

KFoundation

v1.0.2

Generated by Doxygen 1.8.4

Fri Oct 30 2015 13:00:04

Contents

1	Overview	1
2	Module Index	5
2.1	Modules	5
3	Hierarchical Index	7
3.1	Class Hierarchy	7
4	Class Index	11
4.1	Class List	11
5	Module Documentation	15
5.1	Macros 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	Exceptions	17
5.2.1	Detailed Description	17
5.3	Memory 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/Output	19
5.4.1	Detailed Description	19
5.5	Mutithreading	20
5.5.1	Detailed Description	20
5.5.2	Macro Definition Documentation	20

5.5.2.1	KF_SYNCHRONIZED	20
5.6	Wrappers and Containers	21
5.6.1	Detailed Description	21
5.7	Utilities	22
5.7.1	Detailed Description	22
5.8	Object Serialization and Deserialization	23
5.8.1	Detailed Description	23
5.9	Range Arithmetics	24
5.9.1	Detailed Description	24
6	Class Documentation	25
6.1	kfoundation::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	27
6.1.3.1	at	27
6.1.3.2	at	27
6.1.3.3	contains	27
6.1.3.4	indexOf	27
6.1.3.5	indexOf	27
6.1.3.6	insert	28
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	29
6.2	kfoundation::AttributeToken Class Reference	29
6.2.1	Detailed Description	30
6.3	kfoundation::Bool Class Reference	30
6.3.1	Detailed Description	31
6.3.2	Constructor & Destructor Documentation	31
6.3.2.1	Bool	31
6.3.3	Member Function Documentation	31
6.3.3.1	get	31
6.3.3.2	set	31
6.3.3.3	toString	31
6.4	kfoundation::BufferInputStream Class Reference	31
6.4.1	Detailed Description	32
6.4.2	Constructor & Destructor Documentation	33

6.4.2.1	BufferInputStream	33
6.4.3	Member Function Documentation	34
6.4.3.1	isMarkSupported	34
6.4.3.2	mark	34
6.4.3.3	peek	34
6.4.3.4	read	34
6.4.3.5	read	35
6.4.3.6	reset	35
6.4.3.7	skip	35
6.5	kfoundation::BufferOutputStream Class Reference	35
6.5.1	Detailed Description	36
6.5.2	Constructor & Destructor Documentation	36
6.5.2.1	BufferOutputStream	36
6.5.3	Member Function Documentation	37
6.5.3.1	close	37
6.5.3.2	getData	37
6.5.3.3	write	37
6.5.3.4	write	37
6.5.3.5	write	37
6.6	kfoundation::Logger::Channel Class Reference	38
6.6.1	Detailed Description	38
6.6.2	Constructor & Destructor Documentation	38
6.6.2.1	Channel	38
6.6.2.2	Channel	38
6.6.3	Member Function Documentation	39
6.6.3.1	checkLevel	39
6.6.3.2	checkName	40
6.6.3.3	isSilent	40
6.6.3.4	setFormat	40
6.6.3.5	setLevel	40
6.6.3.6	setSilent	40
6.7	kfoundation::CodeLocation Class Reference	41
6.7.1	Detailed Description	42
6.7.2	Member Function Documentation	42
6.7.2.1	set	42
6.7.2.2	set	42
6.7.2.3	setByteIndex	42
6.7.2.4	setCharIndex	42
6.7.2.5	setCol	43
6.7.2.6	setLine	43

6.8	kfoundation::CodeRange Class Reference	43
6.8.1	Detailed Description	44
6.8.2	Constructor & Destructor Documentation	44
6.8.2.1	CodeRange	44
6.9	kfoundation::CollectionToken Class Reference	45
6.9.1	Detailed Description	45
6.10	kfoundation::Condition Class Reference	46
6.10.1	Detailed Description	46
6.10.2	Member Function Documentation	46
6.10.2.1	block	46
6.10.2.2	release	46
6.11	kfoundation::Direction Class Reference	47
6.11.1	Detailed Description	48
6.11.2	Member Enumeration Documentation	48
6.11.2.1	component_t	48
6.11.3	Member Function Documentation	48
6.11.3.1	set	48
6.12	kfoundation::DirectionIterator Class Reference	48
6.12.1	Detailed Description	49
6.12.2	Constructor & Destructor Documentation	49
6.12.2.1	DirectionIterator	49
6.13	kfoundation::Double Class Reference	50
6.13.1	Detailed Description	50
6.13.2	Constructor & Destructor Documentation	51
6.13.2.1	Double	51
6.13.3	Member Function Documentation	51
6.13.3.1	get	51
6.13.3.2	parse	51
6.13.3.3	set	51
6.14	kfoundation::EndCollectionToken Class Reference	51
6.14.1	Detailed Description	52
6.15	kfoundation::EndObjectToken Class Reference	53
6.15.1	Detailed Description	53
6.16	kfoundation::FileInputStream Class Reference	54
6.16.1	Detailed Description	55
6.16.2	Constructor & Destructor Documentation	55
6.16.2.1	FileInputStream	55
6.16.2.2	FileInputStream	55
6.16.3	Member Function Documentation	55
6.16.3.1	isMarkSupported	55

6.16.3.2	mark	56
6.16.3.3	peek	56
6.16.3.4	read	56
6.16.3.5	read	56
6.16.3.6	reset	56
6.16.3.7	skip	57
6.17	kfoundation::FileOutputStream Class Reference	57
6.17.1	Detailed Description	58
6.17.2	Constructor & Destructor Documentation	58
6.17.2.1	FileOutputStream	58
6.17.3	Member Function Documentation	58
6.17.3.1	close	58
6.17.3.2	isLocked	58
6.17.3.3	lock	59
6.17.3.4	truncate	59
6.17.3.5	unlock	59
6.17.3.6	write	59
6.17.3.7	write	60
6.17.3.8	write	60
6.18	kfoundation::IndexOutOfBoundsException Class Reference	60
6.18.1	Detailed Description	61
6.18.2	Constructor & Destructor Documentation	61
6.18.2.1	IndexOutOfBoundsException	61
6.19	kfoundation::InputStream Class Reference	62
6.19.1	Detailed Description	63
6.19.2	Member Function Documentation	63
6.19.2.1	isMarkSupported	63
6.19.2.2	mark	63
6.19.2.3	peek	63
6.19.2.4	read	63
6.19.2.5	read	64
6.19.2.6	reset	64
6.19.2.7	skip	64
6.20	kfoundation::Int Class Reference	65
6.20.1	Detailed Description	65
6.20.2	Constructor & Destructor Documentation	66
6.20.2.1	Int	66
6.20.2.2	Int	66
6.20.3	Member Function Documentation	66
6.20.3.1	get	66

