutex ( bool** *isShared* = false )

Constructor.

#### **Parameters**

isShared	Should be set true if the mutex is shared between multiple dynamic libraries (shared objects).	]
	Default value is false.	

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

- · Mutex.h
- · Mutex.cpp

# 6.36 kfoundation::NullPointerException Class Reference

Thrown on attempt to access to a null pointer.

#include <kfoundation/NullPointerException.h>

#### **Public Member Functions**

• NullPointerException ()

Default constructor.

NullPointerException (string message)

Message constructor.

#### **Additional Inherited Members**

#### 6.36.1 Detailed Description

Thrown on attempt to access to a null pointer.

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

- · NullPointerException.h
- NullPointerException.cpp

# 6.37 kfoundation::NumericVector < T > Class Template Reference

A subclass of Array, adds numeric operations.

#include <kfounadtion/NumericVector.h>

#### **Public Member Functions**

NumericVector ()

Default constructor, creates an empty array.

• NumericVector (T \*values, kf\_int32\_t size)

Constructor, creats a new Numeric Vector copying the specified elements from a C-style array.

Ptr< NumericVector< T >> negate () const

Returns the pointer to a new NumericVector whos elements are the negative of their corresponding elements in this object.

• Ptr< NumericVector< T >> add (const Ptr< NumericVector< T >> &other) const

Returns the pointer to a new Numeric Vector representing the summation of this vector with the given one.

Ptr< NumericVector< T >> sub (const Ptr< NumericVector< T >> &other) const

Returns the pointer to a new NumericVector representing the substaction of the given vector from this one.

Ptr< NumericVector< T >> mul (const T &coef) const

Resturns the pointer to a new Numeric Vector resulted from multiplying this vector to the given scalar value.

void printToStream (ostream &os) const

Implements compatibility with Streamer interface.

#### **Static Public Member Functions**

static Ptr< NumericVector< T >> parseInt (const string &str)

Returns the pointer to a new NumericVector parsing the given string.

#### **Additional Inherited Members**

#### 6.37.1 Detailed Description

template<typename T>class kfoundation::NumericVector< T>

A subclass of Array, adds numeric operations.

Can also convert the contents to and from string.

#### 6.37.2 Constructor & Destructor Documentation

6.37.2.1 template<typename T > kfoundation::NumericVector< T >::NumericVector ( T \* values, kf\_int32\_t size )

Constructor, creats a new NumericVector copying the specified elements from a C-style array.

#### **Parameters**

values	The begining of the C-style array containing values to be copied.
size	The number of elements to be copied.

#### 6.37.3 Member Function Documentation

 $6.37.3.1 \quad template < typename \ T > Ptr < \ Numeric Vector < T > > k foundation:: Numeric Vector < T > :: add ( \ const \ Ptr < \ Numeric Vector < T > > & \textit{other} \ ) \ const$ 

Returns the pointer to a new Numeric Vector representing the summation of this vector with the given one.

#### **Parameters**

other	The vector to add this one to.

6.37.3.2 template < typename T > Ptr < Numeric Vector < T > > kfoundation::Numeric Vector < T >::mul ( const T & coef ) const

Resturns the pointer to a new NumericVector resulted from multiplying this vector to the given scalar value.

### Parameters

coef	The scalar value to multiply this vector to.

6.37.3.3 template < typename T > Ptr < Numeric Vector < T >> k foundation::Numeric Vector < T >::parseInt (const string & str) [static]

Returns the pointer to a new NumericVector parsing the given string.

A valid input string should have a format like "{1.1, 2.3, 4.2, 3}".

**Parameters** 

str | The string to parse.

 $6.37.3.4 \quad template < typename \ T > Ptr < \ Numeric \ Vector < T > > k foundation:: Numeric \ Vector < T > :: sub ( const Ptr < \ Numeric \ Vector < T > > & \textit{other} \ ) const$ 

Returns the pointer to a new NumericVector representing the substaction of the given vector from this one.

Parameters 4 8 1

other The vector to substract from this one.

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

- · NumericVectorDecl.h
- · NumericVector.h

# 6.38 kfoundation::ObjectDumpBuilderException Class Reference

Thrown when ObjectSerializer is used in an invalid way.

#include <kfoundation/ObjectSerializer.h>

#### **Public Member Functions**

ObjectDumpBuilderException (string message)
 Constructor.

#### **Additional Inherited Members**

# 6.38.1 Detailed Description

Thrown when ObjectSerializer is used in an invalid way.

#### 6.38.2 Constructor & Destructor Documentation

6.38.2.1 kfoundation::ObjectDumpBuilderException::ObjectDumpBuilderException ( string message )

Constructor.

**Parameters** 

message A message describing the problem.

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

- · ObjectSerializer.h
- ObjectSerializer.cpp

# 6.39 kfoundation::ObjectPoolMemoryManager< T > Class Template Reference

Reuses the objects in a preallocated pool whenever a new instance is needed.

#include <kfoundation/ObjectPoolMemoryManager.h>

#### **Public Member Functions**

ObjectPoolMemoryManager (const int initialCapacity, const int growthRate)

Constructor.

∼ObjectPoolMemoryManager ()

Deconstructor.

• Ptr< T > get ()

Returns an unused objects in the pool with reference count of 1.

const ObjectRecord & registerObject (ManagedObject \*obj)

Creates a new entry for the object at the given memory location.

void retain (kf\_int32\_t index, kf\_int16\_t key)

Retains the object associated with the given index if key maches, otherwise throws InvalidPointerException.

• void release (kf\_int32\_t index, kf\_int16\_t key)

Releases the object associated with the given index if the key matches, otherwise throws InvalidPointerException.

void remove (kf\_int32\_t index, kf\_int16\_t key)

Unmanages the object at the given index if the key matches, otherwise, throws InvalidPointerException.

ObjectRecord \* getTable ()

Returns the memory locatoin of the begining of this manager's object table.

kf\_int32\_t getTableSize () const

Returns the number of available table records.

void serialize (PPtr< ObjectSerializer > seralizer) const

Serializer.

# 6.39.1 Detailed Description

template<typename T>class kfoundation::ObjectPoolMemoryManager< T>

Reuses the objects in a preallocated pool whenever a new instance is needed.

When an object is no longer needed, it will not be deleted, instead, it's PoolObject::finalize() method will be called to clean it up for next use. When pool is full and a new instance is needed, the pool size will be automatically increased.

Call get() method to obtain a clean instance to use.

#### 6.39.2 Constructor & Destructor Documentation

6.39.2.1 template < typename T > kfoundation::ObjectPoolMemoryManager < T >::ObjectPoolMemoryManager ( const int initialCapacity, const int growthRate )

Constructor.

**Parameters** 

initialCapacity	Initial capacity.
growthRate	The capacity will be multiplied by this value every time more objects than capacity is needed.

6.39.2.2 template < typename T > kfoundation::ObjectPoolMemoryManager < T >::  $\sim$  ObjectPoolMemoryManager ( )

Deconstructor.

Deconstructs all objects in the pool internally.

#### 6.39.3 Member Function Documentation

```
 \textbf{6.39.3.1} \quad \textbf{template} < \textbf{typename T} > \textbf{ObjectRecord} * \textbf{kfoundation::ObjectPoolMemoryManager} < \textbf{T} > :: \texttt{getTable ( )} \\ [\texttt{virtual}]
```

Returns the memory locatoin of the begining of this manager's object table.

The object table is a one dimentional array of ObjectRecord.

See Also

```
getTableSize()
```

Implements kfoundation::MemoryManager.

Returns the number of available table records.

Since some records can be unused, this is usually not equal but larger than the the number of objects managed by this manager.

See Also

```
getTable()
```

Implements kfoundation::MemoryManager.

```
6.39.3.3 template < typename T > void kfoundation::ObjectPoolMemoryManager < T >::release ( kf_int32_t index, kf int16 t key ) [virtual]
```

Releases the object associated with the given index if the key matches, otherwise throws InvalidPointerException.

If the retain count drops to zero the object will be deleted.

Implements kfoundation::MemoryManager.

```
6.39.3.4 template < typename T > void kfoundation::ObjectPoolMemoryManager < T >::remove ( kf_int32_t index, kf_int16_t key ) [virtual]
```

Unmanages the object at the given index if the key matches, otherwise, throws InvalidPointerException.

The index will be reused in the future.

Implements kfoundation::MemoryManager.

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

- · ObjectPoolMemoryManagerDecl.h
- · ObjectPoolMemoryManager.h

# 6.40 kfoundation::ObjectRecord Struct Reference

Structure of memory manager's table records.

```
#include <kfoundation/MemoryManager.h>
```

#### **Public Attributes**

ManagedObject \* ptr

Memory location of the target object.

• kf\_int16\_t retainCount

Retain count.

· kf\_int16\_t key

Key.

kf\_int8\_t manager

The ID of the manager owning this table.

kf\_int16\_t index

Index of this record.

• kf\_int32\_t serialNumber

Unique serial number for this object.

· bool isStatic

'true' if the object is static

· bool isBeingDeleted

Flag for internal use.

### 6.40.1 Detailed Description

Structure of memory manager's table records.

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

· MemoryManager.h

# 6.41 kfoundation::ObjectSerializer Class Reference

Provides APIs to serialize an object.

```
#include <kfoundation/ObjectSerializer.h>
```

# **Public Types**

enum output\_type\_t { DUMP, XML, JSON }

Output format.

#### **Public Member Functions**

ObjectSerializer (ostream &stream, output type t outputType, int indentUnit)

Constructor, sets output stream, type, and indent units.

• ObjectSerializer (ostream &stream, output\_type\_t outputType)

Constructor, sets the output stream and type, and sets indent units to 4.

PPtr< ObjectSerializer > member (const string &name)

Used to output an object owned by the current one.

PPtr < ObjectSerializer > object (const string &className)

Used to output an object.

PPtr< ObjectSerializer > object (const SerializingStreamer &ref)

Used to output a field which already has SerializingStreamer interface implemented.

• PPtr< ObjectSerializer > endObject ()

Marks the end of an object started by the latest call of object() method.

• PPtr< ObjectSerializer > null ()

Used to print a field that is NULL.

PPtr< ObjectSerializer > attribute (const string &name, const string &value)

Serializes a string attribute.

PPtr< ObjectSerializer > attribute (const string &name, char value)

Serializes a char attribute.

PPtr< ObjectSerializer > attribute (const string &name, int value)

Serializes an int attribute.

• PPtr< ObjectSerializer > attribute (const string &name, unsigned int value)

Serializes an unsigned int attribute.

• PPtr< ObjectSerializer > attribute (const string &name, long int value)

Serializes a long int attribute.

PPtr < ObjectSerializer > attribute (const string &name, unsigned long int value)

Serializes an unsigned long int attribute.

• PPtr< ObjectSerializer > attribute (const string &name, double value)

Serializes a double attribuet.

PPtr< ObjectSerializer > attribute (const string &name, bool value)

Serializes a bool attribute.

PPtr< ObjectSerializer > attribute (const string &name)

Serializes an attribute with no value.

PPtr< ObjectSerializer > collection ()

Used to print a collection.

PPtr< ObjectSerializer > endCollection ()

Marks end of a collection started by the latest unclosed call to collection().

### 6.41.1 Detailed Description

Provides APIs to serialize an object.

This is usually used in conjuction with SerializingStreamer.

Supported output formats are KFOR (KFoundation Format), XML, and JSON. To use, the methods should be called in particular order. If the order is not observed, an expection will be thrown.

- A call to object(const string&) should be made first.
- object() can be followed by attribute(), member(), or endObject().
- attribute() can be followed by attribute(), member() or endObject().

- member() can be followed by object(), collection(), or null().
- There shold be an endObjectr() corresponding to each object() and endCollection() corresponding to each collection().

An expception is when using object(const SerializingStreamer&) or object(const PPtr<T>) it is not needed to call endObject() because it is already called in the serializer() method of the given argument.

All of these methods can be used chained sytax. Example:

```
void serialize(PPtr<ObjectSerializer> os) const {
  os->object("MyClass")
   ->attribute("counter", _counter)
   ->attribute("name", _name)
   ->member("innerObject")->object(innerObject)
   ->endObject();
```

#### 6.41.2 Member Enumeration Documentation

6.41.2.1 enum kfoundation::ObjectSerializer::output\_type\_t

Output format.

**Enumerator** 

```
DUMP KFOR (KFoundation Format)

XML XML.

JSON JSON.
```

# 6.41.3 Constructor & Destructor Documentation

6.41.3.1 kfoundation::ObjectSerializer::ObjectSerializer (ostream & stream, output type toutputType, int indentUnit)

Constructor, sets output stream, type, and indent units.

#### **Parameters**

stream	The stream to print the output to.
outputType	The output format.
indentUnit	Number of spaces for each indention level.

6.41.3.2 kfoundation::ObjectSerializer::ObjectSerializer ( ostream & stream, output type t outputType )

Constructor, sets the output stream and type, and sets indent units to 4.

#### **Parameters**

stream	The stream to print the output to.
outputType	The output format.

### 6.41.4 Member Function Documentation

6.41.4.1 PPtr < ObjectSerializer > kfoundation::ObjectSerializer::collection ( )

Used to print a collection.

Allowed only after member().

Returns

Pointer to self.

6.41.4.2 PPtr < ObjectSerializer > kfoundation::ObjectSerializer::endCollection ( )

Marks end of a collection started by the latest unclosed call to collection().

Returns

Pointer to self.

6.41.4.3 PPtr < ObjectSerializer > kfoundation::ObjectSerializer::endObject ( )

Marks the end of an object started by the latest call of object() method.

**Returns** 

Pointer to self.

6.41.4.4 PPtr < ObjectSerializer > kfoundation::ObjectSerializer::member ( const string & name )

Used to output an object owned by the current one.

Only allowed after object() and attribute().

**Parameters** 

name The corresponding member variable name (property name).

Returns

Pointer to self.

6.41.4.5 PPtr < ObjectSerializer > kfoundation::ObjectSerializer::null ( )

Used to print a field that is NULL.

Only allowed after member().

6.41.4.6 PPtr < ObjectSerializer > kfoundation::ObjectSerializer::object ( const string & className )

Used to output an object.

Allowed in the begining or after member().

**Parameters** 

className Class name of the designated object.

Returns

Pointer to self.

6.41.4.7 PPtr < ObjectSerializer > kfoundation::ObjectSerializer::object ( const SerializingStreamer & ref )

Used to output a field which already has SerializingStreamer interface implemented.

Allowed only after member(). endObject() should NOT be called for this method.

#### **Parameters**

ref The field to be printed.

#### Returns

Pointer to self.

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

- · ObjectSerializer.h
- · ObjectSerializer.cpp

# 6.42 kfoundation::ObjectStreamReader Class Reference

Generic interface for utility object used to read objets from an stream of a given format.

```
#include <kfoundation/ObjectStreamReader.h>
```

## **Public Member Functions**

virtual Ptr < Token > next ()=0 throw (ParseException)
 Returns the next token in the stream (not the token after this one).

# 6.42.1 Detailed Description

Generic interface for utility object used to read objets from an stream of a given format.

No matter what the format of the stream is, it should be organized as if produced by ObjectSerializer.

## See Also

Token

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

- · ObjectStreamReader.h
- ObjectStreamReader.cpp

# 6.43 kfoundation::ObjectToken Class Reference

Represents begining of an object in the parsed stream.

```
#include <kfoundation/ObjectStreamReader.h>
```

## **Public Member Functions**

ObjectToken (const CodeRange &range)

Constructor.

bool checkClass (const string &name) const

Checks if the class name for the parsed object equals the given argument.

void validateClass (const string &name) const

Checks if the class name for the parsed object equals the given argument, and if not will throw an appropriate Parse-Exception.

void throwMissingAttribute (const string &name) const

Throws an exception explaining an attribute with the given name is missing.

· void throwInvlaidClass () const

Throws a ParseException indicating the class name is invalid.

• type\_t getType () const

Returns the type of this token.

## **Static Public Attributes**

static const type\_t TYPE = Token::OBJECT
 Type this token, that is Token::OBJECT.

#### **Additional Inherited Members**

# 6.43.1 Detailed Description

Represents begining of an object in the parsed stream.