6.20.3.2	parse	66
6.20.3.3	set	66
6.20.3.4	toHexString	66
6.21	kfoundation::InternetAddress Class Reference	67
6.21.1	Detailed Description	68
6.21.2	Member Enumeration Documentation	68
6.21.2.1	class_t	68
6.21.3	Constructor & Destructor Documentation	68
6.21.3.1	InternetAddress	68
6.21.3.2	InternetAddress	68
6.21.3.3	InternetAddress	68
6.21.4	Member Function Documentation	69
6.21.4.1	copyWithPort	69
6.22	kfoundation::InternetInputStream Class Reference	69
6.22.1	Detailed Description	70
6.22.2	Constructor & Destructor Documentation	70
6.22.2.1	InternetInputStream	70
6.22.3	Member Function Documentation	71
6.22.3.1	bind	71
6.22.3.2	isBound	71
6.22.3.3	isMarkSupported	71
6.22.3.4	listen	71
6.22.3.5	mark	71
6.22.3.6	peek	72
6.22.3.7	read	72
6.22.3.8	read	72
6.22.3.9	reset	72
6.22.3.10	skip	73
6.22.3.11	unbind	74
6.23	kfoundation::InternetOutputStream Class Reference	74
6.23.1	Detailed Description	75
6.23.2	Constructor & Destructor Documentation	75
6.23.2.1	InternetOutputStream	75
6.23.3	Member Function Documentation	75
6.23.3.1	close	75
6.23.3.2	connect	75
6.23.3.3	write	76
6.23.3.4	write	76
6.23.3.5	write	76
6.24	kfoundation::InvalidFormatException Class Reference	76

6.24.1 Detailed Description	77
6.24.2 Constructor & Destructor Documentation	77
6.24.2.1 InvalidFormatException	77
6.25 kfoundation::InvalidPointerException Class Reference	78
6.25.1 Detailed Description	78
6.26 kfoundation::IOException Class Reference	79
6.26.1 Detailed Description	79
6.26.2 Constructor & Destructor Documentation	79
6.26.2.1 IOException	79
6.27 kfoundation::KFException Class Reference	80
6.27.1 Detailed Description	80
6.27.2 Constructor & Destructor Documentation	81
6.27.2.1 KFException	81
6.27.2.2 KFException	81
6.27.3 Member Function Documentation	81
6.27.3.1 getMessage	81
6.27.3.2 serialize	81
6.27.3.3 setName	81
6.28 kfoundation::Logger Class Reference	81
6.28.1 Detailed Description	82
6.28.2 Member Enumeration Documentation	84
6.28.2.1 level_t	84
6.28.3 Member Function Documentation	84
6.28.3.1 addChannel	84
6.28.3.2 addChannel	84
6.28.3.3 getChannelByName	84
6.28.3.4 log	84
6.28.3.5 log	84
6.28.3.6 setLevel	85
6.29 kfoundation::LongInt Class Reference	85
6.29.1 Detailed Description	86
6.29.2 Member Enumeration Documentation	86
6.29.2.1 encoding_t	86
6.29.3 Constructor & Destructor Documentation	86
6.29.3.1 LongInt	86
6.29.3.2 LongInt	86
6.29.4 Member Function Documentation	87
6.29.4.1 get	87
6.29.4.2 parse	87
6.29.4.3 set	87

6.30	kfoundation::ManagedArray< T > Class Template Reference	87
6.30.1	Detailed Description	89
6.30.2	Constructor & Destructor Documentation	89
6.30.2.1	ManagedArray	89
6.30.2.2	~ManagedArray	89
6.30.3	Member Function Documentation	89
6.30.3.1	at	89
6.30.3.2	indexOf	89
6.30.3.3	indexOf	90
6.30.3.4	insert	90
6.30.3.5	pop	90
6.30.3.6	push	91
6.30.3.7	remove	91
6.30.3.8	setSize	91
6.31	kfoundation::ManagedObject Class Reference	91
6.31.1	Detailed Description	92
6.31.2	Constructor & Destructor Documentation	92
6.31.2.1	~ManagedObject	92
6.32	kfoundation::MasterMemoryManager Class Reference	93
6.32.1	Detailed Description	93
6.32.2	Member Function Documentation	93
6.32.2.1	registerManager	93
6.32.2.2	unregisterManager	94
6.32.2.3	updateTable	94
6.33	kfoundation::MemoryException Class Reference	94
6.33.1	Detailed Description	94
6.34	kfoundation::MemoryManager Class Reference	95
6.34.1	Detailed Description	95
6.34.2	Member Function Documentation	95
6.34.2.1	getTable	95
6.34.2.2	getTableSize	96
6.34.2.3	release	96
6.34.2.4	remove	96
6.35	kfoundation::Mutex Class Reference	96
6.35.1	Detailed Description	97
6.35.2	Constructor & Destructor Documentation	97
6.35.2.1	Mutex	97
6.36	kfoundation::NullPointerException Class Reference	97
6.36.1	Detailed Description	98
6.37	kfoundation::NumericVector< T > Class Template Reference	99

6.37.1	Detailed Description	100
6.37.2	Constructor & Destructor Documentation	100
6.37.2.1	NumericVector	100
6.37.3	Member Function Documentation	100
6.37.3.1	add	100
6.37.3.2	mul	100
6.37.3.3	parseInt	100
6.37.3.4	sub	100
6.38	kfoundation::ObjectDumpBuilderException Class Reference	101
6.38.1	Detailed Description	101
6.38.2	Constructor & Destructor Documentation	102
6.38.2.1	ObjectDumpBuilderException	102
6.39	kfoundation::ObjectPoolMemoryManager< T > Class Template Reference	103
6.39.1	Detailed Description	104
6.39.2	Constructor & Destructor Documentation	104
6.39.2.1	ObjectPoolMemoryManager	104
6.39.2.2	~ObjectPoolMemoryManager	104
6.39.3	Member Function Documentation	104
6.39.3.1	getTable	104
6.39.3.2	getTableSize	104
6.39.3.3	release	105
6.39.3.4	remove	105
6.40	kfoundation::ObjectRecord Struct Reference	105
6.40.1	Detailed Description	106
6.41	kfoundation::ObjectSerializer Class Reference	106
6.41.1	Detailed Description	107
6.41.2	Member Enumeration Documentation	108
6.41.2.1	output_type_t	108
6.41.3	Constructor & Destructor Documentation	108
6.41.3.1	ObjectSerializer	108
6.41.3.2	ObjectSerializer	108
6.41.4	Member Function Documentation	108
6.41.4.1	collection	108
6.41.4.2	endCollection	108
6.41.4.3	endObject	109
6.41.4.4	member	109
6.41.4.5	null	109
6.41.4.6	object	109
6.41.4.7	object	109
6.42	kfoundation::ObjectStreamReader Class Reference	110

6.42.1 Detailed Description	110
6.43 kfoundation::ObjectToken Class Reference	110
6.43.1 Detailed Description	111
6.43.2 Member Function Documentation	112
6.43.2.1 validateClass	112
6.44 kfoundation::OutOfMemoryException Class Reference	112
6.44.1 Detailed Description	113
6.44.2 Constructor & Destructor Documentation	113
6.44.2.1 OutOfMemoryException	113
6.45 kfoundation::OutputStream Class Reference	113
6.45.1 Detailed Description	114
6.45.2 Member Function Documentation	114
6.45.2.1 close	114
6.45.2.2 write	114
6.45.2.3 write	114
6.45.2.4 write	114
6.46 kfoundation::ParseException Class Reference	115
6.46.1 Detailed Description	116
6.46.2 Constructor & Destructor Documentation	116
6.46.2.1 ParseException	116
6.46.2.2 ParseException	116
6.46.2.3 ParseException	116
6.46.3 Member Function Documentation	116
6.46.3.1 getBegin	116
6.46.3.2 getEnd	116
6.47 kfoundation::Path Class Reference	117
6.47.1 Detailed Description	118
6.47.2 Constructor & Destructor Documentation	118
6.47.2.1 Path	118
6.47.3 Member Function Documentation	118
6.47.3.1 changeExtension	118
6.47.3.2 hasExtention	118
6.47.3.3 isAbsolute	119
6.47.3.4 parent	119
6.47.3.5 printToStream	119
6.48 kfoundation::PoolObject Class Reference	119
6.48.1 Detailed Description	120
6.48.2 Member Function Documentation	120
6.48.2.1 finalize	120
6.49 kfoundation::PPtr< T > Class Template Reference	120

6.49.1 Detailed Description	121
6.49.2 Constructor & Destructor Documentation	121
6.49.2.1 PPtr	121
6.49.2.2 PPtr	121
6.49.2.3 PPtr	121
6.49.2.4 PPtr	121
6.49.2.5 PPtr	121
6.50 kfoundation::PredictiveParserBase Class Reference	122
6.50.1 Detailed Description	124
6.50.2 Constructor & Destructor Documentation	124
6.50.2.1 PredictiveParserBase	124
6.50.3 Member Function Documentation	124
6.50.3.1 isSpace	124
6.50.3.2 isValidIdentifierBeginChar	125
6.50.3.3 isValidIdentifierChar	125
6.50.3.4 readAllAlphabet	125
6.50.3.5 readAllAlphanumeric	125
6.50.3.6 readAllBeforeChar	125
6.50.3.7 readAllBeforeCharSkipEscaped	126
6.50.3.8 readAllBeforeNewLine	126
6.50.3.9 readAllBeforeSequence	126
6.50.3.10 readAllBeforeSpace	126
6.50.3.11 readAllBeforeSpaceOrNewLine	127
6.50.3.12 readAllNumeric	127
6.50.3.13 readAlphabet	127
6.50.3.14 readAlphanumeric	127
6.50.3.15 readAny	128
6.50.3.16 readChar	128
6.50.3.17 readChar	128
6.50.3.18 readIdentifier	129
6.50.3.19 readIdentifierBeginChar	130
6.50.3.20 readIdentifierChar	130
6.50.3.21 readNewLine	130
6.50.3.22 readNumber	130
6.50.3.23 readNumber	131
6.50.3.24 readNumber	131
6.50.3.25 readNumeric	131
6.50.3.26 readNumeric	131
6.50.3.27 readSequence	132
6.50.3.28 readSpace	132

6.50.3.29 skipSpaces	132
6.50.3.30 skipSpacesAndNewLines	132
6.50.3.31 testChar	132
6.50.3.32 testChar	133
6.50.3.33 testSequence	133
6.50.3.34 testSpace	133
6.51 kfoundation::ProximityIterator Class Reference	133
6.51.1 Detailed Description	134
6.51.2 Constructor & Destructor Documentation	134
6.51.2.1 ProximityIterator	134
6.51.2.2 ProximityIterator	134
6.51.3 Member Function Documentation	135
6.51.3.1 next	135
6.52 kfoundation::Ptr< T > Class Template Reference	135
6.52.1 Detailed Description	136
6.52.2 Constructor & Destructor Documentation	137
6.52.2.1 Ptr	137
6.52.2.2 Ptr	138
6.52.2.3 ~Ptr	138
6.52.3 Member Function Documentation	138
6.52.3.1 cast	138
6.52.3.2 del	138
6.52.3.3 getRetainCount	138
6.52.3.4 isa	138
6.52.3.5 isValid	139
6.52.3.6 operator*	139
6.52.3.7 operator->	139
6.52.3.8 operator=	139
6.52.3.9 operator=	139
6.52.3.10 release	139
6.52.3.11 replace	139
6.52.3.12 replace	140
6.52.3.13 retain	140
6.52.3.14 trace	140
6.52.3.15 untrace	140
6.53 kfoundation::Range Class Reference	141
6.53.1 Detailed Description	142
6.53.2 Constructor & Destructor Documentation	143
6.53.2.1 Range	143
6.53.3 Member Function Documentation	143

6.53.3.1	border	143
6.53.3.2	isEmpty	143
6.54	kfoundation::Rangelterator Class Reference	143
6.54.1	Detailed Description	144
6.55	kfoundation::RefCountMemoryManager Class Reference	144
6.55.1	Detailed Description	145
6.55.2	Member Function Documentation	145
6.55.2.1	getTable	145
6.55.2.2	getTableSize	146
6.55.2.3	release	146
6.55.2.4	remove	146
6.56	kfoundation::SerializingStreamer Class Reference	146
6.56.1	Detailed Description	147
6.56.2	Member Function Documentation	148
6.56.2.1	printToJsonStream	148
6.56.2.2	printToStream	149
6.56.2.3	printToXmlStream	149
6.57	kfoundation::SPtr< T > Class Template Reference	149
6.57.1	Detailed Description	150
6.57.2	Constructor & Destructor Documentation	150
6.57.2.1	SPtr	150
6.57.2.2	SPtr	151
6.58	kfoundation::StandardInputStreamAdapter Class Reference	151
6.58.1	Detailed Description	152
6.58.2	Member Function Documentation	152
6.58.2.1	isMarkSupported	152
6.58.2.2	mark	152
6.58.2.3	peek	152
6.58.2.4	read	152
6.58.2.5	read	153
6.58.2.6	reset	153
6.58.2.7	skip	153
6.59	kfoundation::StandardOutputStreamAdapter Class Reference	154
6.59.1	Detailed Description	154
6.59.2	Member Function Documentation	154
6.59.2.1	close	154
6.59.2.2	write	155
6.59.2.3	write	155
6.59.2.4	write	155
6.60	kfoundation::Logger::Stream Class Reference	155

6.60.1 Detailed Description	155
6.60.2 Constructor & Destructor Documentation	155
6.60.2.1 Stream	156
6.61 kfoundation::StreamDeserializer Class Reference	156
6.61.1 Detailed Description	156
6.61.2 Member Function Documentation	156
6.61.2.1 readFromXmlStream	156
6.62 kfoundation::Streamer Class Reference	156
6.62.1 Detailed Description	158
6.63 kfoundation::StringInputStream Class Reference	158
6.63.1 Detailed Description	159
6.63.2 Constructor & Destructor Documentation	159
6.63.2.1 StringInputStream	159
6.63.3 Member Function Documentation	159
6.63.3.1 isMarkSupported	159
6.63.3.2 mark	159
6.63.3.3 peek	160
6.63.3.4 read	160
6.63.3.5 read	160
6.63.3.6 reset	160
6.63.3.7 skip	160
6.64 kfoundation::System Class Reference	161
6.64.1 Detailed Description	162
6.64.2 Member Enumeration Documentation	162
6.64.2.1 operating_system_t	162
6.64.3 Member Function Documentation	162
6.64.3.1 exec	162
6.64.3.2 getCurrentTimeInMiliseconds	162
6.64.3.3 getExePath	162
6.64.3.4 sleep	163
6.65 kfoundation::TextToken Class Reference	163
6.65.1 Detailed Description	164
6.66 kfoundation::Thread Class Reference	164
6.66.1 Detailed Description	165
6.66.2 Constructor & Destructor Documentation	165
6.66.2.1 Thread	165
6.66.2.2 Thread	165
6.66.3 Member Function Documentation	165
6.66.3.1 isRunning	165
6.66.3.2 isTheCurrentThread	165