## See Also

Token

ObjectStreamReader

## 6.43.2 Member Function Documentation

6.43.2.1 void kfoundation::ObjectToken::validateClass ( const string & name ) const

Checks if the class name for the parsed object equals the given argument, and if not will throw an appropriate ParseException.

## **Parameters**

name The name to check against.

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

- · ObjectStreamReader.h
- · ObjectStreamReader.cpp

# 6.44 kfoundation::OutOfMemoryException Class Reference

# Thrown on out of memory.

#include <kfoundation/OutOfMemoryException.h>

## **Public Member Functions**

OutOfMemoryException (string message)

Constructor.

## **Additional Inherited Members**

## 6.44.1 Detailed Description

Thrown on out of memory.

#### 6.44.2 Constructor & Destructor Documentation

6.44.2.1 kfoundation::OutOfMemoryException::OutOfMemoryException ( string message )

Constructor.

**Parameters** 

message The message to be displayed once the exception is thrown.

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

- · OutOfMemoryException.h
- OutOfMemoryException.cpp

# 6.45 kfoundation::OutputStream Class Reference

Abstract inferface for all output streams.

```
#include <kfoundation/OutputStream.h>
```

## **Public Member Functions**

• virtual void write (const kf\_octet\_t \*buffer, const kf\_int32\_t nOctets)=0

Writes the given number of octets of the given buffer to the stream.

virtual void write (const kf\_octet\_t octet)=0

Writes a single octet to the stream.

virtual void write (PPtr< InputStream > is)=0

Writes the available contents from the given input stream to this output stream.

• virtual void close ()=0

Closes the stream.

• virtual bool isBigEndian () const =0

Checks if the stream is big-endian.

## 6.45.1 Detailed Description

Abstract inferface for all output streams.

## 6.45.2 Member Function Documentation

**6.45.2.1** virtual void kfoundation::OutputStream::close() [pure virtual]

Closes the stream.

It will no longer be readable.

Implemented in kfoundation::InternetOutputStream, kfoundation::BufferOutputStream, kfoundation::Standard-OutputStreamAdapter, and kfoundation::FileOutputStream.

6.45.2.2 virtual void kfoundation::OutputStream::write ( const kf\_octet\_t \* buffer, const kf\_int32\_t nOctets ) [pure virtual]

Writes the given number of octets of the given buffer to the stream.

#### **Parameters**

buffer	The octets to write.
nOctets	Number of octets to write.

Implemented in kfoundation::InternetOutputStream, kfoundation::BufferOutputStream, kfoundation::FileOutputStream, and kfoundation::StandardOutputStreamAdapter.

6.45.2.3 virtual void kfoundation::OutputStream::write ( const kf octet t octet ) [pure virtual]

Writes a single octet to the stream.

#### **Parameters**

octet	The octet to write

Implemented in kfoundation::InternetOutputStream, kfoundation::BufferOutputStream, kfoundation::FileOutputStream, and kfoundation::StandardOutputStreamAdapter.

**6.45.2.4** virtual void kfoundation::OutputStream::write ( PPtr < InputStream > is ) [pure virtual]

Writes the available contents from the given input stream to this output stream.

#### **Parameters**

is	The stream to read from.

Implemented in kfoundation::InternetOutputStream, kfoundation::BufferOutputStream, kfoundation::FileOutputStream, and kfoundation::StandardOutputStreamAdapter.

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

· OutputStream.h

# 6.46 kfoundation::ParseException Class Reference

Thrown when a parsing error happens.

#include <kfoundation/ParseException.h>

## **Public Member Functions**

• ParseException (const string &message)

Constructor, creates a new instance with the given message but no code location.

ParseException (const string &message, const CodeLocation &location)

Constructor, stores the given string message and CodeLocation to report when thrown.

• ParseException (const string &message, const CodeRange &range)

Constructor, stores the given string message and CoreRange to report when thrown.

∼ParseException () throw ()

Deconstrcutor.

· bool hasLocation () const

Checks if a CodeLocation is stored in this object.

bool hasRange () const

Checks if a CodeRange is stored in this object.

· const CodeLocation & getBegin () const

Returns the begining of the range that the error is observed.

· const CodeLocation & getEnd () const

Returns the end of the range that the error is observed.

## **Additional Inherited Members**

## 6.46.1 Detailed Description

Thrown when a parsing error happens.

Stores a string message and a CodeLocation or CodeRange if needed.

#### 6.46.2 Constructor & Destructor Documentation

6.46.2.1 kfoundation::ParseException::ParseException ( const string & message )

Constructor, creates a new instance with the given message but no code location.

#### **Parameters**

message	A message describing the error.

6.46.2.2 kfoundation::ParseException::ParseException ( const string & message, const CodeLocation & location )

Constructor, stores the given string message and CodeLocation to report when thrown.

#### **Parameters**

message	A message describing the error.
The	location of the parsed stream in which the error is observed.

6.46.2.3 kfoundation::ParseException: (const string & message, const CodeRange & range)

Constructor, stores the given string message and CoreRange to report when thrown.

#### **Parameters**

message	A message describing the error.
range	The code range in which the error is observed.

## 6.46.3 Member Function Documentation

6.46.3.1 const CodeLocation & kfoundation::ParseException::getBegin ( ) const

Returns the begining of the range that the error is observed.

Can be used when either hasLocation() or hadRange() is true.

6.46.3.2 const CodeLocation & kfoundation::ParseException::getEnd ( ) const

Returns the end of the range that the error is observed.

Can be used only if hasRange() is true.

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

· ParseException.h

ParseException.cpp

## 6.47 kfoundation::Path Class Reference

Used to represent and manipulate file and directory pathnames.

```
#include <kfoundation/Path.h>
```

## **Public Member Functions**

• Path (const string &str)

Constructor.

• ~Path ()

Deconstructor.

· bool hasExtention () const

Checks if the path has extention.

• bool isAbsolute () const

Checks if the path is absolute.

• int getNSegments () const

Returns the number of segments in the path.

• string getSegement (int index) const

Returns the path segment at the given index.

string getExtention () const

Returns the extension if exists, otherwise, an empty string.

• string getFileName () const

Returns the last segment of the path excluding the extension if exists.

string getFileNameWithExtension () const

Returns the last segment of the path including extention.

Ptr< Path > addSegement (const string &s)

Add a segment to the path.

• Ptr< Path > changeExtension (const string &ex)

Changes the file extensions.

Ptr< Path > removeExtension ()

Removes the extension.

• Ptr< Path > parent ()

Returns the parent path of the current path by removing its last segment.

void makeDir () const

Makes a new directory at the path encoded by this object.

· bool exists () const

Checks if a file or directory pointed by this path object exists.

• void remove () const

Deletes the file or directory pointed by this path object.

• const string & getString () const

Returns the string value of this pat.

void serialize (PPtr< ObjectSerializer > builder) const

Implements compatibility with SerializingStreamer interface.

void printToStream (ostream &os) const

Implements compatibility with Streamer interface.

## **Static Public Attributes**

static const char PATH SEPARATOR = '/'

Path separator symbol on the current platform.

# 6.47.1 Detailed Description

Used to represent and manipulate file and directory pathnames.

Note

For better performance use getString() rather than toString() to get the string value of this object.

#### 6.47.2 Constructor & Destructor Documentation

6.47.2.1 kfoundation::Path::Path ( const string & str )

Constructor.

**Parameters** 

str String containing the desired path. It can be empty.

## 6.47.3 Member Function Documentation

6.47.3.1 Ptr < Path > kfoundation::Path::changeExtension ( const string & ex )

Changes the file extensions.

Adds one if it does not exist.

6.47.3.2 bool kfoundation::Path::hasExtention ( ) const

Checks if the path has extention.

That is, if the last segment of the path contatins a dot '.'.

6.47.3.3 bool kfoundation::Path::isAbsolute ( ) const

Checks if the path is absolute.

That is, if it begins with a separator character.

6.47.3.4 Ptr < Path > kfoundation::Path::parent ( )

Returns the parent path of the current path by removing its last segment.

If the path has only one segment or is empty, returns an empty path.

**6.47.3.5 void kfoundation::Path::printToStream ( ostream &**  *os* **) const** [virtual]

Implements compatibility with Streamer interface.

Prints the serialization of this object in KFOR format to the given stream.

#### **Parameters**

os The stream to print to.

Reimplemented from kfoundation::SerializingStreamer.

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

- · Path.h
- · Path.cpp

# 6.48 kfoundation::PoolObject Class Reference

Superclass for all objects that are meant to be allocated by a pool memory manager.

```
#include <kfoundation/ManagedObject.h>
```

#### **Public Member Functions**

virtual void finalize ()=0

Cleansup this object after each use.

## 6.48.1 Detailed Description

Superclass for all objects that are meant to be allocated by a pool memory manager.

These objects will never be destructed. Instead, they should implement the finalize() method to cleanup themselves after each use.

## 6.48.2 Member Function Documentation

```
6.48.2.1 kfoundation::PoolObject::finalize() [pure virtual]
```

Cleansup this object after each use.

This method should be implemented by subclasses in place of destructor.

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

- · ManagedObject.h
- ManagedObject.cpp

# 6.49 kfoundation::PPtr < T > Class Template Reference

Passive pointer, will not release and retain automatically.

```
#include <kfoundation/Ptr.h>
```

## **Public Member Functions**

• PPtr ()

Default constructor, creates a passive NULL-pointer.

PPtr (const PPtr < T > &pptr)

Copy constructor.

PPtr (const Ptr < T > &aptr)

Copy constructor from nonpassive pointer.

PPtr (const SPtr < T > &sptr)

Copy constructor from static pointer.

• PPtr (T \*const obj)

Creates a passive pointer to the object at the given memory location.

## 6.49.1 Detailed Description

template < typename T> class kfoundation::PPtr< T>

Passive pointer, will not release and retain automatically.

This is useful to produce a faster code in situations that you are sure the retain count of an object is the same in the begining and the end of the scope of difinition of the pointer.

## 6.49.2 Constructor & Destructor Documentation

```
6.49.2.1 template < typename T > kfoundation::PPtr < T >::PPtr ( )
```

Default constructor, creates a passive NULL-pointer.

Passive pointers do not automatically retain or release.

```
6.49.2.2 template<typename T> kfoundation::PPtr< T>::PPtr ( const PPtr< T> & obj )
```

Copy constructor.

**Parameters** 

obi	The pointer to be copied.

6.49.2.3 template<typename T> kfoundation::PPtr< T>::PPtr ( const Ptr< T> & obj )

Copy constructor from nonpassive pointer.

**Parameters** 

obj   The pointer to be copied.
---------------------------------

## 6.49.2.4 template<typename T> kfoundation::PPtr< T>::PPtr ( const SPtr< T> & obj )

Copy constructor from static pointer.

**Parameters** 

obj	The pointer to be copied.

6.49.2.5 template<typename T> kfoundation::PPtr< T>::PPtr ( T\*const obj )

Creates a passive pointer to the object at the given memory location.

#### **Parameters**

obj Should be a valid instance of ManagedObject or one of its subclasses, or NULL.

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

- · ObjectPoolMemoryManagerDecl.h
- · PtrDecl.h
- Ptr.h

## 6.50 kfoundation::PredictiveParserBase Class Reference

Packs ample of basic functionalities to implement any predictive parser.

#include <kfoundation/PredictiveParserBase.h>

#### **Public Member Functions**

PredictiveParserBase (PPtr< InputStream > input)

Constructor, creates a parser that reads symbols from the given stream.

∼PredictiveParserBase ()

Deconstructor.

bool testChar (const wchar\_t &t)

Tests if the given character is next in the stream.

bool testChar (const wchar t \*chars, const int &n)

Thest if any of the given characters is next in the stream.

bool testAlphabet ()

Thest if the next character is alphabet.

• bool testAlphanumeric ()

Tests if the next character is alphanumeric.

bool testSpace ()

Test if the next character is space.

• bool testNewLine ()

Test if the next character is a newline character  $' \ n'$ .

• bool testEndOfStream ()

Checks if the end of stream is reached.

bool testSequence (const wstring &str)

Checks if the given string is next in the stream.

unsigned short int readChar (const wchar\_t &t, kf\_octet\_t \*octets=0)

Checks if the given character is next in the stream, and reads it if so.

• unsigned short int readChar (const wchar\_t \*chars, const int &n, kf\_octet\_t \*octets=0)

Checks if any of the given characters is next in the stream, and reads it if so.

unsigned short int readAlphabet (wchar\_t &ch, kf\_octet\_t \*octets=0)

Checks if the next character in the stream is an alphabet, and reads it if so.

unsigned short int readNumeric (wchar\_t &ch, kf\_octet\_t \*octets=0)

Checks if the next character in the stream is a digit, and reads it if so.

unsigned short int readNumeric (unsigned short int &digit)

Checks if the next character in the stream is a digit, and reads it and converts it to its equivalant numeric value.

• unsigned short int readAlphanumeric (wchar t &ch, kf octet t \*octets=0)

Checks if the next character in the stream is a letter or a digit, and reads it and converts it to its equivalant numeric value.

• unsigned short int readIdentifierBeginChar (wchar\_t &ch, kf\_octet\_t \*octets=0)

Checks if the next character in the stream is a valid identifier begining character, and if so, reads it.

• unsigned short int readIdentifierChar (wchar\_t &ch, kf\_octet\_t \*octets=0)

Checks if the next character in the stream is a valid identifier character, and if so, reads it.

• unsigned short int readSpace ()

Checks if the next character in the stream is a space, and reads it if so.

unsigned short int readNewLine ()

Checks if the next character in the stream is newline ' and reads it if so.

unsigned short int readAny (wchar\_t &ch, kf\_octet\_t \*octets=0)

Reads any character next in the stream unless the end of stream is reached.

unsigned short int readSequence (const wstring &str)

Checks if the next sequence of characters match the given string, and reads them if so.

• size\_t readAllAlphabet (string &storage)

Reads all the next characters that are alphabet and appends them to the given argument.

• size\_t readAllAlphanumeric (string &storage)

Reads all the next characters that are alphanumeric and appends them to the given argument.

• size\_t readAllNumeric (string &storage)

Reads all the next numeric characters and appends them to the given parameter.

• size\_t readNumber (string &storage)

Checks if the next character(s) represent a number (+|-)?[0..9]+(.

size t readNumber (long int &output)

Checks if the next character(s) represents an integer number and if so, reads and converts it to it equivalant integer

size\_t readNumber (double &output)

Checks if the next character(s) represents a real number and if so, reads and converts it to it equivalant integer value.

• size t readIdentifier (string &storage)

Reads all the next characters in the stream that form an identifier, and appends them to the given argument.

size\_t readAllBeforeSpace (string &storage)

Reads all character before a space (or end of stream) is encountered.

size\_t readAllBeforeNewLine (string &storage)

Reads all character before a new line (or end of stream) is encountered.

size\_t readAllBeforeSpaceOrNewLine (string &storage)

Reads all character before space or new line (or end of stream) is encountered.

size\_t readAllBeforeChar (const wchar\_t ch, string &storage)

Read all characters before the given character (or end of stream) is encountered.

size\_t readAllBeforeCharSkipEscaped (const wchar\_t ch, const wchar\_t escape, string &storage)

Reads all characters before the given character, but skips all the matching ones immidiately after the given excape character.

• size\_t readAllBeforeSequence (const wstring &str, string &storage)

Reads all character before the given sequence.

size\_t skipSpaces ()

Consumes all the spaces next in the stream.

size\_t skipSpacesAndNewLines ()

Consumes all the spaces and new line characters next in the stream.

const CodeLocation & getCodeLocation () const

Returns the CodeLocation object corresponding to the current position in the stream.

#### **Protected Member Functions**

virtual bool isValidIdentifierBeginChar (const wchar t &ch) const

Checks if the given argument is a valid begining of an identifier name.

virtual bool isValidIdentifierChar (const wchar\_t &ch) const

Checks if given argument is can be used after the first character of an identifier name.

virtual bool isSpace (const wchar\_t &ch) const

Checks if the given character is a white space.

## 6.50.1 Detailed Description

Packs ample of basic functionalities to implement any predictive parser.

It can be directly instantiated, or be extended to implement a parser for the desired grammar. Protected methods isValidIdentifierBeginChar(), isValidIdentifierChar(), and isSpace() can be overriden if needed.

There are two set of methods for reading the input. The readXXX methods consume the stream characters if successful, and return the number read character. If faild the stream will remain at its current position and return value will be zero. The testXXX methods will test if the given argument is there next in the stream without consuming anything.

#### 6.50.2 Constructor & Destructor Documentation

6.50.2.1 kfoundation::PredictiveParserBase::PredictiveParserBase ( PPtr < InputStream > input )

Constructor, creates a parser that reads symbols from the given stream.

Parameters

input	The stream to parse.

## 6.50.3 Member Function Documentation

6.50.3.1 bool kfoundation::PredictiveParserBase::isSpace ( const wchar\_t & ch ) const [protected], [virtual]

Checks if the given character is a white space.

The default implementation returns 'ch == ' ". Override to customize the parser based on your particular needs.

Parameters

ch	The character to be checked.

**6.50.3.2** bool kfoundation::PredictiveParserBase::isValidIdentifierBeginChar ( const wchar\_t & ch ) const [protected], [virtual]

Checks if the given argument is a valid begining of an identifier name.

Override to customize the parser based on your particular needs. The default implementation returns 'isAplhabet(ch) || ch == '\_'.

**Parameters** 

ch | The character to be tested.

**6.50.3.3** bool kfoundation::PredictiveParserBase::isValidIdentifierChar( const wchar\_t & ch ) const [protected], [virtual]

Checks if given argument is can be used after the first character of an identifier name.

Override to customize the parser based on your particular needs. The default implementation returns 'is-Alphanumeric(ch) || ch == '\_'.

#### **Parameters**

ch The character to be checked.

6.50.3.4 size\_t kfoundation::PredictiveParserBase::readAllAlphabet ( string & storage )

Reads all the next characters that are alphabet and appends them to the given argument.

#### **Parameters**

storage	Output, read octets will be appended to it.
---------	---

#### Returns

The number of octets read.

6.50.3.5 size\_t kfoundation::PredictiveParserBase::readAllAlphanumeric ( string & storage )

Reads all the next characters that are alphanumeric and appends them to the given argument.

#### **Parameters**

storage	Output, read octets will be appended to it.
---------	---

## Returns

The number of octets read.

6.50.3.6 size\_t kfoundation::PredictiveParserBase::readAllBeforeChar ( const wchar\_t t, string & storage )

Read all characters before the given character (or end of stream) is encountered.

## **Parameters**

t	The character to read until.
storage	Output, the read characters will be appended to it.

## Returns

The number of read octets.

6.50.3.7 size\_t kfoundation::PredictiveParserBase::readAllBeforeCharSkipEscaped ( const wchar\_t t, const wchar\_t escape, string & storage )

Reads all characters before the given character, but skips all the matching ones immidiately after the given excape character.

#### **Parameters**

t	The character to read until.
escape	The escape character.
storage	Output, the read characters will be appended to it.

#### **Returns**

The number of read octets.

6.50.3.8 size\_t kfoundation::PredictiveParserBase::readAllBeforeNewLine ( string & storage )

Reads all character before a new line (or end of stream) is encountered.

#### **Parameters**

storage	Output, the octets read will be appended to it.

## Returns

The number of read octets.

6.50.3.9 size\_t kfoundation::PredictiveParserBase::readAllBeforeSequence ( const wstring & str, string & storage )

Reads all character before the given sequence.

## **Parameters**

str	The sequence of characters to read until.
storage	Output, the read characters will be appended to it.

## Returns

The number of read octets.

6.50.3.10 size\_t kfoundation::PredictiveParserBase::readAllBeforeSpace ( string & storage )

Reads all character before a space (or end of stream) is encountered.

# Parameters

storage	Output, the octets read will be appended to it.

## Returns

The number of read octets.

6.50.3.11 size\_t kfoundation::PredictiveParserBase::readAllBeforeSpaceOrNewLine ( string & storage )

Reads all character before space or new line (or end of stream) is encountered.

#### **Parameters**

storage	Output, the octets read will be appended to it.

## Returns

The number of read octets.

6.50.3.12 size\_t kfoundation::PredictiveParserBase::readAllNumeric ( string & storage )

Reads all the next numeric characters and appends them to the given parameter.

#### **Parameters**

storage	Output, read octets will be appended to it.

## Returns

The number of read octets.

6.50.3.13 unsigned short int kfoundation::PredictiveParserBase::readAlphabet ( wchar\_t & ch, kf\_octet\_t \* octets = 0 )

Checks if the next character in the stream is an alphabet, and reads it if so.

#### **Parameters**

ch	Output, will be assigned with the read character.
octets	If not $\mathtt{NULL}$ , the read octets are written on the buffer pointed by this argument. Default value
	is NULL.

## Returns

The number of read octets.

6.50.3.14 unsigned short int kfoundation::PredictiveParserBase::readAlphanumeric ( wchar\_t & ch, kf\_octet\_t \* octets = 0 )

Checks if the next character in the stream is a letter or a digit, and reads it and converts it to its equivalant numeric value.

#### **Parameters**

ch	Output, will be assigned to the read character.
octets	If not NULL, the read octets are written on the buffer pointed by this argument. Default value
	is NULL.

## Returns

The number of read octets.

6.50.3.15 unsigned short int kfoundation::PredictiveParserBase::readAny ( wchar\_t & ch, kf\_octet\_t \* octets = 0 )

Reads any character next in the stream unless the end of stream is reached.

#### **Parameters**

ch	Output, will be assigned to the character read.
octets	If not NULL, the read octets are written on the buffer pointed by this argument. Default value
	is NULL.

## Returns

The number of octets read.

6.50.3.16 unsigned short int kfoundation::PredictiveParserBase::readChar ( const wchar\_t & t, kf\_octet\_t \* octets = 0 )

Checks if the given character is next in the stream, and reads it if so.

#### **Parameters**

t	The character to be expected next in the stream.
octets	If not NULL, the read octets are written on the buffer pointed by this argument. Default value
	is NULL.

## Returns

The number of read octets.

6.50.3.17 unsigned short int kfoundation::PredictiveParserBase::readChar ( const wchar\_t \* chars, const int & n, kf\_octet\_t \* octets = 0 )

Checks if any of the given characters is next in the stream, and reads it if so.

## **Parameters**

chars	The list of character to check against.
n	The number of characters in the given list.
octets	If not $\mathtt{NULL}$ , the read octets are written on the buffer pointed by this argument. Default value is $\mathtt{NULL}$ .

## Returns

The number of read octets.

6.50.3.18 size\_t kfoundation::PredictiveParserBase::readIdentifier ( string & storage )

Reads all the next characters in the stream that form an identifier, and appends them to the given argument.

Override is ValidIdentifier Begin Char() and is ValidIdentifier Char() to customize the behavior of this method.

## **Parameters**

storage	Output, the octets read will be appended to it.

## Returns

The number of read octets.

6.50.3.19 unsigned short int kfoundation::PredictiveParserBase::readIdentifierBeginChar ( wchar\_t & ch, kf\_octet\_t \* octets = 0 )

Checks if the next character in the stream is a valid identifier begining character, and if so, reads it.

Override is Valid Identifier Begin Char() to customize the behavior of this method.

#### **Parameters**

ch	Output, will be assigned to the read character.
octets	If not NULL, the read octets are written on the buffer pointed by this argument. Default value
	is NULL.

#### Returns

The number of read octets.

6.50.3.20 unsigned short int kfoundation::PredictiveParserBase::readIdentifierChar ( wchar\_t & ch, kf\_octet\_t \* octets = 0 )

Checks if the next character in the stream is a valid identifier character, and if so, reads it.

Override is ValidIdentifier Begin Char() to customize the behavior of this method.

#### **Parameters**

ch	Output, will be assigned to the read character.
octets	If not NULL, the read octets are written on the buffer pointed by this argument. Default value
	is NULL.

## Returns

The number of read octets.

6.50.3.21 unsigned short int kfoundation::PredictiveParserBase::readNewLine ( )

Checks if the next character in the stream is newline '

' and reads it if so.

#### Returns

The number of read character.

6.50.3.22 size\_t kfoundation::PredictiveParserBase::readNumber ( string & storage )

Checks if the next character(s) represent a number (+|-)?[0..9]+(...

[0..9]\*)?, and reads them if so. Read characters are appended to the given parameter.

#### **Parameters**

storage	Output, the read octets will be appended to it.

#### Returns

The number of octets read.

6.50.3.23 size\_t kfoundation::PredictiveParserBase::readNumber ( long int & output )

Checks if the next character(s) represents an integer number and if so, reads and converts it to it equivalant integer value.

#### **Parameters**

output	The integer number read.
--------	--------------------------

## Returns

The number of octets read.

6.50.3.24 size\_t kfoundation::PredictiveParserBase::readNumber ( double & output )

Checks if the next character(s) represents a real number and if so, reads and converts it to it equivalent integer value.

#### **Parameters**

output	The integer number read.
--------	--------------------------

## Returns

The number of octets read.

6.50.3.25 unsigned short int kfoundation::PredictiveParserBase::readNumeric ( wchar\_t & ch, kf\_octet\_t \* octets = 0 )

Checks if the next character in the stream is a digit, and reads it if so.

#### **Parameters**

ch	Output, will be assigned with the read character.
octets	If not NULL, the read octets are written on the buffer pointed by this argument. Default value
	is NULL.

## Returns

The number of read octets.

6.50.3.26 unsigned short int kfoundation::PredictiveParserBase::readNumeric ( unsigned short int & digit )

Checks if the next character in the stream is a digit, and reads it and converts it to its equivalant numeric value.

#### **Parameters**

digit	Output, will be assigned to the numeric equivalant of the read character.

## Returns

The number of read octets.

6.50.3.27 unsigned short int kfoundation::PredictiveParserBase::readSequence ( const wstring & str )

Checks if the next sequence of characters match the given string, and reads them if so.

#### **Parameters**

str	The sequence of characters to check agains.

## Returns

The number of octets read.

6.50.3.28 unsigned short int kfoundation::PredictiveParserBase::readSpace ( )

Checks if the next character in the stream is a space, and reads it if so.

Override isSpace() to customzie the behavior of this function.

#### Returns

The number of read octers.

6.50.3.29 size\_t kfoundation::PredictiveParserBase::skipSpaces ( )

Consumes all the spaces next in the stream.

#### Returns

The number of read octets.

6.50.3.30 size\_t kfoundation::PredictiveParserBase::skipSpacesAndNewLines ( )

Consumes all the spaces and new line characters next in the stream.

## Returns

The number of read octets.

6.50.3.31 bool kfoundation::PredictiveParserBase::testChar ( const wchar\_t & ch )

Tests if the given character is next in the stream.

## Parameters

ch	The character to test.
On	The character to toot.

6.50.3.32 bool kfoundation::PredictiveParserBase::testChar ( const wchar\_t \* chars, const int & n )

Thest if any of the given characters is next in the stream.

#### **Parameters**

chars	The list of characters to test against.
n	The number of characters in the given list.

6.50.3.33 bool kfoundation::PredictiveParserBase::testSequence ( const wstring & str )

Checks if the given string is next in the stream.

#### **Parameters**

str The sequence of characters to check against.

6.50.3.34 bool kfoundation::PredictiveParserBase::testSpace ( )

Test if the next character is space.

Override isSpace() to customize the behavior of this function.

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

- · PredictiveParserBase.h
- PredictiveParserBase.cpp

# 6.51 kfoundation::ProximityIterator Class Reference

Iterates the proximity of a desired point.

#include <kfoundation/ProximityIterator.h>

#### **Public Member Functions**

ProximityIterator (int radius)

Constructor, sets the rectangular radius around the point to be iterated.

ProximityIterator (int radius, const Tuple &center)

Constructor, sets the point and the rectangular radius around it to be iterated.

ProximityIterator & centerAt (const Tuple &center)

Resets this iterator, centering the iteration region at the given point.

ProximityIterator & next ()

Moves on to the next point.

• bool hasMore ()

Checks if there are more points to iterate.

## **Additional Inherited Members**

## 6.51.1 Detailed Description

Iterates the proximity of a desired point.

Particularly useful for stencil computation.

#### 6.51.2 Constructor & Destructor Documentation

6.51.2.1 kfoundation::ProximityIterator::ProximityIterator (int radius)

Constructor, sets the rectangular radius around the point to be iterated.

The point which its proximity is being iterated can be set later using centerAt() method.

6.51.2.2 kfoundation::ProximityIterator::ProximityIterator (int radius, const Tuple & center)

Constructor, sets the point and the rectangular radius around it to be iterated.

The point at center can later be changes using centerAt() method.

## 6.51.3 Member Function Documentation

## 6.51.3.1 ProximityIterator & kfoundation::ProximityIterator::next ( )

Moves on to the next point.

Returns

Reference to self.

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

- · ProximityIterator.h
- · ProximityIterator.cpp

# 6.52 kfoundation::Ptr < T > Class Template Reference

Managed pointer to a class of given template type.

```
#include <kfoundation/Ptr.h>
```

## **Public Member Functions**

• Ptr ()

Constructs a NULL-pointer to the given template class type.

Ptr (T \*obj, bool trace=false)

Converts a given C pointer to managed pointer.

Ptr (const Ptr < T > &other)

Copy constructor.

virtual ∼Ptr ()

Deconstructor.

Ptr< T > & trace ()

Enables trace mode.

Ptr< T > & untrace ()

Disables trace mode.

Ptr< T > & del ()

Manually releases the pointed object, and makes the pointer passive for the rest of its life time.

• Ptr< T > & retain ()

Increases the retain count for the object pointed by this pointer by one.

• Ptr< T > & release ()

Decreases the retain count for the object pointed by this pinter by one.

Ptr< T > & replace (T \*const &replacement)

Changes the object pointed to by this pointer, releases the previous object and retains the new one.

Ptr< T > & replace (const Ptr< T > &replacement)

Changes the object pointed to by this pointer, releases the previous object and retains the new one.

• int getRetainCount () const

Returns the retain count for the pointed object.

bool isNull () const

Checks if this is a NULL-pointer.

• bool isValid () const

Checks if this is a valid pointed.

• T & operator\* () const

Dereference operator.

• T \* toPurePtr () const

Returns the actual memory location the pointed object is stored.

• T \* operator-> () const

Structure dereference operator.

Ptr< T > & operator= (const Ptr< T > &other)

Replaces the pointed object with a new one, releases the previous object, and retains the new one.

Ptr < T > & operator= (T \*const &obj)

Replaces the pointed object with a new one, releases the previous object, and retains the new one.

bool operator== (const Ptr< T > &other) const

Equality operator.

bool operator== (const T \*ptr) const

Equality operator between a managed pointer and a classic pointer.

bool operator!= (const Ptr< T > &other) const

Inequality operator.

bool operator!= (const T \*ptr) const

Inequality operator between a managed pointer and a class pointer.

template<typename K >

bool isa () const

Checks if the pointed object is an instance of the given template argument.

template<typename K >

```
Ptr < K > cast () const
```

Casts the pointed object to the type given as template argument.

string toShortString () const

Returns a short string representation of this pointer.

void serialize (PPtr< ObjectSerializer > builder) const

Serializing method.

## 6.52.1 Detailed Description

```
template<typename T>class kfoundation::Ptr< T>
```

Managed pointer to a class of given template type.

The template type should be a subclass of ManagedObject. To use, try

```
Ptr<MyClass> myObject = new MyClass();
```

To create a null pointer, try

```
Ptr<MyClass> myObject = NULL;
```

or just

```
Ptr<MyClass> myObject;
```

After this, it can be used just like an ordinary poitner.

```
{\tt myObject->myMethod();}
```

Ptr prevents segmentation fault situations to happen. If the object being dereferenced is NULL, a NullPointer-Exception will be thrown. If the object is invalid, i.e. it is previously deconstructed, an InvalidPointerException will be thrown.

You can also manually check the validity of the pointer using isValid() method. isValid() will also return false if the pointer is NULL. To check the for NULL pointer, use isNull() method.