6.66.3.3	run	166
6.66.3.4	start	166
6.67	kfoundation::Timer Class Reference	166
6.67.1	Detailed Description	167
6.67.2	Member Function Documentation	167
6.67.2.1	getCpuTime	167
6.68	kfoundation::Token Class Reference	167
6.68.1	Detailed Description	168
6.68.2	Constructor & Destructor Documentation	169
6.68.2.1	Token	169
6.68.3	Member Function Documentation	169
6.68.3.1	asAttribute	169
6.68.3.2	asCollection	170
6.68.3.3	asEndCollection	170
6.68.3.4	asEndObject	170
6.68.3.5	asObject	170
6.68.4	Member Data Documentation	171
6.68.4.1	codeRange	171
6.69	kfoundation::Tuple Class Reference	171
6.69.1	Detailed Description	172
6.69.2	Member Function Documentation	173
6.69.2.1	negate	173
6.69.2.2	operator%	173
6.69.2.3	operator*	173
6.69.2.4	operator*	173
6.69.2.5	operator+	173
6.69.2.6	operator+	174
6.69.2.7	operator-	174
6.69.2.8	operator-	174
6.69.2.9	operator-	174
6.69.2.10	operator/	174
6.69.2.11	operator/	175
6.70	kfoundation::Tuple1D Class Reference	175
6.70.1	Detailed Description	176
6.70.2	Constructor & Destructor Documentation	176
6.70.2.1	Tuple1D	176
6.71	kfoundation::Tuple2D Class Reference	176
6.71.1	Detailed Description	177
6.71.2	Constructor & Destructor Documentation	178
6.71.2.1	Tuple2D	178

6.71.3	Member Function Documentation	179
6.71.3.1	set	179
6.72	kfoundation::Tuple3D Class Reference	179
6.72.1	Detailed Description	180
6.72.2	Constructor & Destructor Documentation	180
6.72.2.1	Tuple3D	180
6.72.3	Member Function Documentation	180
6.72.3.1	set	180
6.73	kfoundation::UniChar Class Reference	181
6.73.1	Detailed Description	182
6.73.2	Constructor & Destructor Documentation	182
6.73.2.1	UniChar	182
6.73.2.2	UniChar	183
6.73.3	Member Function Documentation	183
6.73.3.1	get	183
6.73.3.2	readUtf8	183
6.73.3.3	readUtf8	183
6.73.3.4	readUtf8	183
6.73.3.5	readUtf8	184
6.73.3.6	set	184
6.73.3.7	toLowerCase	184
6.73.3.8	toUpperCase	184
6.73.3.9	writeUtf8	185
6.73.3.10	writeUtf8	186
6.74	kfoundation::XmlObjectStreamReader Class Reference	186
6.74.1	Detailed Description	187
6.74.2	Constructor & Destructor Documentation	187
6.74.2.1	XmlObjectStreamReader	187

Chapter 1

Overview

KFoundation is a collection of C++ APIs that are essential to every program. Some of them like `Logger` have been missing in C++ standard libraries, and some of them like managed pointers (`Ptr`) replace existing C++ standard APIs like `auto_ptr`, etc. with better functionality.

Most classes in KFoundation implement `Streamer` interface. This interface provides a `toString()` function and also overloads `<<` operator for both ostream and logger stream. For example, if `myObject` is a `Streamer`, you can do any of 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 Arithmetics
- Exceptions

The following briefly 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 series of macros helping to detect the operating system. These are, `KF_LINUX`, `KF_MAC`, `KF_MACH`, `KF_SOLARIS`, and `KF_FREE_BSD`. If the target machine or operating system is supported by KFoundation, `KF_SUPPORTED` will be defined.

`IS_NULL(X)` and `NOT_NULL(X)` macros are provided to help creating more readable codes.

`ISA(X)`, and `AS(X)` that are related to memory management module. Usage:

```
if(myObject.ISA(MyClass)) {  
    myObject.AS(MyClass)->methodSpecificToMyClass();  
}
```

[See all APIs here.](#)

Memory Management

KFoundation provides an automatic memory management solution with several advantages. 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 instead throws `NullPointerException` 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 [MemoryManager](#).

[See all APIs here.](#)

Wrappers and Containers

Languages like Java offer type-wrapper classes corresponding to primitive types. These are useful 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. `NumericVector<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.](#)

I/O

KFoundation offers `InputStream` and `OutputStream` classes as minimalist equivalent of standard `ostream` and `istream`. Thanks to their minimalist design, an extensive set of stream types could be provided by KFoundation. These are, `BufferInputStream`, `BufferOutputStream`, `FileInputStream`, `FileOutputStream`, `InternetInputStream`, `InternetOutputStream`, and `StringInputStream`. If you need to use a standard `istream` or `ostream` object with KFoundation simply use `StandardInputStreamAdapter` and `StandardOutputStreamAdapter`.

[See all APIs here.](#)

Object Serialization and Deserialization

KFoundation offers a powerful and intuitive way for objects to implement serialization and deserialization capabilities. To be serializable, objects need to implement `SerializingStreamer` interface. And to be deserializable they 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 DLOG_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 object deserialization in KFoundation.

[See all APIs here.](#)

Range Arithmetics

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

[Tuple](#) represents an element in n-dimensional 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 adjacent ranges and so on.

[Rangelterator](#) is an exciting feature that summarizes all your `for` loops into one small one, and is supplemented by [Proximitylterator](#) to make stencil 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.](#)

Chapter 2

Module Index

2.1 Modules

Here is a list of all modules:

Macros and Typedefs	15
Exceptions	17
Memory Management	18
Input/Output	19
Mutithreading	20
Wrappers and Containers	21
Utilities	22
Object Serialization and Deserialization	23
Range Arithmetics	24

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

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

kfoundation::Logger::Channel	38
kfoundation::Condition	46
kfoundation::Logger	81
kfoundation::ManagedObject	91
kfoundation::Array< T >	25
kfoundation::NumericVector< T >	99
kfoundation::Bool	30
kfoundation::Double	50
kfoundation::InputStream	62
kfoundation::BufferInputStream	31
kfoundation::FileInputStream	54
kfoundation::InternetInputStream	69
kfoundation::StandardInputStreamAdapter	151
kfoundation::StringInputStream	158
kfoundation::Int	65
kfoundation::LongInt	85
kfoundation::ManagedArray< T >	87
kfoundation::ObjectSerializer	106
kfoundation::ObjectStreamReader	110
kfoundation::Token	167
kfoundation::AttributeToken	29
kfoundation::CollectionToken	45
kfoundation::EndCollectionToken	51
kfoundation::EndObjectToken	53
kfoundation::ObjectToken	110
kfoundation::TextToken	163
kfoundation::OutputStream	113
kfoundation::BufferOutputStream	35
kfoundation::FileOutputStream	57
kfoundation::InternetOutputStream	74
kfoundation::StandardOutputStreamAdapter	154
kfoundation::Path	117
kfoundation::PoolObject	119
kfoundation::PredictiveParserBase	122
kfoundation::Thread	164
kfoundation::Timer	166
kfoundation::UniChar	181