There is an elegant way to type-case a managed pointer:

```
Ptr<MySuperClass> myCastedObject = myObject.AS(MySuperClass);
```

Similiarily, there is an elegant way to check the type of a managed pointer.

```
if(myObject.ISA(MySuperClass)) {
  LOG << "I Love KFoundation <3" << EL;
}</pre>
```

Managed objects do not need to be explicitly destructed. Ptr will automatically retain and release the instance of the object it is pointing to whenever necessary, and calls the destructor when the object instance is no longer needed.

You can check the retain count using getRetainCount() method. Inputting a Ptr to stream or logger will print a more detailed description of it.

```
LOG << myObject << EL;
```

If you mean to print the content of the object being pointed to, rather than the content of the pointer, make sure to dereference it.

```
LOG << *myObject << EL;
```

There are two variants of Ptr: Passive Pointer (PPtr) and Static Pointer (SPtr). PPtr do not retain or release the object with the scope they are defined. If you are sure within a cerain scope the retain count of an object will not be changed, it is advisable to use PPtr to produce a faster program. SPtr makes the pointed object immortal. It should be used for static class members.

KFoundation managed pointers are designed to be fast and efficient. The size of Ptr is exactly 8 bytes — the same as normal pointers on most platforms. To make it safe, the validity of the pointer is checked against the memory manager's registery on each access. To make it fast, a novel fast algorithm with O(1) time complexity is developed to do the task.

On rare ocasions it might be needed to manage the reference count manually. In such cases you may use retain(), release() and replace() methods. But unless absolutely necessary please refrain from using these methods.

#### 6.52.2 Constructor & Destructor Documentation

```
6.52.2.1 template < typename T > kfoundation::Ptr < T >::Ptr ( T * obj, bool trace = false )
```

Converts a given C pointer to managed pointer.

#### **Parameters**

obj	Pointer to a ManagedObject or one of its subclasses, or NULL.
trace	Used for debugging. Causes a lot of information to be printed everytime the pointer is modi-
	fied.

6.52.2.2 template<typename T> kfoundation::Ptr< T>::Ptr( const Ptr< T> & other) [inline]

Copy constructor.

# Parameters

other	The other pointer to be copied.

6.52.2.3 template<typename  $T > kfoundation::Ptr < T > :: \sim Ptr()$  [virtual]

Deconstructor.

If not a passive pointer, the pointed object will be released.

## 6.52.3 Member Function Documentation

```
6.52.3.1 template<typename T > template<typename K > Ptr< K > kfoundation::Ptr< T >::cast ( ) const
```

Casts the pointed object to the type given as template argument.

```
Ptr<MySuperClass> castedPtr = myPtr.cast<MySuperClass>();
```

A more elegant syntax is available through AS(X) macro.

```
Ptr<MySuperClass> castedPtr = myPtr.AS(MySuperClass);
```

```
6.52.3.2 template<typename T > Ptr< T > & kfoundation::Ptr< T >::del( )
```

Manually releases the pointed object, and makes the pointer passive for the rest of its life time.

This method is only available with DEBUG macro defined.

Returns

Self

```
6.52.3.3 template<typename T > int kfoundation::Ptr< T >::getRetainCount( ) const
```

Returns the retain count for the pointed object.

If this pointer is invaild (or NULL) the returned value will be negative.

```
\textbf{6.52.3.4} \quad \textbf{template} < \textbf{typename T} > \textbf{template} < \textbf{typename K} > \textbf{bool kfoundation} :: \textbf{Ptr} < \textbf{T} > :: \textbf{isa ( ) const} \quad \texttt{[inline]}
```

Checks if the pointed object is an instance of the given template argument.

```
if(myPtr.isa<MyClass>()) { ... }
```

A more elegant syntax is provided by ISA(X) macro.

```
if(myPtr.ISA(MyClass)) { ... }
```

```
6.52.3.5 template<typename T > bool kfoundation::Ptr< T >::isValid( ) const [inline]
```

Checks if this is a valid pointed.

A pointer is valid if it is not NULL and it points to an existing object.

```
6.52.3.6 template<typename T > T & kfoundation::Ptr< T >::operator*( ) const [inline]
```

Dereference operator.

Functions the same as normal C/C++ dereference operator.

```
6.52.3.7 template<typename T > T * kfoundation::Ptr< T >::operator-> ( ) const [inline]
```

Structure dereference operator.

Functions the same as normal C/C++ equivalant.

Replaces the pointed object with a new one, releases the previous object, and retains the new one.

Internally, it calls replace(const Ptr<T>&).

Returns

Self

```
6.52.3.9 template < typename T > Ptr < T > & kfoundation::Ptr < T >::operator= ( T *const & obj ) [inline]
```

Replaces the pointed object with a new one, releases the previous object, and retains the new one.

Internally, it calls replace(T\* const&).

Returns

Self

```
6.52.3.10 template < typename T > Ptr< T > & kfoundation::Ptr< T >::release ( )
```

Decreases the retain count for the object pointed by this pinter by one.

If the retain count reaches zero, the object will be deleted.

Returns

Self

```
6.52.3.11 template < typename T > Ptr < T > & kfoundation::Ptr < T >::replace ( T * const & replacement ) [inline]
```

Changes the object pointed to by this pointer, releases the previous object and retains the new one.

**Parameters** 

replacement	The new object to point to. It should be an instance of ManagedObject or one of its sub-
	classes, or NULL.

Returns

Self

```
6.52.3.12 template < typename T > Ptr< T > & kfoundation::Ptr< T >::replace ( const Ptr< T > & replacement ) [inline]
```

Changes the object pointed to by this pointer, releases the previous object and retains the new one.

**Parameters** 

replacement	A pointer to the replacement object.

Returns

Self

```
6.52.3.13 template < typename T > Ptr < T > & kfoundation::Ptr < T > ::retain ( )

Increases the retain count for the object pointed by this pointer by one.

Returns
Self

6.52.3.14 template < typename T > Ptr < T > & kfoundation::Ptr < T > ::trace ( )

Enables trace mode.

Causes a lot of information to be printed every time this pointer is modified. This method is only available when compiled with DEBUG macro defined.

Returns
Self

See Also
untrace()
```

6.52.3.15 template<typename T > Ptr< T > & kfoundation::Ptr< T >::untrace ( )

Disables trace mode.

This method is only available when compiled with DEBUG macro defined.

Returns

Self

See Also

trace()

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

- · PtrDecl.h
- Ptr.h

# 6.53 kfoundation::Range Class Reference

Represents a range in n-dimensional space.

```
#include <kfoundation/Range.h>
```

## **Public Member Functions**

Range ()

Default constructor.

• Range (const Tuple &begin, const Tuple &end)

Constructor, creates a range with the given begining and end.

Range (const Tuple &end)

Constructor, creates a range with its begining at the origin and end at the given parameter.

Range (const Range &other)

Copy constructor.

kf\_int8\_t getNDimensions () const

Returns the number of dimensions.

• const Tuple & getBegin () const

Returns the begining of this range.

const Tuple & getEnd () const

Returns the end of this range.

kf int64 t indexToOrdinal (const Tuple &index) const

Returns a unique integer number for the given point in this range.

Range translate (const Tuple & amount) const

Returns the translation of this range moved by the given tuple.

Range grow (const int s) const

Returns the result of moving the boundaries of this range outwards by the given value.

• Range shrink (const int s) const

Returns the result of moving the boundaries of this range inwards by the given value.

• Range border (const Direction &d, const int s) const

Returns the range of elements on the border of this range with given thickness.

· Range flip (const Direction &d) const

Returns the result of flipping this range to the given direction.

Range intersectWith (const Range & other) const

Returns the result of intesection of this range and the given parameter.

Range joinWith (const Range & other) const

Returns the smallest range containing both this range and the given parameter.

Range joinWith (const Tuple &point) const

Returns the smallest range containing both this range and the given point.

• Range divide (const Tuple &divisor, const Tuple &selector) const

Divides this range to the given divisor and returns the one at the given index.

bool isAdjecentTo (const Range &other) const

Checks if this range is adjecent to the given one.

bool contains (const Tuple &point) const

Checks of this range contains the given point.

• bool contains (const Range &other) const

Checks if this range contains the given range.

Direction getRelativePositionTo (const Range &other) const

Returns the relative direction of this range to the given one.

• bool isEmpty () const

Checks of this range is empty, i.e.

• const Tuple & getSize () const

Returns the number of dimensions of this range.

kf\_int64\_t getVolume () const

Returns the volume of this range.

· Rangelterator getIterator () const

Returns an iterator for this range.

Range operator+ (const int n) const

Returns the result of adding the given scalar to all elements of this range.

• Range operator- (const int n) const

Returns the result of substracting the given scalar from all elements of this range.

• Range operator\* (const int n) const

Returns the result of multiplying the given scalar to all elements of this range.

Range operator/ (const int n) const

Returns the result of dividing each element of this range to the given scalar.

void printToStream (ostream &os) const

Implements compatibility with Streamer interface.

## 6.53.1 Detailed Description

Represents a range in n-dimensional space.

A range has a begining and and end, each represented by a tuple of the same dimensions.

## 6.53.2 Constructor & Destructor Documentation

6.53.2.1 kfoundation::Range::Range()

Default constructor.

Creates a range in 0-D space.

#### 6.53.3 Member Function Documentation

6.53.3.1 Range kfoundation::Range::border ( const Direction & d, const int s ) const

Returns the range of elements on the border of this range with given thickness.

#### **Parameters**

d	The direction of the desired broder.
S	The thickness of the desired border.

#### 6.53.3.2 bool kfoundation::Range::isEmpty ( ) const

Checks of this range is empty, i.e.

it's volume is zero.

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

- · Range.h
- · Range.cpp

# 6.54 kfoundation::Rangelterator Class Reference

Used to iterate over all points in a given range.

#include <kfoundation/RangeIterator.h>

## **Public Member Functions**

• Rangelterator (const Tuple &upperBound)

Constructor, the resulting object iterates points from origin to the given upper bound (exclusive).

Rangelterator (const Tuple &lowerBound, const Tuple &upperBound)

Constructor, the resulting object iterates points from the given lower bound (inclusive) to the given upper bound (exclusive).

Rangelterator (const Range &range)

Copy constructor.

∼Rangelterator ()

Deconstructor.

· const Rangelterator & first ()

Resets this iterator.

· const Rangelterator & next ()

Moves on to the next point.

• bool hasMore () const

Checks if there are more points to iterator.

#### **Additional Inherited Members**

## 6.54.1 Detailed Description

Used to iterate over all points in a given range.

The begining of the range is inclusive and the end of it is exclusive. Usage:

```
Range r(Tupel2D(10, 10), Tuple2D(20, 30));
for(RangeIterator(r); r.hasMore(); r.next()) {
    Tuple point = r;
    ... do somethinf with r ...
}
```

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

- · Rangelterator.h
- · Rangelterator.cpp

# 6.55 kfoundation::RefCountMemoryManager Class Reference

Reference counting memory manager.

```
#include <kfoundation/RefCountMemoryManager.h>
```

## **Public Member Functions**

RefCountMemoryManager (MasterMemoryManager \*master)

Constructor.

∼RefCountMemoryManager ()

Deconstructor.

const ObjectRecord & registerObject (ManagedObject \*obj)

Creates a new entry for the object at the given memory location.

void retain (kf int32 t index, kf int16 t key)

Retains the object associated with the given index if key maches, otherwise throws InvalidPointerException.

• void release (kf\_int32\_t index, kf\_int16\_t key)

Releases the object associated with the given index if the key matches, otherwise throws InvalidPointerException.

void remove (kf\_int32\_t index, kf\_int16\_t key)

Unmanages the object at the given index if the key matches, otherwise, throws InvalidPointerException.

ObjectRecord \* getTable ()

Returns the memory locatoin of the begining of this manager's object table.

kf\_int32\_t getTableSize () const

Returns the number of available table records.

 void serialize (PPtr< ObjectSerializer > seralizer) const Serializing method.

## 6.55.1 Detailed Description

Reference counting memory manager.

#### 6.55.2 Member Function Documentation

```
6.55.2.1 ObjectRecord * kfoundation::RefCountMemoryManager::getTable() [virtual]
```

Returns the memory locatoin of the begining of this manager's object table.

The object table is a one dimentional array of ObjectRecord.

See Also

```
getTableSize()
```

Implements kfoundation::MemoryManager.

```
6.55.2.2 kf_int32_t kfoundation::RefCountMemoryManager::getTableSize() const [virtual]
```

Returns the number of available table records.

Since some records can be unused, this is usually not equal but larger than the the number of objects managed by this manager.

See Also

```
getTable()
```

Implements kfoundation::MemoryManager.

```
6.55.2.3 void kfoundation::RefCountMemoryManager::release( kf_int32_t index, kf_int16_t key) [virtual]
```

Releases the object associated with the given index if the key matches, otherwise throws InvalidPointerException.

If the retain count drops to zero the object will be deleted.

Implements kfoundation::MemoryManager.

```
6.55.2.4 void kfoundation::RefCountMemoryManager::remove( kf_int32_t index, kf_int16_t key ) [virtual]
```

Unmanages the object at the given index if the key matches, otherwise, throws InvalidPointerException.

The index will be reused in the future.

Implements kfoundation::MemoryManager.

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

- · RefCountMemoryManager.h
- · RefCountMemoryManager.cpp

# 6.56 kfoundation::SerializingStreamer Class Reference

Objects implementing this class can be serialized into any format allowed by ObjectSerializer.

#include <kfoundation/SerializingStreamer.h>

#### **Public Member Functions**

· virtual string toString () const

Returns a string containing the serialization of this object in KFOR format.

virtual void serialize (PPtr< ObjectSerializer > builder) const =0

Implements compatibility with SerializingStreamer interface.

virtual void printToStream (ostream &os) const

Implements compatibility with Streamer interface.

void printToJsonStream (ostream &os) const

Prints the serialization of this object in JSON format to the given stream.

void printToXmlStream (ostream &os) const

Prints the serialization of this object in XML format to the given stream.

• string toJsonString () const

Returns a string containing the serialization of this object in JSON format.

• string toXmlString () const

Returns a string containing the serialization of this object in XML format.

## 6.56.1 Detailed Description

Objects implementing this class can be serialized into any format allowed by ObjectSerializer.

The output is used to provide compatibility with Streamer interfaceo.

## 6.56.2 Member Function Documentation

6.56.2.1 void kfoundation::SerializingStreamer::printToJsonStream (ostream & os) const

Prints the serialization of this object in JSON format to the given stream.

**Parameters** 

os The stream to print to.

**6.56.2.2** void kfoundation::SerializingStreamer::printToStream (ostream & os ) const [virtual]

Implements compatibility with Streamer interface.

Prints the serialization of this object in KFOR format to the given stream.

**Parameters** 

os The stream to print to.

Implements kfoundation::Streamer.

Reimplemented in kfoundation::Path.

6.56.2.3 void kfoundation::SerializingStreamer::printToXmlStream (ostream & os) const

Prints the serialization of this object in XML format to the given stream.

#### **Parameters**

os The stream to print to.

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

- · SerializingStreamer.h
- · SerializingStreamer.cpp

# 6.57 kfoundation::SPtr < T > Class Template Reference

Static pointer, makes the pointed object immortal.

#include <kfoundation/Ptr.h>

#### **Public Member Functions**

• SPtr ()

Default constructor, creates a static NULL-pointer.

• SPtr (T \*obj)

Constructs a static pointer to the object at the given memory location.

SPtr (const Ptr < T > &obj)

Copy constructor.

## 6.57.1 Detailed Description

template<typename T>class kfoundation::SPtr< T>

Static pointer, makes the pointed object immortal.

When an object is pointed to by an SPtr it will be marked so that it never be deleted. This is necessary when defining static class fields.

A normal Ptr can be assigned to SPtr and viceversa, however the immortal properties of the object will never be changed.

## 6.57.2 Constructor & Destructor Documentation

```
6.57.2.1 template < typename T > kfoundation::SPtr < T >::SPtr ( T * obj )
```

Constructs a static pointer to the object at the given memory location.

This constructor is called when initializing object as follows.

```
public: static SPtr<MyClass> MY_STATIC_OBJECT = new MyClass();
```

#### **Parameters**

obj Should be pointing to a valid instance of ManagedObject or one of its subclasses, or NULL.