kfoundation::XmlObjectStreamReader	186
kfoundation::MasterMemoryManager	93
kfoundation::MemoryManager	95
kfoundation::ObjectPoolMemoryManager< T >	103
kfoundation::RefCountMemoryManager	144
kfoundation::Mutex	96
kfoundation::ObjectRecord	105
kfoundation::Logger::Stream	155
kfoundation::StreamDeserializer	156
kfoundation::Streamer	156
kfoundation::Bool	30
kfoundation::Direction	47
kfoundation::DirectionIterator	48
kfoundation::Double	50
kfoundation::Int	65
kfoundation::InternetAddress	67
kfoundation::LongInt	85
kfoundation::NumericVector< T >	99
kfoundation::Range	141
kfoundation::SerializingStreamer	146
kfoundation::CodeLocation	41
kfoundation::CodeRange	43
kfoundation::InternetInputStream	69
kfoundation::InternetOutputStream	74
kfoundation::KFException	80
kfoundation::InvalidFormatException	76
kfoundation::IOException	79
kfoundation::MemoryException	94
kfoundation::IndexOutOfBoundsException	60
kfoundation::InvalidPointerException	78
kfoundation::NullPointerException	97
kfoundation::OutOfMemoryException	112
kfoundation::ObjectDumpBuilderException	101
kfoundation::ParseException	115
kfoundation::ManagedArray< T >	87
kfoundation::ObjectPoolMemoryManager< T >	103
kfoundation::Path	117
kfoundation::Ptr< T >	135
kfoundation::PPtr< T >	120
kfoundation::SPtr< T >	149
kfoundation::PPtr< kfoundation::XmlObjectStreamReader >	120
kfoundation::RefCountMemoryManager	144
kfoundation::Ptr< Insertion >	135
kfoundation::Ptr< kfoundation::Array< int > >	135
kfoundation::Ptr< kfoundation::InputStream >	135
kfoundation::Ptr< kfoundation::ManagedArray< Insertion > >	135
kfoundation::Ptr< kfoundation::ManagedArray< Removal > >	135
kfoundation::Ptr< kfoundation::ManagedObject >	135
kfoundation::Ptr< kfoundation::Path >	135
kfoundation::Ptr< kfoundation::PredictiveParserBase >	135
kfoundation::Ptr< kfoundation::Token >	135
kfoundation::SPtr< kfoundation::Token >	149
kfoundation::Ptr< UniString >	135
kfoundation::Ptr< XmlAttribute >	135
kfoundation::Timer	166
kfoundation::Tuple	171
kfoundation::ProximityIterator	133

kfoundation::RangeIterator	143
kfoundation::Tuple1D	175
kfoundation::Tuple2D	176
kfoundation::Tuple3D	179
kfoundation::UniChar	181
kfoundation::System	161

Chapter 4

Class Index

4.1 Class List

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

kfoundation::Array< T >	25
A resizable, one-dimensional indexed container	
kfoundation::AttributeToken	29
Represents an attribute in the parsed stream	
kfoundation::Bool	30
Wrapper class for <code>bool</code> type	
kfoundation::BufferInputStream	31
Input stream to read from a buffer in memory	
kfoundation::BufferOutputStream	35
Output stream used to write to a buffer in memory	
kfoundation::Logger::Channel	38
Logger channel	
kfoundation::CodeLocation	41
Encodes location of a character in a text file	
kfoundation::CodeRange	43
Encodes the location of the begining and end of a text file fragment	
kfoundation::CollectionToken	45
Represents begining of a collection in the parsed stream	
kfoundation::Condition	46
Condition variable, used to control a thread from another	
kfoundation::Direction	47
Represents directions in n-dimensional space	
kfoundation::DirectionIterator	48
Used to iterate all possible directions in an space of given dimensions	
kfoundation::Double	50
Wrapper class for <code>double</code> type	
kfoundation::EndCollectionToken	51
Represents end of a collection in the parsed stream	
kfoundation::EndObjectToken	53
Represents end of an object in the parsed stream	
kfoundation::FileInputStream	54
Input stream to read from file	
kfoundation::FileOutputStream	57
Output stream used to write data on file	
kfoundation::IndexOutOfBoundsException	60
Thrown to signal access to a nonexistent element in an array	
kfoundation::InputStream	62
Abstract interface for all input streams	

kfoundation::Int	65
Wrapper for <code>int</code> type	
kfoundation::InternetAddress	67
Encodes an IP address and port number	
kfoundation::InternetInputStream	69
Input stream used to read from a TCP/IP port	
kfoundation::InternetOutputStream	74
Input stream used to write to TCP/IP socket	
kfoundation::InvalidFormatException	76
Thrown when an input with an invalid format is encountered	
kfoundation::InvalidPointerException	78
Thrown on attempt to access an invalid pointer	
kfoundation::IOException	79
Thrown to signal an IO-related exception	
kfoundation::KFException	80
Superclass for all exceptions in KFoundation	
kfoundation::Logger	81
Multi-channel, multi-level logger utility	
kfoundation::LongInt	85
Wrapper class for 'long int' type	
kfoundation::ManagedArray< T >	87
One-dimensional indexed collection of ManagedObjects	
kfoundation::ManagedObject	91
The root class for all classes using KFoundation framework	
kfoundation::MasterMemoryManager	93
Manages all the memory managers used in a process	
kfoundation::MemoryException	94
Used to throw exceptions related to memory	
kfoundation::MemoryManager	95
Abstract interface to be implemented by all memory managers	
kfoundation::Mutex	96
Mutex, used to prevent multiple threads to enter a critical region	
kfoundation::NullPointerException	97
Thrown on attempt to access to a null pointer	
kfoundation::NumericVector< T >	99
A subclass of Array , adds numeric operations	
kfoundation::ObjectDumpBuilderException	101
Thrown when ObjectSerializer is used in an invalid way	
kfoundation::ObjectPoolMemoryManager< T >	103
Reuses the objects in a preallocated pool whenever a new instance is needed	
kfoundation::ObjectRecord	105
Structure of memory manager's table records	
kfoundation::ObjectSerializer	106
Provides APIs to serialize an object	
kfoundation::ObjectStreamReader	110
Generic interface for utility object used to read objects from an stream of a given format	
kfoundation::ObjectToken	110
Represents beginning of an object in the parsed stream	
kfoundation::OutOfMemoryException	112
Thrown on out of memory	
kfoundation::OutputStream	113
Abstract interface for all output streams	
kfoundation::ParseException	115
Thrown when a parsing error happens	
kfoundation::Path	117
Used to represent and manipulate file and directory pathnames	
kfoundation::PoolObject	119
Superclass for all objects that are meant to be allocated by a pool memory manager	

kfoundation::PPtr< T >	Passive pointer, will not release and retain automatically	120
kfoundation::PredictiveParserBase	Packs ample of basic functionalities to implement any predictive parser	122
kfoundation::ProximityIterator	Iterates the proximity of a desired point	133
kfoundation::Ptr< T >	Managed pointer to a class of given template type	135
kfoundation::Range	Represents a range in n-dimensional space	141
kfoundation::RangeIterator	Used to iterate over all points in a given range	143
kfoundation::RefCountMemoryManager	Reference counting memory manager	144
kfoundation::SerializingStreamer	Objects implementing this class can be serialized into any format allowed by ObjectSerializer	146
kfoundation::SPtr< T >	Static pointer, makes the pointed object immortal	149
kfoundation::StandardInputStreamAdapter	Wraps around the given <code>istream</code> (C++ standard libraries) to be read as a KFoundation input stream	151
kfoundation::StandardOutputStreamAdapter	KFoundation wrapper for C++ <code>ostream</code>	154
kfoundation::Logger::Stream	Log stream	155
kfoundation::StreamDeserializer	Interface to be implemented by any class that can be deserialized from stream	156
kfoundation::Streamer	Base class for all classes that can print information about themselves to a <code>std::ostream</code>	156
kfoundation::StringInputStream	Input stream to read from string	158
kfoundation::System	Provides a cross-platform way to access system features	161
kfoundation::TextToken	Represents a text body (CDATA) in the parsed stream	163
kfoundation::Thread	An object-oriented, cross-platform abstraction for thread	164
kfoundation::Timer	Utility class to measure execution time of a code fragment	166
kfoundation::Token	Represents a token in a stream	167
kfoundation::Tuple	Represents a point in n-dimensional space	171
kfoundation::Tuple1D	1-dimensional specialization of Tuple	175
kfoundation::Tuple2D	2-dimensional specialization of Tuple	176
kfoundation::Tuple3D	3-dimensional specialization of Tuple	179
kfoundation::UniChar	Wrapper class for unicode character	181
kfoundation::XmlObjectStreamReader	ObjectStreamReader to deserialize XML streams	186

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.

Enumerations

- enum `kfoundation::kf_comparison_t` { `kfoundation::SMALLER` = -1, `kfoundation::EQUAL` = 0, `kfoundation::GREATER` = 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 Equal.

GREATER Greater.

5.2 Exceptions

Classes

- class [kfoundation::IndexOutOfBoundsException](#)
Thrown to signal access to a nonexistent 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 exceptions 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

5.3 Memory Management

Classes

- class [kfoundation::InvalidPointerException](#)
Thrown on attempt to access an invalid pointer.
- class [kfoundation::ManagedArray< T >](#)
One-dimensional indexed collection of [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

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 beginning 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 interface 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 *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 themselves 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

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`,

```
KF_SYNCHRONIZED(m,
    // Critical region
)
```

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

```
KF_SYNCHRONIZED(m, {
    // Critical region
});
```

5.6 Wrappers and Containers

Classes

- class [kfoundation::Array< T >](#)
A resizable, one-dimensional indexed container.
- class [kfoundation::Bool](#)
Wrapper class for `bool` type.
- class [kfoundation::Double](#)
Wrapper class for `double` type.
- class [kfoundation::IndexOutOfBoundsException](#)
Thrown to signal access to a nonexistent 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-dimensional 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

5.7 Utilities

Classes

- class [kfoundation::CodeLocation](#)
Encodes location of a character in a text file.
- class [kfoundation::CodeRange](#)
Encodes the location of the beginning and end of a text file fragment.
- class [kfoundation::Logger](#)
Multi-channel, multi-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 system 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 objects from a stream of a given format.
- class [kfoundation::Token](#)
Represents a token in a stream.
- class [kfoundation::ObjectToken](#)
Represents beginning 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 beginning 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

5.9 Range Arithmetics

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

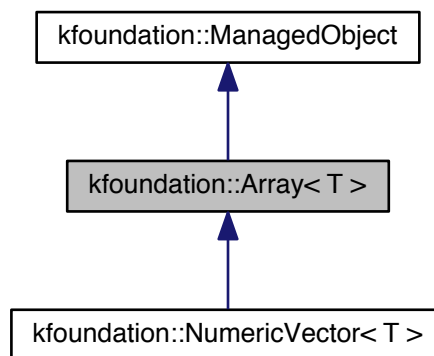
Class Documentation

6.1 kfoundation::Array< T > Class Template Reference

A resizable, one-dimensional indexed container.

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

Inheritance diagram for kfoundation::Array< T >:



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 ()`
Destructor.
- `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

<i>values</i>	Location of the memory containing values to be copied.
<i>size</i>	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

<i>index</i>	The index of the item to be accessed.
--------------	---------------------------------------

Exceptions

<i>Throws</i>	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

<i>index</i>	The index of the item to be accessed.
--------------	---------------------------------------

Exceptions

<i>Throws</i>	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

<i>value</i>	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

<i>value</i>	The value to search for.
--------------	--------------------------

Returns

If found, the index of the first occurrence 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 occurrence of the given value starting from the given offset.

Parameters

<i>offset</i>	The index to start the search from.
<i>value</i>	The value to search for.

Returns

If found, the index of the first occurrence 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

<i>index</i>	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

<i>index</i>	The index to insert at.
<i>value</i>	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

<i>Throws</i>	IndexOutOfBoundsException if the array is empty.
---------------	--

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

<i>value</i>	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:

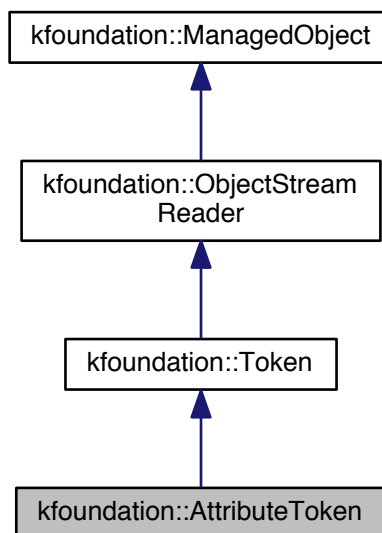
- ArrayDecl.h
- Array.h

6.2 kfoundation::AttributeToken Class Reference

Represents an attribute in the parsed stream.

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

Inheritance diagram for kfoundation::AttributeToken:



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 throwInvalidName` () const
Throws a `ParseException` explaining 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:

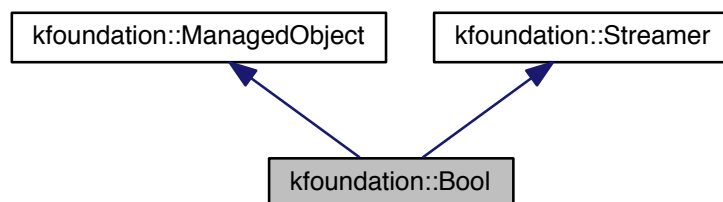
- ObjectStreamReader.h
- ObjectStreamReader.cpp

6.3 kfoundation::Bool Class Reference

Wrapper class for `bool` type.

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

Inheritance diagram for kfoundation::Bool:



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

<i>value</i>	The value to be converted to string.
--------------	--------------------------------------

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

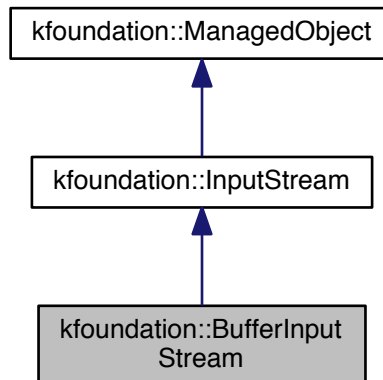
- Bool.h
- Bool.cpp

6.4 kfoundation::BufferInputStream Class Reference

Input stream to read from a buffer in memory.

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

Inheritance diagram for kfoundation::BufferInputStream:



Public Member Functions

- [BufferInputStream](#) (const [kf_octet_t](#) *const buffer, const [kf_int32_t](#) size, bool takeover)
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

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

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

<i>buffer</i>	The octets to read.
<i>nOctets</i>	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

<i>nOctets</i>	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:

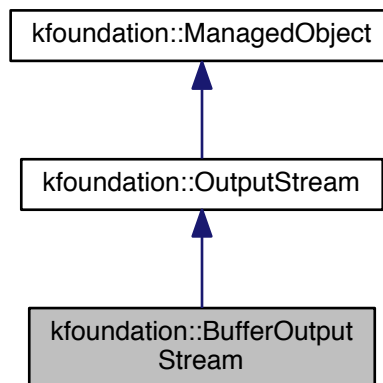
- [BufferInputStream.h](#)
- [BufferInputStream.cpp](#)

6.5 kfoundation::BufferOutputStream Class Reference

Output stream used to write to a buffer in memory.