## 6.57.2.2 template<typename T> kfoundation::SPtr< T>::SPtr ( const Ptr< T> & obj )

Copy constructor.

If the given parameter is not static, it will be made static.

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

- · PtrDecl.h
- Ptr.h

# 6.58 kfoundation::StandardInputStreamAdapter Class Reference

Wraps around the given istream (C++ standard libraries) to be read as a a KFoundation input stream.

```
#include <kfoundation/StandardInputStreamAdapter.h>
```

## **Public Member Functions**

• StandardInputStreamAdapter (istream &stream)

Constructor, wraps the new instance around the given istream object.

kf\_int32\_t read (kf\_octet\_t \*buffer, const kf\_int32\_t nBytes)

Reads at most the given number of octets from the given buffer.

• int read ()

Reads a single octet.

• int peek ()

Reads a single octet without advancing.

kf\_int32\_t skip (kf\_int32\_t nBytes)

Skips the at most the given number of octets without reading them.

• bool isEof ()

Checks if the end of stream is reached.

• bool isMarkSupported ()

If returns true, mark() and reset() methods can be used.

· void mark ()

Marks the current position of the stream.

• void reset ()

Returns the stream position to where mark() was used last time.

• bool isBigEndian ()

Checks if the stream is big-endian.

## 6.58.1 Detailed Description

Wraps around the given istream (C++ standard libraries) to be read as a a KFoundation input stream.

# 6.58.2 Member Function Documentation

```
6.58.2.1 bool kfoundation::StandardInputStreamAdapter::isMarkSupported( ) [virtual]
```

If returns true, mark() and reset() methods can be used.

See Also

mark()

reset()

Implements kfoundation::InputStream.

```
6.58.2.2 void kfoundation::StandardInputStreamAdapter::mark( ) [virtual]
```

Marks the current position of the stream.

Use reset() to return to this position later and read the data again.

#### See Also

```
isMarkSupported()
reset()
```

Implements kfoundation::InputStream.

```
6.58.2.3 int kfoundation::StandardInputStreamAdapter::peek( ) [virtual]
```

Reads a single octet without advancing.

#### Returns

The value of read octet, or -1 if no data is available.

Implements kfoundation::InputStream.

```
6.58.2.4 kf_int32_t kfoundation::StandardInputStreamAdapter::read ( kf_octet_t * buffer, const kf_int32_t nOctets )

[virtual]
```

Reads at most the given number of octets from the given buffer.

Returns the actual number of octets read.

#### **Parameters**

buffer	The octets to read.
nOctets	Maximum number of octets to read.

## Returns

The actual number of octets read.

Implements kfoundation::InputStream.

```
6.58.2.5 int kfoundation::StandardInputStreamAdapter::read() [virtual]
```

Reads a single octet.

#### Returns

The value of read octet, or -1 if no data is available.

Implements kfoundation::InputStream.

```
6.58.2.6 void kfoundation::StandardInputStreamAdapter::reset() [virtual]
```

Returns the stream position to where mark() was used last time.

If mark() is never called, it will reset to the begining of the stream.

#### See Also

```
isMarkSupported()
mark()
```

Implements kfoundation::InputStream.

```
6.58.2.7 kf_int32_t kfoundation::StandardInputStreamAdapter::skip(kf_int32_t nOctets) [virtual]
```

Skips the at most the given number of octets without reading them.

If there are less number of octets in the stream, all available octets will be skipped.

#### **Parameters**

nOctets The desired number of octets to skeep.

#### Returns

The actual number of octets skipped.

Implements kfoundation::InputStream.

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

- StandardInputStreamAdapter.h
- StandardInputStreamAdapter.cpp

# 6.59 kfoundation::StandardOutputStreamAdapter Class Reference

KFoundation wrapper for C++ ostream.

#include <kfoundation/StandardOutputStreamAdapter.h>

## **Public Member Functions**

StandardOutputStreamAdapter (ostream &os)

Constructor, wraps the new object around a standard C++ ostream object.

• bool isBigEndian () const

Checks if the stream is big-endian.

void write (const kf\_octet\_t \*buffer, const kf\_int32\_t nBytes)

Writes the given number of octets of the given buffer to the stream.

• void write (kf\_octet\_t byte)

Writes a single octet to the stream.

void write (PPtr< InputStream > os)

Writes the available contents from the given input stream to this output stream.

• void close ()

Closes the stream.

## 6.59.1 Detailed Description

KFoundation wrapper for C++ ostream.

## 6.59.2 Member Function Documentation

**6.59.2.1 void kfoundation::StandardOutputStreamAdapter::close()** [virtual]

Closes the stream.

It will no longer be readable.

Implements kfoundation::OutputStream.

6.59.2.2 void kfoundation::StandardOutputStreamAdapter::write ( const kf\_octet\_t \* buffer, const kf\_int32\_t nOctets )

[virtual]

Writes the given number of octets of the given buffer to the stream.

## **Parameters**

buffer	The octets to write.
nOctets	Number of octets to write.

Implements kfoundation::OutputStream.

6.59.2.3 void kfoundation::StandardOutputStreamAdapter::write(kf\_octet\_t octet) [virtual]

Writes a single octet to the stream.

#### **Parameters**

octet	The octet to write

Implements kfoundation::OutputStream.

6.59.2.4 void kfoundation::StandardOutputStreamAdapter::write ( PPtr < InputStream > is ) [virtual]

Writes the available contents from the given input stream to this output stream.

#### **Parameters**

is	The stream to read from.

Implements kfoundation::OutputStream.

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

- · StandardOutputStreamAdapter.h
- StandardOutputStreamAdapter.cpp

# 6.60 kfoundation::Logger::Stream Class Reference

## Log stream.

#include <kfoundation/Logger.h>

## **Public Member Functions**

Stream (const Meta &meta, vector < Channel \* > &channels)

Constructor, do not use directly.

## 6.60.1 Detailed Description

Log stream.

Created when logger::log() method is called and distroyed once EL is passed to it.

## 6.60.2 Constructor & Destructor Documentation

6.60.2.1 kfoundation::Logger::Stream::Stream ( const Meta & meta, vector < Channel \* > & channels )

Constructor, do not use directly.

Use Logger::log() instread.

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

- · Logger.h
- · Logger.cpp

## 6.61 kfoundation::StreamDeserializer Class Reference

Interface to be implemented by any class that can be deserialized from stream.

```
#include <StreamDeserializer.h>
```

#### **Public Member Functions**

- virtual void deserialize (PPtr< ObjectToken > headToken)=0
   Implements object deserialization capability.
- void readFromXmlStream (PPtr< InputStream > stream)

Sets this object deserializing the given XML stream.

## 6.61.1 Detailed Description

Interface to be implemented by any class that can be deserialized from stream.

StreamDeserializer.h < kfoundation/StreamDeserializer.h >

## 6.61.2 Member Function Documentation

6.61.2.1 void kfoundation::StreamDeserializer::readFromXmlStream ( PPtr< InputStream > stream )

Sets this object deserializing the given XML stream.

**Parameters** 

stream | The XML stream to read the object from.

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

- · StreamDeserializer.h
- · StreamDeserializer.cpp

## 6.62 kfoundation::Streamer Class Reference

Base class for all classes that can print information about themeselves to a std::ostream.

```
#include <kfoundation/Streamer.h>
```

#### **Public Member Functions**

virtual void printToStream (ostream &stream) const =0

Implements compatibility with Streamer interface.

· virtual string toString () const

Converts the result of invocation of printToStream (ostream&) to a std::string object.

## 6.62.1 Detailed Description

Base class for all classes that can print information about themeselves to a std::ostream.

Subclasses should implement printToStream(ostream&) pure virtual function. This class defines and implements toString() function which internally invokes printToStream(ostream&) feeding it with std::stringstream.

#### See Also

```
operator<<(ostream&, const Streamer&)
operator+(const string&, const Streamer&)
operator+(const Streamer&, const string&)
```

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

- · Streamer.h
- · Streamer.cpp

# 6.63 kfoundation::StringInputStream Class Reference

Input stream to read from string.

```
#include <kfoundation/StringInputStream.h>
```

## **Public Member Functions**

• StringInputStream (const string &str)

Constructor, uses the given string as input.

∼StringInputStream ()

Deconstructor.

kf\_int32\_t read (kf\_octet\_t \*buffer, const kf\_int32\_t nBytes)

Reads at most the given number of octets from the given buffer.

• int read ()

Reads a single octet.

• int peek ()

Reads a single octet without advancing.

kf int32 t skip (kf int32 t bytes)

Skips the at most the given number of octets without reading them.

bool isEof ()

Checks if the end of stream is reached.

• bool isMarkSupported ()

If returns true, mark() and reset() methods can be used.

· void mark ()

Marks the current position of the stream.

• void reset ()

Returns the stream position to where mark() was used last time.

bool isBigEndian ()

Checks if the stream is big-endian.

## 6.63.1 Detailed Description

Input stream to read from string.

#### 6.63.2 Constructor & Destructor Documentation

6.63.2.1 kfoundation::StringInputStream::StringInputStream ( const string & str )

Constructor, uses the given string as input.

**Parameters** 

str | The string to read.

## 6.63.3 Member Function Documentation

```
6.63.3.1 bool kfoundation::StringInputStream::isMarkSupported() [virtual]
```

If returns true, mark() and reset() methods can be used.

See Also

mark()
reset()

Implements kfoundation::InputStream.

```
6.63.3.2 void kfoundation::StringInputStream::mark() [virtual]
```

Marks the current position of the stream.

Use reset() to return to this position later and read the data again.

See Also

isMarkSupported()
reset()

Implements kfoundation::InputStream.

**6.63.3.3** int kfoundation::StringInputStream::peek( ) [virtual]

Reads a single octet without advancing.

Returns

The value of read octet, or -1 if no data is available.

Implements kfoundation::InputStream.

```
6.63.3.4 kf_int32_t kfoundation::StringInputStream::read ( kf_octet_t * buffer, const kf_int32_t nOctets ) [virtual]
```

Reads at most the given number of octets from the given buffer.

Returns the actual number of octets read.

**Parameters** 

buffer	The octets to read.
nOctets	Maximum number of octets to read.

#### Returns

The actual number of octets read.

Implements kfoundation::InputStream.

```
6.63.3.5 int kfoundation::StringInputStream::read() [virtual]
```

Reads a single octet.

Returns

The value of read octet, or -1 if no data is available.

Implements kfoundation::InputStream.

```
6.63.3.6 void kfoundation::StringInputStream::reset() [virtual]
```

Returns the stream position to where mark() was used last time.

If mark() is never called, it will reset to the begining of the stream.

See Also

```
isMarkSupported()
mark()
```

Implements kfoundation::InputStream.

```
6.63.3.7 kf_int32_t kfoundation::StringInputStream::skip(kf_int32_t nOctets) [virtual]
```

Skips the at most the given number of octets without reading them.

If there are less number of octets in the stream, all available octets will be skipped.

**Parameters** 

nOctets The desired number of octets to skeep.

#### Returns

The actual number of octets skipped.

Implements kfoundation::InputStream.

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

- · StringInputStream.h
- StringInputStream.cpp

# 6.64 kfoundation::System Class Reference

Provides a cross-platform way to access sytsem features.

```
#include <kfoundation/System.h>
```

## **Public Types**

```
    enum operating_system_t {
        LINUX, FREE_BSD, SOLARIS, APPLE_OS_X,
        MACH, UNSUPPORTED }
```

Operating system type.

#### Static Public Member Functions

static PPtr< Path > getExePath ()

Returns executable path for the current process.

static Ptr< Path > getCurrentWorkingDirectory ()

Returns the current working directory.

• static bool isBigEndian ()

Used to check if the current platform is big-endian.

static string demangle (string str)

Demangles a C++ ABI symbol into a human readable one.

• static string resolveSymbolName (void \*ptr)

Given pointer to a symbol, returns its name.

static string getLastSystemError ()

Returns the human-readable description of the last system error as indicated by errno variable.

static operating\_system\_t getOperatingSystem ()

Return type of the undelying operating system.

• static Logger & getLogger ()

Returns the default system logger.

static void sleep (const int msecs)

Causes the current threed to sleep for the given number of miliseconds.

• static int exec (const char \*command, char \*\*args, int argc)

Executes the given command with the given set of arguments.

• static MasterMemoryManager & getMasterMemoryManager ()

Returns reference to the master memory manager.

static kf\_int64\_t getCurrentTimeInMiliseconds ()

Returns the current time in miliseconds from the standard origin time.

static int getPid ()

Returns the process id for this process.

## 6.64.1 Detailed Description

Provides a cross-platform way to access sytsem features.

It cannot be instantiated.

#### 6.64.2 Member Enumeration Documentation

6.64.2.1 enum kfoundation::System::operating system t

Operating system type.

#### **Enumerator**

LINUX Linux.

FREE BSD Free BSD.

SOLARIS Solaris.

APPLE\_OS\_X Apple Mac OS X.

MACH Mach.

UNSUPPORTED Other / Unsupported.

#### 6.64.3 Member Function Documentation

**6.64.3.1** int kfoundation::System::exec ( const char \* command, char \*\* args, int argc ) [static]

Executes the given command with the given set of arguments.

#### **Parameters**

command	The command to be executed.
args	An array of C strings, each containing a signle argument.
argc	The number of elements in args.

6.64.3.2 kf\_int64\_t kfoundation::System::getCurrentTimeInMiliseconds( ) [static]

Returns the current time in miliseconds from the standard origin time.

The origin time is usually midnigh January 1, 1970 UTC.

**6.64.3.3 PPtr**< **Path** > **kfoundation::System::getExePath()** [static]

Returns executable path for the current process.

Symlinks are resolved internally.

6.64.3.4 void kfoundation::System::sleep (const int msecs) [static]

Causes the current threed to sleep for the given number of miliseconds.

Most platforms provide sleep() which works with 1-second resolution, and usleep() which works with 1-microsecond resolution. System::sleep() provides milisecond resolution which makes sense in most devices, and has the same interface as Java.System.sleep(). Can be used in conjunction with System::getCurrentTimeInMiliseconds() without the fuss of timeval struct.

#### **Parameters**

msecs	The number of miliseconds to sleep.	
-------	-------------------------------------	--

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

- · System.h
- · System.cpp

## 6.65 kfoundation::TextToken Class Reference

Represents a text body (CDATA) in the parsed stream.

```
#include <kfoundation/ObjectStreamReader.h>
```

## **Public Member Functions**

• TextToken (const CodeRange &range)

Constructor.

• type\_t getType () const

Returns the type of this token.

## **Static Public Attributes**

• static const type\_t TYPE = Token::TEXT

Type this token, that is Token::TEXT.

## **Additional Inherited Members**

## 6.65.1 Detailed Description

Represents a text body (CDATA) in the parsed stream.

## See Also

Token ObjectStreamReader

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

- · ObjectStreamReader.h
- · ObjectStreamReader.cpp

# 6.66 kfoundation::Thread Class Reference

An object-oriented, cross-platform abstraction for thread.

#include <kfoundation/Thread.h>

## **Public Member Functions**

Thread ()

Default constructor.

Thread (const string &name)

Constructs a named thread.

∼Thread ()

Deconstructor.

• virtual void run ()=0

The implementation of this method will be called once a new thread is started.

• void start ()

Creates a new thread, invoking the run() method.

• bool isRunning () const

Checks if this thread is running.

• const string & getName () const

Returns the name of this thread.

· bool isTheCurrentThread () const

Returns true if this method is invoked on the thread represented by this object.

#### Static Public Member Functions

static string getNameOfCurrentThread ()

Returns the name of the thread that this method is invoked on.

## 6.66.1 Detailed Description

An object-oriented, cross-platform abstraction for thread.

To use, implement the run() method and call start().

## 6.66.2 Constructor & Destructor Documentation

```
6.66.2.1 kfoundation::Thread::Thread()
```

Default constructor.

Assigns a default name to this thread as "KFoundation Thread N" where N is a incremental counter.

```
6.66.2.2 kfoundation::Thread::Thread ( const string & name )
```

Constructs a named thread.

See Also

```
getName()
getNameOfCurrentThread()
```

## 6.66.3 Member Function Documentation

6.66.3.1 bool kfoundation::Thread::isRunning ( ) const

Checks if this thread is running.

A thead is in running state after the start() method is called, and before run() method ends.

```
6.66.3.2 bool kfoundation::Thread::isTheCurrentThread ( ) const
```

Returns true if this method is invoked on the thread represented by this object.

If invoked from another thread, returns false.

```
6.66.3.3 kfoundation::Thread::run() [pure virtual]
```

The implementation of this method will be called once a new thread is started.

When this method exits, the thread is considered to be ended, and it will be destroyed.

```
6.66.3.4 void kfoundation::Thread::start ( )
```

Creates a new thread, invoking the run() method.

This object is retained once the thread is started and released once it is ended.

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

- · Thread.h
- · Thread.cpp

## 6.67 kfoundation::Timer Class Reference

Utility class to measure execution time of a code fragment.

```
#include <kfoundation/Timer.h>
```

## **Public Member Functions**

• Timer ()

Default constructor.

• Timer (string name)

Constructor, creates a named timer.

• void start ()

Starts measuring.

• bool isStarted () const

Checks if the timer is started.

· double get () const

Returns the elapsed time since start() is called.

double getCpuTime () const

Returns amount of CPU time consumed by this process since start() is called.

• virtual void printToStream (ostream &os) const

Implements compatibility with Streamer interface.

## 6.67.1 Detailed Description

Utility class to measure execution time of a code fragment.

It keeps elapsed time and consumed CPU time. Usage:

```
Timer t;
t.start();
... do whatever you want to measure here ...
LOG << t << EL;</pre>
```

If you want to store the elapsed time to use it later, use get() method.

```
double duration = t.get();
```

To reset an already started timer, call start() method again.

#### 6.67.2 Member Function Documentation

```
6.67.2.1 double kfoundation::Timer::getCpuTime ( ) const
```

Returns amount of CPU time consumed by this process since start() is called.

This might be bigger or smaller than get() depending of the amount of CPU time and number of cores specified by OS to execute the process.

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

- Timer.h
- Timer.cpp

## 6.68 kfoundation::Token Class Reference

Represents a token in a stream.

```
#include <kfoundation/ObjectStreamReader.h>
```

#### **Public Member Functions**

• Token (const CodeRange &cr)

Constructor.

virtual type\_t getType () const =0

Returns the type of this token.

bool is (const type\_t &t) const

Checks if the type of this token matches the given argument.

void validateType (const type\_t &t) const

Checks of the type of this token matches the given argument, and if not, produces a ParseException explaning the problem.

PPtr< ObjectToken > asObject ()

Validates if this token represents an object.

PPtr< EndObjectToken > asEndObject ()

Validates if this token represents end of an object.

• PPtr< AttributeToken > asAttribute ()

Validates if this token represents an attribute.

• PPtr< CollectionToken > asCollection ()

Validates if this token represents begining of a collection.

PPtr< EndCollectionToken > asEndCollection ()

Validates if this token represents end of a collection.

## Static Public Member Functions

• static string toString (const type\_t &t)

Returns a string corresponding the given parameter.

## **Public Attributes**

const CodeRange codeRange

CodeRange marking begining and end of this token.

## **Static Public Attributes**

static const SPtr < Token > END\_STREAM\_TOKEN = new EndStreamItem()
 End stream token.

## 6.68.1 Detailed Description

Represents a token in a stream.

This is an abstract class. The actual object might be of any of the following types:

- ObjectToken
- AttributeToken
- EndObjectToken
- CollectionToken
- EndCollectionToken
- TextToken

Use getType() method to determine the type and cast accordingly. Most often this object is used in predictive parsing manner. For example:

```
void deserialize(PPtr<ObjectToken> headToken) {
   headToken->validateClass("MyClass");
   Ptr<Token> token = headToken->next();
   token->validateType(Token::ATTRIBUTE);
   _name = token.AS(AttributeToken)->validateName("name")->getValue();
   token->next()->validateType(END_OBJECT);
}
```

Conditional statements can be added if desired:

validateXXX() methods cause an expection to be thrown if the given argument does not match the current token. The exception message will include a code location that helps the user to understand the problem.

# See Also

ObjectDeserializer

## 6.68.2 Constructor & Destructor Documentation

6.68.2.1 kfoundation::Token::Token ( const CodeRange & cr )

Constructor.

**Parameters** 

cr The range marking the begining and the end of input containing this token.

## 6.68.3 Member Function Documentation

6.68.3.1 PPtr < AttributeToken > kfoundation::Token::asAttribute ( )

Validates if this token represents an attribute.

If so casts itself into AttributeToken, otherwise throws a ParseException.

Returns

Pointer to this object casted into AttributeToken.

#### **Exceptions**

ParseException | if the type of this token is not AttributeToken.

## 6.68.3.2 PPtr < CollectionToken > kfoundation::Token::asCollection ( )

Validates if this token represents begining of a collection.

If so casts itself into CollectionToken, otherwise throws a ParseException.

Returns

Pointer to this object casted into CollectionToken.

# Exceptions

ParseException if the type of this token is not CollectionToken.

## 6.68.3.3 PPtr < EndCollectionToken > kfoundation::Token::asEndCollection ( )

Validates if this token represents end of a collection.

If so casts itself into EndCollectionToken, otherwise throws a ParseException.

Returns

Pointer to this object casted into EndCollectionToken.

## **Exceptions**

ParseException if the type of this token is not EndCollectionToken.

## 6.68.3.4 PPtr < EndObjectToken > kfoundation::Token::asEndObject ( )

Validates if this token represents end of an object.

If so casts itself into EndObjectToken, otherwise throws a ParseException.

#### Returns

Pointer to this object casted into EndObjectToken.

## **Exceptions**

ParseException if the type of this token is not EndObjectToken.

6.68.3.5 PPtr < ObjectToken > kfoundation::Token::asObject ( )

Validates if this token represents an object.

If so casts itself into ObjectToken, otherwise throws a ParseException.

Returns

Pointer to this object casted into ObjectToken.

#### **Exceptions**

ParseException | if the type of this token is not ObjectToken.

## 6.68.4 Member Data Documentation

6.68.4.1 const CodeRange kfoundation::Token::codeRange

CodeRange marking begining and end of this token.

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

- · ObjectStreamReader.h
- ObjectStreamReader.cpp

# 6.69 kfoundation::Tuple Class Reference

Represents a point in n-dimensional space.

#include <kfoundation/Tuple.h>

## **Public Member Functions**

• Tuple ()

Default constructor, creates a tuple of size 0.

Tuple (kf\_int8\_t size)

Constructor, creates a tuple of the given size.

• Tuple (const Tuple &other)

Copy constructor.

kf int8 t getSize () const

Returns the size of this tuple.

kf\_int32\_t & at (const kf\_int8\_t index)

Returns the value of the element at the given index.

kf\_int32\_t at (const kf\_int8\_t index) const

Returns the value of the element at the given index.

void set (const Tuple &other)

Setter, sets the elements of this tuple to the ones of the given parameter.

bool equals (const Tuple & other) const

Checks if this tuple equals to the given parameter.

kf\_int32\_t sumAll () const

Returns the sum of all the elements.

kf\_int64\_t productAll () const

Returns the product of all elements.

Tuple max (const Tuple &other) const

Returns the value of the element with the highest value.

Tuple min (const Tuple &other) const

Returns the value of the element with the lowest value.

• Tuple negate () const

Negates all the elements.

• Tuple operator- () const

Additive inverse operator.

• Tuple operator+ (const Tuple &other) const

Addition operator.

• Tuple operator- (const Tuple &other) const

Substraction operator.

• Tuple operator+ (const int n) const

Adds the given scalar to all elements of this tuple.

• Tuple operator- (const int n) const

Substracts the given scalar from all elements of this tuple.

Tuple operator\* (const int n) const

Multiplies all elements of this tuple by the given number.

• Tuple operator\* (const Tuple &other) const

Multiplies all elements of this tuple by the given number.

Tuple operator/ (const int n) const

Divides all elements of this tuple by the given number.

• Tuple operator/ (const Tuple &other) const

Divides each element of this tuple by corresponding element of the given tuple.

Tuple operator% (const Tuple &other) const

Calculates remainder of each element of this tuple divided by corresponding element of the given tuple.

• bool operator== (const Tuple &other) const

Equality operator.

void printToStream (ostream &os) const

Implements compatibility with Streamer interface.

## **Static Public Member Functions**

static Tuple one (kf\_int8\_t size)

Returns a unity tuple of the given size.

static Tuple zero (kf\_int8\_t size)

Returns a zero tuple of the given size.

# 6.69.1 Detailed Description

Represents a point in n-dimensional space.

n can be a number between 0 to 4. Tuple1D, Tuple2D, and Tuple3D are specializations provided for convenience.

144	Class Documentation
6.69.2 Member Function Documentation	
6.69.2.1 Tuple kfoundation::Tuple::negate ( ) const	
Negates all the elements.	
Returns  A new tuple with each of its element being the inverse of that of this one's.	
6.69.2.2 Tuple kfoundation::Tuple::operator% ( const Tuple & other ) const	
Calculates remainder of each element of this tuple divided by corresponding element of the Parameters	ne given tuple.
other The tuple to divide to.	
Returns  The resulting tuple.	
6.69.2.3 Tuple kfoundation::Tuple::operator* ( const int $n$ ) const	
Multiplies all elements of this tuple by the given number.	
Parameters	
other The scalar to multiply.	
Returns  The resulting tuple.	
6.69.2.4 Tuple kfoundation::Tuple::operator* ( const Tuple & other ) const	
Multiplies all elements of this tuple by the given number.  Parameters	
other The scalar to multiply.	
Returns  The resulting tuple.	
6.69.2.5 Tuple kfoundation::Tuple::operator+ ( const Tuple & other ) const	
Addition operator.	

**Parameters** 

other	The tuple to add to.
-------	----------------------

Returns

The resulting tuple.

6.69.2.6 Tuple kfoundation::Tuple::operator+ ( const int n ) const

Adds the given scalar to all elements of this tuple.

**Parameters** 

other	The scalar to add.

Returns

The resulting tuple.

6.69.2.7 Tuple kfoundation::Tuple::operator-( ) const

Additive inverse operator.

Returns

A new tuple with each of its element being the inverse of that of this one's.

6.69.2.8 Tuple kfoundation::Tuple::operator- ( const Tuple & other ) const

Substraction operator.

Parameters

other	The tuple to substract.

Returns

The resulting tuple.

6.69.2.9 Tuple kfoundation::Tuple::operator- ( const int n ) const

Substracts the given scalar from all elements of this tuple.

**Parameters** 

other The scalar to substract.
--------------------------------

Returns

The resulting tuple.

6.69.2.10 Tuple kfoundation::Tuple::operator/ ( const int n ) const

Divides all elements of this tuple by the given number.

#### **Parameters**

other	The scalar to divide to.
-------	--------------------------

## Returns

The resulting tuple.

6.69.2.11 Tuple kfoundation::Tuple::operator/ ( const Tuple & other ) const

Divides each element of this tuple by corresponding element of the given tuple.

## **Parameters**

other	The tuple to divide to.

#### Returns

The resulting tuple.

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

- Tuple.h
- Tuple.cpp

# 6.70 kfoundation::Tuple1D Class Reference

1-dimensional specialization of Tuple.

```
#include <kfoundation/Tuple.h>
```

# **Public Member Functions**

• Tuple1D ()

Default constructor, creates a zero 1D tuple.

• Tuple1D (kf\_int32\_t x)

Constructor, assigns the given parameter to the solve element of this tuple.

Tuple & set (kf\_int32\_t x)

Setter, sets the only element of this tuple to the given value.

kf\_int32\_t get () const

Getter, returns the value of the sole element of this tuple.

• operator kf\_int32\_t () const

Cast operator, casts this tuple to scalar of type kf\_int32\_t.

## **Static Public Attributes**

static Tuple1D ONE

Unity, (1)

static Tuple1D ZERO

Zero, (0)

## **Additional Inherited Members**

## 6.70.1 Detailed Description

1-dimensional specialization of Tuple.

## 6.70.2 Constructor & Destructor Documentation

6.70.2.1 kfoundation::Tuple1D::Tuple1D ( kf\_int32\_t x )

Constructor, assigns the given parameter to the solve element of this tuple.

**Parameters** 

x The value to be assigned.

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

- Tuple.h
- · Tuple.cpp

# 6.71 kfoundation::Tuple2D Class Reference

2-dimensional specialization of Tuple.

#include <kfoundation/Tuple.h>

# **Public Member Functions**

• Tuple2D ()

Defautl constructor, creates a zero 2-dimensional tuple.

• Tuple2D (int x, int y)

Constructor, assigns the elements of the new tuple according to the given parameters.

• Tuple & set (int x, int y)

Setter, sets the elements of this tuple according to the given parameters.

• Tuple & setX (int x)

Setter, sets the first element of this tuple to the given parameter.

Tuple & setY (int y)

Setter, sets the second element of this tuple to the given parameter.

• int getX () const

Getter, returns the value of the first element of this tuple.

int getY () const

Getter, returns the value of the second element of this tuple.

## **Static Public Attributes**

static Tuple2D ONE

Unity, (1, 1)

• static Tuple2D ZERO

Zero, (0, 0)

## **Additional Inherited Members**

## 6.71.1 Detailed Description

2-dimensional specialization of Tuple.

#### 6.71.2 Constructor & Destructor Documentation

6.71.2.1 kfoundation::Tuple2D::Tuple2D (int x, int y)

Constructor, assigns the elements of the new tuple according to the given parameters.

#### **Parameters**

X	Value for the first element.
У	Value for the second element.

## 6.71.3 Member Function Documentation

6.71.3.1 Tuple & kfoundation::Tuple2D::set (int x, int y)

Setter, sets the elements of this tuple according to the given parameters.

#### **Parameters**

X	Value for the first element.
У	Value for the second element.

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

- Tuple.h
- · Tuple.cpp

# 6.72 kfoundation::Tuple3D Class Reference

3-dimensional specialization of Tuple.

```
#include <kfoundation/Tuple.h>
```

## **Public Member Functions**

• Tuple3D ()

Default constructor, creates a zero 3-dimensional tuple.

• Tuple3D (int x, int y, int z)

Constructor, assigns the elements of the new tuple according to the given parameters.

• Tuple & set (int x, int y, int z)

Setter, sets the elements of this tuple according to the given parameters.

Tuple & setX (int x)

Setter, sets the first element of this tuple to the given parameter.

Tuple & setY (int y)

Setter, sets the second element of this tuple to the given parameter.

Tuple & setZ (int z)

Setter, sets the third element of this tuple to the given parameter.

• int getX () const

Getter, returns the value of the first element of this tuple.

• int getY () const

Getter, returns the value of the second element of this tuple.

int getZ () const

Getter, returns the value of the third element of this tuple.

## **Static Public Attributes**

static Tuple3D ONE

Unity, (1, 1, 1)

static Tuple3D ZERO

Zeor, (0, 0, 0)

#### **Additional Inherited Members**

## 6.72.1 Detailed Description

3-dimensional specialization of Tuple.

## 6.72.2 Constructor & Destructor Documentation

6.72.2.1 kfoundation::Tuple3D::Tuple3D ( int x, int y, int z )

Constructor, assigns the elements of the new tuple according to the given parameters.

## **Parameters**

X	The value for the first element.
У	The value for the second element.
Z	The value for the third element.

## 6.72.3 Member Function Documentation

6.72.3.1 Tuple & kfoundation::Tuple3D::set (int x, int y, int z)

Setter, sets the elements of this tuple according to the given parameters.

#### **Parameters**

X	The value for the first element.
у	The value for the second element.
Z	The value for the third element.

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

- Tuple.h
- · Tuple.cpp

# 6.73 kfoundation::UniChar Class Reference

Wrapper class for unicode character.

#include <kfoundation/UniChar.h>

## **Public Member Functions**

UniChar (const char value)

Constructor.

• UniChar (const wchar t value)

Constructor.

• wchar\_t get () const

Getter method.

void set (const wchar\_t value)

Setter method.

• unsigned short int getNumberOfUtf8Octets () const

Returns the number of octets needed to represent the value of this object in UTF-8.

unsigned short int writeUtf8 (kf octet t \*buffer) const

Writes the value of this object to to the given buffer as UTF-8.

unsigned short int readUtf8 (kf\_octet\_t \*buffer)

Reads a unicode caracter encoded in UTF-8 from the given buffer.

unsigned short int readUtf8 (Ptr< InputStream > stream)

Reads a unicode character encoded in UTF-8 from the given stream.

· bool isLowerCase () const

Checks if this is a lowercase character.

• bool isUpperCase () const

Checks if this is an uppercase character.

• bool isNumeric () const

Checks if this is numeric character.

bool isAlphabet () const

Checks if this is an alphabetic character.

• bool isAlphanumeric () const

Checks if this is an alphanumeric character.

bool isWhiteSpace () const

Checks if this is a white space character.

void toLowerCase ()

Converts this character to lowercase.

void toUpperCase ()

Converts this character to uppercase.

Ptr< UniChar > duplicateToLowerCase () const

Creates a new UniChar obejct representing the lowercase equivalant of this object.

• Ptr< UniChar > duplicateToUpperCase () const

Creates a new UniChar object representing the uppercase equivalant of this object.

• Ptr< UniChar > duplicateUniChar () const

Creates a copy of this object.

void printToStream (ostream &os) const

Implements compatibility with Streamer interface.

## **Static Public Member Functions**

static unsigned short int getNumberOfUtf8Octets (const wchar\_t ch)

Returns the number of octets necessary to represent the given character in UTF-8 format.

• static unsigned short int writeUtf8 (const wchar\_t ch, kf\_octet\_t \*buffer)

Writes the given character onto the given buffer with UTF-8 encoding.

static unsigned short int readUtf8 (const unsigned char \*buffer, wchar\_t &output)

Reads a UTF-8 encoded character from the given input and sets the given output accordingly.

 static unsigned short int readUtf8 (const Ptr< InputStream > &stream, wchar\_t &output, kf\_octet\_t \*read-Octets)

Reads one UTF-8 encoded character from the given stream and assigns the given output accordingly.

static bool isLowerCase (const wchar t ch)

Checks if the given parameter is a lowercase character.

static bool isUpperCase (const wchar\_t ch)

Checks if the given parameter is an uppercase character.

static bool isNumeric (const wchar\_t ch)

Checks if the given parameter is a numeric character.

static bool isAlpabet (const wchar\_t ch)

Checks if the given argument is an alphabetic character.

static bool isAlphanumeric (const wchar\_t ch)

Checks if the given parameter is an alphanumeric character.

static bool isWhiteSpace (const wchar\_t ch)

Checks if the given paramter is a white space.

static wchar\_t toLowerCase (const wchar\_t ch)

Converts the given parameter to lowercase.

static wchar\_t toUpperCase (const wchar\_t ch)

Converts the given parameter to uppercase.

## 6.73.1 Detailed Description

Wrapper class for unicode character.

Uses a 4-byte internal representaion.

#### 6.73.2 Constructor & Destructor Documentation

6.73.2.1 kfoundation::UniChar::UniChar ( const char value )

Constructor.

Assigns the internal value to the given parameter.

**Parameters** 

value The value to be assigned.

6.73.2.2 kfoundation::UniChar::UniChar ( const wchar\_t value )

Constructor.

Assigns the internal value to the given parameter.

**Parameters** 

value The value to be set.

## 6.73.3 Member Function Documentation

6.73.3.1 wchar\_t kfoundation::UniChar::get ( ) const

Getter method.

Returns the internal value.

6.73.3.2 unsigned short int kfoundation::UniChar::readUtf8 (  $kf\_octet\_t*buffer$  )

Reads a unicode caracter encoded in UTF-8 from the given buffer.

#### **Parameters**

buffer	The begining of the buffer to read the value from.
--------	--

## Returns

The number of octets read.

6.73.3.3 unsigned short int kfoundation::UniChar::readUtf8 ( Ptr < InputStream > stream )

Reads a unicode character encoded in UTF-8 from the given stream.

#### **Parameters**

stream	The stream to read the value from.

## Returns

The number of octets read.

6.73.3.4 unsigned short int kfoundation::UniChar::readUtf8 ( const unsigned char \* buffer, wchar\_t & output ) [static]

Reads a UTF-8 encoded character from the given input and sets the given output accordingly.

#### **Parameters**

input	The begining of the buffer to read from.
output	The output to write to.

## Returns

The number of octets read.

6.73.3.5 unsigned short int kfoundation::UniChar::readUtf8 ( const Ptr< InputStream > & stream, wchar\_t & output, kf\_octet\_t \* readBytes ) [static]

Reads one UTF-8 encoded character from the given stream and assigns the given output accordingly.

#### **Parameters**

stream	The stream to read from.
output	The output to write to.
readBytes	The read octets are written to this buffer, and I don't remember why.

## Returns

The number of read octets.

6.73.3.6 void kfoundation::UniChar::set ( const wchar\_t value )

Setter method.

Sets the internal value to the given parameter.

#### **Parameters**

value	The value to be set.
-------	----------------------

6.73.3.7 wchar\_t kfoundation::UniChar::toLowerCase ( const wchar\_t ch ) [static]

Converts the given parameter to lowercase.

#### **Parameters**

ch	The character to be converted.

#### Returns

The result of conversion.

6.73.3.8 wchar\_t kfoundation::UniChar::toUpperCase ( const wchar\_t ch ) [static]

Converts the given parameter to uppercase.

#### **Parameters**

ch	The character to be converted.

## Returns

The result of conversion.

6.73.3.9 unsigned short int kfoundation::UniChar::writeUtf8 ( kf\_octet\_t \* buffer ) const

Writes the value of this object to to the given buffer as UTF-8.

# **Parameters**

buffer	The begining of the buffer to write on.
	The beginning of the barrer to time on

## Returns

The number of octets written.

6.73.3.10 unsigned short int kfoundation::UniChar::writeUtf8 ( const wchar\_t ch, kf\_octet\_t \* buffer ) [static]

Writes the given character onto the given buffer with UTF-8 encoding.

## Parameters

ch	The character to write.
buffer	The begining of the buffer to write on.

## Returns

The number of octets written.

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

- · UniChar.h
- · UniChar.cpp

# 6.74 kfoundation::XmlObjectStreamReader Class Reference

ObjectStreamReader to deserialize XML streams.

#include <kfoundation/XmlObjectStreamReader.h>

## **Public Member Functions**

XmlObjectStreamReader (PPtr< InputStream > input)

Constructor.

∼XmlObjectStreamReader ()

Deconstructor.

• Ptr< Token > next () throw (ParseException)

Returns the next token in the stream.

## 6.74.1 Detailed Description

ObjectStreamReader to deserialize XML streams.

See Also

ObjectStreamReader StreamDeserializer

#### 6.74.2 Constructor & Destructor Documentation

 $\textbf{6.74.2.1} \quad \textbf{kfoundation::XmlObjectStreamReader::XmlObjectStreamReader (} \ \ \textbf{PPtr} < \textbf{InputStream} > \textit{input} \ \ \textbf{)}$ 

Constructor.

**Parameters** 

*input* A stream containing XML representation of an object.

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

- · XmlObjectStreamReader.h
- · XmlObjectStreamReader.cpp

# Index

$\sim$ ManagedArray	kfoundation::BufferOutputStream, 34
kfoundation::ManagedArray, 75	
$\sim$ ManagedObject	С
kfoundation::ManagedObject, 78	kfoundation::InternetAddress, 57
~ObjectPoolMemoryManager	cast
kfoundation::ObjectPoolMemoryManager, 87	kfoundation::Ptr, 117
∼Ptr	changeExtension
kfoundation::Ptr, 116	kfoundation::Path, 99
	Channel
A	kfoundation::Logger::Channel, 35
kfoundation::InternetAddress, 57	checkLevel
APPLE_OS_X	kfoundation::Logger::Channel, 36
kfoundation::System, 135	checkName
AS	
Memory Management, 18	kfoundation::Logger::Channel, 36
add	class_t
kfoundation::NumericVector, 84	kfoundation::InternetAddress, 57
addChannel	close
kfoundation::Logger, 71	kfoundation::BufferOutputStream, 34
Array	kfoundation::FileOutputStream, 50
kfoundation::Array, 26	kfoundation::InternetOutputStream, 64
asAttribute	kfoundation::OutputStream, 94
kfoundation::Token, 141	kfoundation::StandardOutputStreamAdapter, 129
asCollection	CodeRange
kfoundation::Token, 141	kfoundation::CodeRange, 40
asEndCollection	codeRange
kfoundation::Token, 141	kfoundation::Token, 142
asEndObject	collection
kfoundation::Token, 141	kfoundation::ObjectSerializer, 90
asObject	component_t
kfoundation::Token, 142	kfoundation::Direction, 42
	connect
at kfoundation::Array, 26	kfoundation::InternetOutputStream, 64
• •	contains
kfoundation::ManagedArray, 75	kfoundation::Array, 27
В	copyWithPort
kfoundation::InternetAddress, 57	kfoundation::InternetAddress, 58
BACK	
	D
kfoundation::Direction, 42	kfoundation::InternetAddress, 58
bind	DECIMAL
kfoundation::InternetInputStream, 60	
block	kfoundation::LongInt, 73
kfoundation::Condition, 41	DUMP
Bool	kfoundation::ObjectSerializer, 90
kfoundation::Bool, 30	del
border	kfoundation::Ptr, 117
kfoundation::Range, 121	DirectionIterator
BufferInputStream	kfoundation::DirectionIterator, 43
kfoundation::BufferInputStream, 31	Double
BufferOutputStream	kfoundation::Double, 44

E kfoundation::InternetAddress, 58	kfoundation::ObjectPoolMemoryManager, 87 kfoundation::RefCountMemoryManager, 123
EQUAL	modification of countries y manager, 120
Macros and Typedefs, 16	HEXADECIMAL
ERR	kfoundation::LongInt, 73
kfoundation::Logger, 71	hasExtention
encoding_t	kfoundation::Path, 99
kfoundation::LongInt, 73	IOException
endCollection	kfoundation::IOException, 67
kfoundation::ObjectSerializer, 91	IS_NULL
endObject	Macros and Typedefs, 16
kfoundation::ObjectSerializer, 91 Exceptions, 17	ISA
exec	Memory Management, 18
kfoundation::System, 135	indexOf
Modridationoyotom, 100	kfoundation::Array, 27
FORTH	kfoundation::ManagedArray, 76
kfoundation::Direction, 42	IndexOutOfBoundException
FREE_BSD	kfoundation::IndexOutOfBoundException, 52
kfoundation::System, 135	Input/Output, 19
FileInputStream	insert
kfoundation::FileInputStream, 47	kfoundation::Array, 27, 28
FileOutputStream	kfoundation::ManagedArray, 76
kfoundation::FileOutputStream, 50	Int
finalize	kfoundation::Int, 55
kfoundation::PoolObject, 100	InternetAddress kfoundation::InternetAddress, 58
GREATER	InternetInputStream
Macros and Typedefs, 16	kfoundation::InternetInputStream, 60
get	InternetOutputStream
kfoundation::Bool, 30	kfoundation::InternetOutputStream, 64
kfoundation::Double, 44	InvalidFormatException
kfoundation::Int, 56	kfoundation::InvalidFormatException, 66
kfoundation::LongInt, 73	isAbsolute
kfoundation::UniChar, 151	kfoundation::Path, 99
getBegin	isBound
kfoundation::ParseException, 97	kfoundation::InternetInputStream, 60
getChannelByName	isEmpty
kfoundation::Logger, 71	kfoundation::Range, 121
getCpuTime	isLocked
kfoundation::Timer, 139	kfoundation::FileOutputStream, 50
getCurrentTimeInMiliseconds	isMarkSupported
kfoundation::System, 135	kfoundation::BufferInputStream, 31
getData kfoundation::BufferOutputStream, 34	kfoundation::FileInputStream, 47 kfoundation::InputStream, 53
getEnd	kfoundation::InternetInputStream, 60
kfoundation::ParseException, 97	kfoundation::StandardInputStreamAdapter, 126
getExePath	kfoundation::StringInputStream, 132
kfoundation::System, 135	isRunning
getMessage	kfoundation::Thread, 137
kfoundation::KFException, 68	isSilent
getRetainCount	kfoundation::Logger::Channel, 36
kfoundation::Ptr, 117	isSpace
getTable	kfoundation::PredictiveParserBase, 104
kfoundation::MemoryManager, 80	isTheCurrentThread
kfoundation::ObjectPoolMemoryManager, 87	kfoundation::Thread, 137
kfoundation::RefCountMemoryManager, 123	isValid
getTableSize	kfoundation::Ptr, 117
kfoundation::MemoryManager, 80	isValidIdentifierBeginChar

kfoundation::PredictiveParserBase, 104 isValidIdentifierChar	kfoundation::AttributeToken, 28 kfoundation::Bool, 29
kfoundation::PredictiveParserBase, 105	Bool, 30
isa	get, 30
kfoundation::Ptr, 117	set, 30
	toString, 30
JSON	kfoundation::BufferInputStream, 30
kfoundation::ObjectSerializer, 90	BufferInputStream, 31
WE EVENE	isMarkSupported, 31
KF_EXPORT	mark, 31
Macros and Typedefs, 16	peek, 32
KF_NOP	read, 32
Macros and Typedefs, 16	reset, 32
KF_SYNCHRONIZED	skip, 32
Mutithreading, 20	kfoundation::BufferOutputStream, 33
KFException	BufferOutputStream, 34
kfoundation::KFException, 68	close, 34
kf_comparison_t	getData, 34
Macros and Typedefs, 16	write, 34
kfoundation::Direction	kfoundation::CodeLocation, 37
BACK, 42	
FORTH, 42	set, 38
NEUTRAL, 42	setByteIndex, 38
kfoundation::InternetAddress	setCharIndex, 38
A, 57	setCol, 39
B, 57	setLine, 39
C, 57	kfoundation::CodeRange, 39
D, 58	CodeRange, 40
É, 58	kfoundation::CollectionToken, 40
kfoundation::Logger	kfoundation::Condition, 41
ERR, 71	block, 41
L1, 71	release, 41
L2, 71	kfoundation::Direction, 41
L3, 71	component_t, 42
WRN, 71	set, 42
kfoundation::LongInt	kfoundation::DirectionIterator, 43
DECIMAL, 73	DirectionIterator, 43
HEXADECIMAL, 73	kfoundation::Double, 44
kfoundation::ObjectSerializer	Double, 44
DUMP, 90	get, 44
JSON, 90	parse, 44
XML, 90	set, 44
kfoundation::System	kfoundation::EndCollectionToken, 45
APPLE OS X, 135	kfoundation::EndObjectToken, 45
FREE_BSD, 135	kfoundation::FileInputStream, 46
	FileInputStream, 47
LINUX, 135	isMarkSupported, 47
MACH, 135	mark, 47
SOLARIS, 135	peek, 48
UNSUPPORTED, 135	-
kfoundation::Array	read, 48
Array, 26	reset, 48
at, 26	skip, 48
contains, 27	kfoundation::FileOutputStream, 49
indexOf, 27	close, 50
insert, 27, 28	FileOutputStream, 50
pop, 28	isLocked, 50
push, 28	lock, 50
remove, 28	truncate, 51
kfoundation::Array< T >, 25	unlock, 51

write, 51	setFormat, 36
kfoundation::IOException, 66	setLevel, 37
IOException, 67	setSilent, 37
kfoundation::IndexOutOfBoundException, 52	kfoundation::Logger::Stream, 129
IndexOutOfBoundException, 52	Stream, 130
kfoundation::InputStream, 52	kfoundation::LongInt, 72
isMarkSupported, 53	encoding_t, 73
mark, 53	get, 73
peek, 53	LongInt, 73
read, 53, 54	parse, 73
reset, 54	set, 74
skip, 54	kfoundation::ManagedArray
kfoundation::Int, 55	∼ManagedArray, 75
get, 56	at, 75
Int, 55	indexOf, 76
parse, 56	insert, 76
set, 56	ManagedArray, 75
toHexString, 56	pop, 76
kfoundation::InternetAddress, 56	push, 77
class_t, 57	remove, 77
copyWithPort, 58	setSize, 77
InternetAddress, 58	kfoundation::ManagedArray< T >, 74
kfoundation::InternetInputStream, 58	kfoundation::ManagedObject, 77
bind, 60	$\sim$ ManagedObject, 78
InternetInputStream, 60	kfoundation::MasterMemoryManager, 78
isBound, 60	registerManager, 79
isMarkSupported, 60	unregisterManager, 79
listen, 60	updataTable, 79
mark, 62	kfoundation::MemoryException, 79
peek, 62	kfoundation::MemoryManager, 80
read, 62	getTable, 80
reset, 62	getTableSize, 80
skip, 63	release, 81
unbind, 63	remove, 81
kfoundation::InternetOutputStream, 63	kfoundation::Mutex, 81
close, 64	Mutex, 82
connect, 64	kfoundation::NullPointerException, 83
InternetOutputStream, 64	kfoundation::NumericVector
write, 65	add, 84
kfoundation::InvalidFormatException, 65	mul, 84
InvalidFormatException, 66	NumericVector, 84
kfoundation::InvalidPointerException, 66	parseInt, 84
kfoundation::KFException, 67	sub, 85
getMessage, 68	kfoundation::NumericVector $<$ T $>$ , 83
KFException, 68	kfoundation::ObjectDumpBuilderException, 85
serialize, 68	ObjectDumpBuilderException, 85
setName, 68	kfoundation::ObjectPoolMemoryManager
kfoundation::Logger, 68	$\sim$ ObjectPoolMemoryManager, 87
addChannel, 71	getTable, 87
getChannelByName, 71	getTableSize, 87
level_t, 71	ObjectPoolMemoryManager, 86
log, 71	release, 87
setLevel, 72	remove, 87
kfoundation::Logger::Channel, 35	${\it kfoundation::} Object Pool Memory Manager < T>, 86$
Channel, 35	kfoundation::ObjectRecord, 88
checkLevel, 36	kfoundation::ObjectSerializer, 88
checkName, 36	collection, 90
isSilent, 36	endCollection, 91

endObject, 91	toetChar 112
member, 91	testChar, 112
	testSequence, 112
null, 91	testSpace, 113
object, 91	kfoundation::ProximityIterator, 113
ObjectSerializer, 90	next, 114
output_type_t, 90	ProximityIterator, 113
kfoundation::ObjectStreamReader, 92	kfoundation::Ptr
kfoundation::ObjectToken, 92	∼Ptr, 116
validateClass, 93	cast, 117
kfoundation::OutOfMemoryException, 93	del, 117
OutOfMemoryException, 94	getRetainCount, 117
kfoundation::OutputStream, 94	isValid, 117
close, 94	isa, 117
write, 94, 96	operator*, 117
kfoundation::PPtr	operator->, 117
PPtr, 101	operator=, 117, 118
kfoundation::PPtr $<$ T $>$ , 100	Ptr, 116
kfoundation::ParseException, 96	release, 118
•	
getBegin, 97	replace, 118
getEnd, 97	retain, 118
ParseException, 97	trace, 119
kfoundation::Path, 98	untrace, 119
changeExtension, 99	kfoundation::Ptr $<$ T $>$ , 114
hasExtention, 99	kfoundation::Range, 119
isAbsolute, 99	border, 121
parent, 99	isEmpty, 121
Path, 99	Range, 121
printToStream, 99	kfoundation::Rangelterator, 121
kfoundation::PoolObject, 100	kfoundation::RefCountMemoryManager, 122
finalize, 100	getTable, 123
kfoundation::PredictiveParserBase, 102	getTableSize, 123
isSpace, 104	release, 123
isValidIdentifierBeginChar, 104	remove, 123
isValidIdentifierChar, 105	kfoundation::SPtr
PredictiveParserBase, 104	SPtr, 125
readAllAlphabet, 105	kfoundation::SPtr $<$ T $>$ , 125
•	
readAllAlphanumeric, 105	kfoundation::SerializingStreamer, 124
readAllBeforeChar, 105	printToJsonStream, 124
readAllBeforeCharSkipEscaped, 105	printToStream, 124
readAllBeforeNewLine, 106	printToXmlStream, 124
readAllBeforeSequence, 106	kfoundation::StandardInputStreamAdapter, 126
readAllBeforeSpace, 106	isMarkSupported, 126
readAllBeforeSpaceOrNewLine, 106	mark, 126
readAllNumeric, 107	peek, 127
readAlphabet, 107	read, 127
readAlphanumeric, 107	reset, 127
readAny, 107	skip, 128
readChar, 108	kfoundation::StandardOutputStreamAdapter, 128
readIdentifier, 108	close, 129
readIdentifierBeginChar, 108	write, 129
readIdentifierChar, 110	kfoundation::StreamDeserializer, 130
readNewLine, 110	readFromXmlStream, 130
readNumber, 110, 111	kfoundation::Streamer, 131
readNumeric, 111	kfoundation::StringInputStream, 131
readSequence, 111	isMarkSupported, 132
readSpace, 112	mark, 132
skipSpaces, 112	peek, 132
skipSpacesAndNewLines, 112	read, 133

reset, 133	kfoundation::System, 135
skip, 133	level_t
StringInputStream, 132	kfoundation::Logger, 71
kfoundation::System, 134	listen
exec, 135	kfoundation::InternetInputStream, 60
getCurrentTimeInMiliseconds, 135	lock
getExePath, 135	kfoundation::FileOutputStream, 50
operating_system_t, 135	log
sleep, 135	kfoundation::Logger, 71
kfoundation::TextToken, 136	LongInt
kfoundation::Thread, 136	kfoundation::LongInt, 73
isRunning, 137	KlouridationLongint, 75
isTheCurrentThread, 137	MACH
	kfoundation::System, 135
run, 138	Macros and Typedefs, 15
start, 138	
Thread, 137	EQUAL, 16
kfoundation::Timer, 138	GREATER, 16
getCpuTime, 139	IS_NULL, 16
kfoundation::Token, 139	KF_EXPORT, 16
asAttribute, 141	KF_NOP, 16
asCollection, 141	kf_comparison_t, 16
asEndCollection, 141	NOT_NULL, 16
asEndObject, 141	SMALLER, 16
asObject, 142	ManagedArray
codeRange, 142	kfoundation::ManagedArray, 75
Token, 141	mark
kfoundation::Tuple, 142	kfoundation::BufferInputStream, 31
negate, 144	kfoundation::FileInputStream, 47
operator*, 144	kfoundation::InputStream, 53
operator+, 144, 145	kfoundation::InternetInputStream, 62
operator-, 145	kfoundation::StandardInputStreamAdapter, 126
operator/, 145, 146	kfoundation::StringInputStream, 132
operator%, 144	member
kfoundation::Tuple1D, 146	kfoundation::ObjectSerializer, 91
Tuple1D, 147	Memory Management, 18
kfoundation::Tuple2D, 147	AS, 18
•	ISA, 18
set, 148	
Tuple2D, 148	mul
kfoundation::Tuple3D, 148	kfoundation::NumericVector, 84
set, 149	Mutex
Tuple3D, 149	kfoundation::Mutex, 82
kfoundation::UniChar, 149	Mutithreading, 20
get, 151	KF_SYNCHRONIZED, 20
readUtf8, 151, 153	AUGUSTOAL
set, 153	NEUTRAL
toLowerCase, 154	kfoundation::Direction, 42
toUpperCase, 154	NOT_NULL
UniChar, 151	Macros and Typedefs, 16
writeUtf8, 154	negate
kfoundation::XmlObjectStreamReader, 155	kfoundation::Tuple, 144
XmlObjectStreamReader, 155	next
-	kfoundation::ProximityIterator, 114
L1	null
kfoundation::Logger, 71	kfoundation::ObjectSerializer, 91
L2	NumericVector
kfoundation::Logger, 71	kfoundation::NumericVector, 84
L3	, -
kfoundation::Logger, 71	object
LINUX	kfoundation::ObjectSerializer, 91
	,

Object Serialization and Deserialization, 23 ObjectDumpBuilderException	kfoundation::SerializingStreamer, 124 printToXmlStream
kfoundation::ObjectDumpBuilderException, 85	kfoundation::SerializingStreamer, 124
ObjectPoolMemoryManager	ProximityIterator
kfoundation::ObjectPoolMemoryManager, 86	kfoundation::ProximityIterator, 113
ObjectSerializer	Ptr
kfoundation::ObjectSerializer, 90	kfoundation::Ptr, 116
operating_system_t kfoundation::System, 135	push kfoundation::Array, 28
	kfoundation::ManagedArray, 77
operator* kfoundation::Ptr, 117	KlouridationManagedArray, 77
kfoundation::Tuple, 144	Range
operator+	kfoundation::Range, 121
kfoundation::Tuple, 144, 145	Range Arithmatics, 24
operator-	read
kfoundation::Tuple, 145	kfoundation::BufferInputStream, 32
operator->	kfoundation::FileInputStream, 48
kfoundation::Ptr, 117	kfoundation::InputStream, 53, 54
operator/	kfoundation::InternetInputStream, 62
kfoundation::Tuple, 145, 146	kfoundation::StandardInputStreamAdapter, 127
operator=	kfoundation::StringInputStream, 133
kfoundation::Ptr, 117, 118	readAllAlphabet
operator%	kfoundation::PredictiveParserBase, 105
kfoundation::Tuple, 144	readAllAlphanumeric
OutOfMemoryException	kfoundation::PredictiveParserBase, 105
kfoundation::OutOfMemoryException, 94	readAllBeforeChar
output_type_t	kfoundation::PredictiveParserBase, 105
kfoundation::ObjectSerializer, 90	readAllBeforeCharSkipEscaped
ModridationObjectochalizor, 30	kfoundation::PredictiveParserBase, 105
PPtr	readAllBeforeNewLine
kfoundation::PPtr, 101	kfoundation::PredictiveParserBase, 106
parent	readAllBeforeSequence
kfoundation::Path, 99	kfoundation::PredictiveParserBase, 106
parse	readAllBeforeSpace
kfoundation::Double, 44	kfoundation::PredictiveParserBase, 106
kfoundation::Int, 56	readAllBeforeSpaceOrNewLine
kfoundation::LongInt, 73	kfoundation::PredictiveParserBase, 106
ParseException	readAllNumeric
kfoundation::ParseException, 97	kfoundation::PredictiveParserBase, 107
parseInt	readAlphabet
kfoundation::NumericVector, 84	kfoundation::PredictiveParserBase, 107
Path	readAlphanumeric
kfoundation::Path, 99	kfoundation::PredictiveParserBase, 107
peek	readAny
kfoundation::BufferInputStream, 32	kfoundation::PredictiveParserBase, 107
kfoundation::FileInputStream, 48	readChar
kfoundation::InputStream, 53	kfoundation::PredictiveParserBase, 108
kfoundation::InternetInputStream, 62	readFromXmlStream
kfoundation::StandardInputStreamAdapter, 127	kfoundation::StreamDeserializer, 130
kfoundation::StringInputStream, 132	readIdentifier
рор	kfoundation::PredictiveParserBase, 108
kfoundation::Array, 28	readIdentifierBeginChar
kfoundation::ManagedArray, 76	kfoundation::PredictiveParserBase, 108
PredictiveParserBase	readIdentifierChar
kfoundation::PredictiveParserBase, 104	kfoundation::PredictiveParserBase, 110
printToJsonStream	readNewLine
kfoundation::SerializingStreamer, 124	kfoundation::PredictiveParserBase, 110
printToStream	readNumber
kfoundation::Path, 99	kfoundation::PredictiveParserBase, 110, 111

readNumeric	kfoundation::CodeLocation, 39
kfoundation::PredictiveParserBase, 111	setFormat
readSequence	kfoundation::Logger::Channel, 36
kfoundation::PredictiveParserBase, 111	setLevel
readSpace	kfoundation::Logger, 72
kfoundation::PredictiveParserBase, 112	kfoundation::Logger::Channel, 37
readUtf8	setLine
kfoundation::UniChar, 151, 153	kfoundation::CodeLocation, 39
registerManager	setName
kfoundation::MasterMemoryManager, 79	kfoundation::KFException, 68
release	setSilent
kfoundation::Condition, 41	kfoundation::Logger::Channel, 37
kfoundation::MemoryManager, 81	setSize
kfoundation::ObjectPoolMemoryManager, 87	kfoundation::ManagedArray, 77
kfoundation::Ptr, 118	skip
kfoundation::RefCountMemoryManager, 123	kfoundation::BufferInputStream, 32
remove	kfoundation::FileInputStream, 48
kfoundation::Array, 28	kfoundation::InputStream, 54
kfoundation::ManagedArray, 77	kfoundation::InternetInputStream, 63
kfoundation::MemoryManager, 81	kfoundation::StandardInputStreamAdapter, 128
kfoundation::ObjectPoolMemoryManager, 87	kfoundation::StringInputStream, 133
kfoundation::RefCountMemoryManager, 123	skipSpaces
replace	kfoundation::PredictiveParserBase, 112
kfoundation::Ptr, 118	skipSpacesAndNewLines
reset	kfoundation::PredictiveParserBase, 112
kfoundation::BufferInputStream, 32	sleep
kfoundation::FileInputStream, 48	kfoundation::System, 135
kfoundation::InputStream, 54	start
kfoundation::InternetInputStream, 62	kfoundation::Thread, 138
kfoundation::StandardInputStreamAdapter, 127	Stream
kfoundation::StringInputStream, 133	kfoundation::Logger::Stream, 130
retain	StringInputStream
	kfoundation::StringInputStream, 132
kfoundation::Ptr, 118	sub
run kfoundation::Thread, 138	kfoundation::NumericVector, 85
RioundationTillead, 136	RiouridationNumeric vector, 65
SMALLER	testChar
Macros and Typedefs, 16	kfoundation::PredictiveParserBase, 112
SOLARIS	testSequence
kfoundation::System, 135	kfoundation::PredictiveParserBase, 112
SPtr	testSpace
kfoundation::SPtr, 125	kfoundation::PredictiveParserBase, 113
serialize	Thread
kfoundation::KFException, 68	kfoundation::Thread, 137
set	toHexString
kfoundation::Bool, 30	kfoundation::Int, 56
kfoundation::CodeLocation, 38	toLowerCase
kfoundation::Direction, 42	kfoundation::UniChar, 154
kfoundation::Double, 44	toString
kfoundation::Int, 56	
	kfoundation::Bool, 30
kfoundation::LongInt, 74	toUpperCase
kfoundation::Tuple2D, 148	kfoundation::UniChar, 154
kfoundation::Tuple3D, 149	Token
kfoundation::UniChar, 153	kfoundation::Token, 141
setByteIndex	trace
kfoundation::CodeLocation, 38	kfoundation::Ptr, 119
setCharIndex	truncate
kfoundation::CodeLocation, 38	kfoundation::FileOutputStream, 51
setCol	Tuple1D

```
kfoundation::Tuple1D, 147
Tuple2D
    kfoundation::Tuple2D, 148
Tuple3D
    kfoundation::Tuple3D, 149
UNSUPPORTED
    kfoundation::System, 135
unbind
    kfoundation::InternetInputStream, 63
UniChar
    kfoundation::UniChar, 151
unlock
    kfoundation::FileOutputStream, 51
unregisterManager
    kfoundation::MasterMemoryManager, 79
untrace
    kfoundation::Ptr, 119
updataTable
    kfoundation::MasterMemoryManager, 79
Utilities, 22
validateClass
    kfoundation::ObjectToken, 93
WRN
    kfoundation::Logger, 71
Wrappers and Containers, 21
write
    kfoundation::BufferOutputStream, 34
    kfoundation::FileOutputStream, 51
    kfoundation::InternetOutputStream, 65
    kfoundation::OutputStream, 94, 96
    kfoundation::StandardOutputStreamAdapter, 129
writeUtf8
    kfoundation::UniChar, 154
XML
    kfoundation::ObjectSerializer, 90
XmlObjectStreamReader
    kfoundation::XmlObjectStreamReader, 155
```