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

Inheritance diagram for kfoundation::BufferOutputStream:



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

<i>capacity</i>	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

<i>buffer</i>	The octets to write.
<i>nOctets</i>	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

<i>octet</i>	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

<i>is</i>	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](#) ()
Destructor.
- [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 printSecondAndMillis, 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 lvl) const`

Checks if the current filtering level equals the given parameter.

Parameters

<i>lvl</i>	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

<i>name</i>	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

<i>printHourAndMinutes</i>	If set <code>true</code> the hour and minute will be printed in HH:MM format.
<i>printSecondAndMilisecond</i>	If set <code>true</code> second and milisecond will be printed in SS.m format. If <code>printHourAndMinutes</code> is already set to <code>true</code> the output will be in HH:MM:SS.m format.
<i>printLocation</i>	If set <code>true</code> 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

<i>level</i>	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

<i>isSilent</i>	If set <code>true</code> this channel will be silenced, otherwise it will resume producing output.
-----------------	--

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

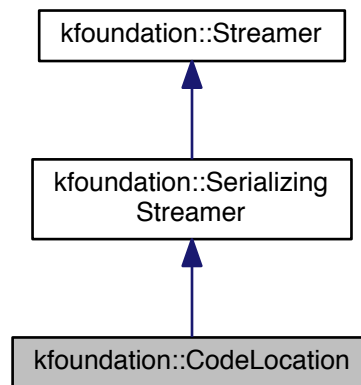
- `Logger.h`
- `Logger.cpp`

6.7 kfoundation::CodeLocation Class Reference

Encodes location of a character in a text file.

```
#include <CodeLocation.h>
```

Inheritance diagram for `kfoundation::CodeLocation`:



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 charIndex 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 beginning 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

<i>line</i>	Line number
<i>col</i>	Column number
<i>byteIndex</i>	Byte index from the beginning of the file.

6.7.2.2 void kfoundation::CodeLocation::set (long int *line*, long int *col*, long int *charIndex*, long int *byteIndex*)

Setter, sets all values as given in arguments.

Parameters

<i>line</i>	Line number
<i>col</i>	Column number
<i>charIndex</i>	Character index from the beginning of the file.
<i>byteIndex</i>	Byte index from the beginning of the file.

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

Setter, sets byte index.

Parameters

<i>byteIndex</i>	Byte index from the beginning of the file.
<i>Reference</i>	to self.

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

Setter, sets character index.

Parameters

<i>charIndex</i>	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

<i>col</i>	Column number
------------	---------------

Returns

Reference to self.

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

Setter, sets the line number.

Parameters

<i>line</i>	Line number
-------------	-------------

Returns

Reference to self.

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

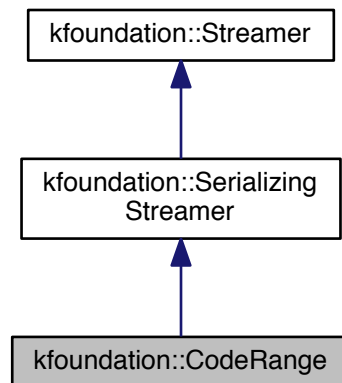
- CodeLocation.h
- CodeLocation.cpp

6.8 kfoundation::CodeRange Class Reference

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

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

Inheritance diagram for kfoundation::CodeRange:



Public Member Functions

- [CodeRange](#) (const [CodeLocation](#) &begin, const [CodeLocation](#) &end)
Constructor.
- const [CodeLocation](#) & [getBegin](#) () const
Returns the beginning 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 beginning 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

<i>begin</i>	The beginning of the range (inclusive).
<i>end</i>	The end of the range (inclusive).

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

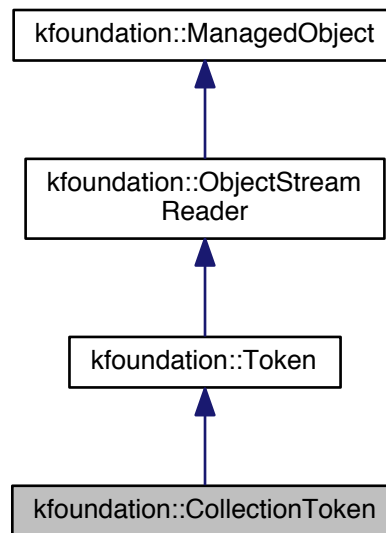
- [CodeRange.h](#)
- [CodeRange.cpp](#)

6.9 kfoundation::CollectionToken Class Reference

Represents begining of a collection in the parsed stream.

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

Inheritance diagram for kfoundation::CollectionToken:



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

6.10.2.1 void kfoundation::Condition::block (const kf_int64_t timeout) [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::getCurrentTimeInMilliseconds() + dt);
```

Parameters

<i>timeout</i>	Target absolute time measured in milliseconds.
----------------	--

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:

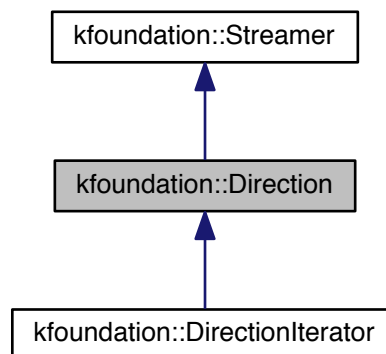
- Condition.h
- Condition.cpp

6.11 kfoundation::Direction Class Reference

Represents directions in n-dimensional space.

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

Inheritance diagram for kfoundation::Direction:



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 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:

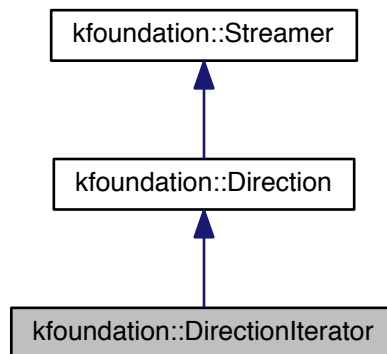
- Direction.h
- Direction.cpp

6.12 kfoundation::DirectionIterator Class Reference

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

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

Inheritance diagram for kfoundation::DirectionIterator:



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

<i>nDims</i>	Number of dimensions.
--------------	-----------------------

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

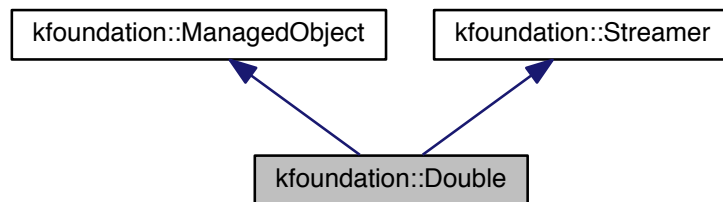
- DirectionIterator.h
- DirectionIterator.cpp

6.13 kfoundation::Double Class Reference

Wrapper class for `double` type.

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

Inheritance diagram for `kfoundation::Double`:



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

<i>str</i>	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:

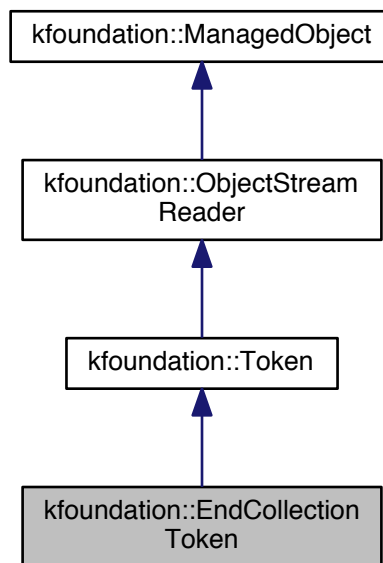
- Double.h
- Double.cpp

6.14 kfoundation::EndCollectionToken Class Reference

Represents end of a collection in the parsed stream.

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

Inheritance diagram for kfoundation::EndCollectionToken:



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:

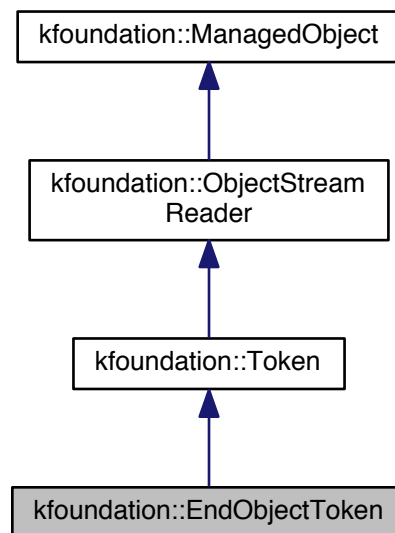
- `ObjectStreamReader.h`
- `ObjectStreamReader.cpp`

6.15 kfoundation::EndObjectToken Class Reference

Represents end of an object in the parsed stream.

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

Inheritance diagram for kfoundation::EndObjectToken:



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:

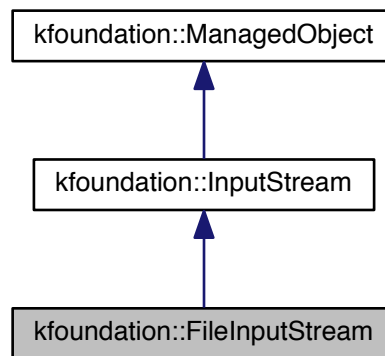
- [ObjectStreamReader.h](#)
- [ObjectStreamReader.cpp](#)

6.16 kfoundation::FileInputStream Class Reference

Input stream to read from file.

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

Inheritance diagram for kfoundation::FileInputStream:



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 opened 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

<code>path</code>	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

<code>fileName</code>	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

<i>buffer</i>	The octets to read.
<i>nOctets</i>	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 beginning 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

<i>nOctets</i>	The desired number of octets to skip.
----------------	---------------------------------------

Returns

The actual number of octets skipped.

Implements [kfoundation::InputStream](#).

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

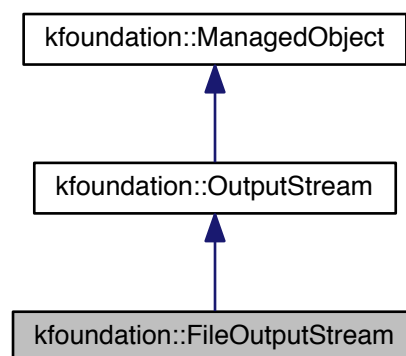
- [FileInputStream.h](#)
- [FileInputStream.cpp](#)

6.17 kfoundation::FileOutputStream Class Reference

Output stream used to write data on file.

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

Inheritance diagram for kfoundation::FileOutputStream:



Public Member Functions

- [FileOutputStream](#) (PPtr< [Path](#) > path)
Constructor, opens the file pointed by the given path object to write.
- [~FileOutputStream](#) ()
Destructor.
- 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
Erases the file contents and resets the stream position to the beginning 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 writing begins from the end of the file. If it is desired to erase the existing contents of the file, use [truncate\(\)](#) method.

Parameters

<i>path</i>	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

<i>Throws</i>	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

<i>Throws</i>	IOException if lock could not be placed.
---------------	--

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

Erases the file contents and resets the stream position to the beginning of the file.

Exceptions

<i>Throws</i>	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

<i>Throw</i>	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

<i>buffer</i>	The octets to write.
<i>nOctets</i>	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

<i>octet</i>	The octet to write
--------------	--------------------

Implements [kfoundation::OutputStream](#).

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

<i>is</i>	The stream to read from.
-----------	--------------------------

Implements [kfoundation::OutputStream](#).

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

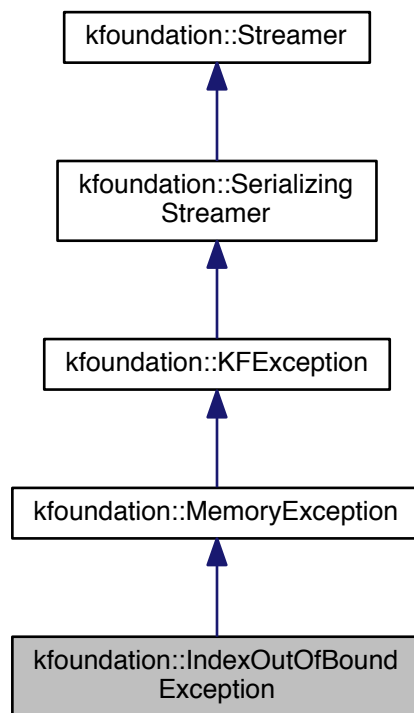
- `FileOutputStream.h`
- `FileOutputStream.cpp`

6.18 kfoundation::IndexOutOfBoundsException Class Reference

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

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

Inheritance diagram for kfoundation::IndexOutOfBoundException:



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

<i>message</i>	A message describing the cause of the exception.
----------------	--

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

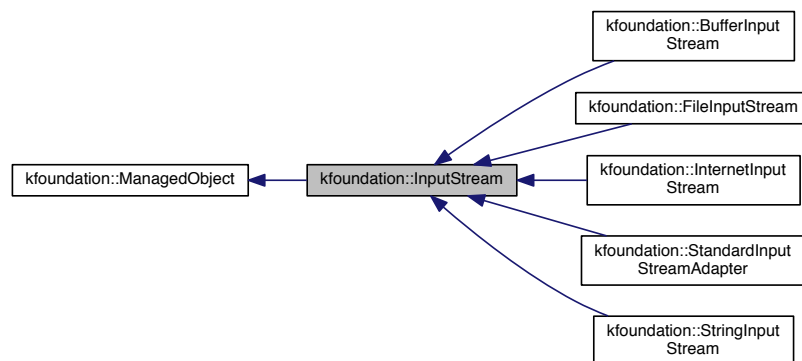
- IndexOutOfBoundsException.h
- IndexOutOfBoundsException.cpp

6.19 kfoundation::InputStream Class Reference

Abstract interface for all input streams.

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

Inheritance diagram for kfoundation::InputStream:



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::StandardInputStreamAdapter](#), and [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::StandardInputStreamAdapter](#), and [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::StandardInputStreamAdapter](#), and [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

<i>buffer</i>	The octets to read.
---------------	---------------------

<i>nOctets</i>	Maximum number of octets to read.
----------------	-----------------------------------

Returns

The actual number of octets read.

Implemented in [kfoundation::InternetInputStream](#), [kfoundation::BufferInputStream](#), [kfoundation::FileInputStream](#), [kfoundation::StandardInputStreamAdapter](#), and [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::StandardInputStreamAdapter](#), and [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::StandardInputStreamAdapter](#), and [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

<i>nOctets</i>	The desired number of octets to skeep.
----------------	--

Returns

The actual number of octets skipped.

Implemented in [kfoundation::InternetInputStream](#), [kfoundation::BufferInputStream](#), [kfoundation::FileInputStream](#), [kfoundation::StandardInputStreamAdapter](#), and [kfoundation::StringInputStream](#).

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

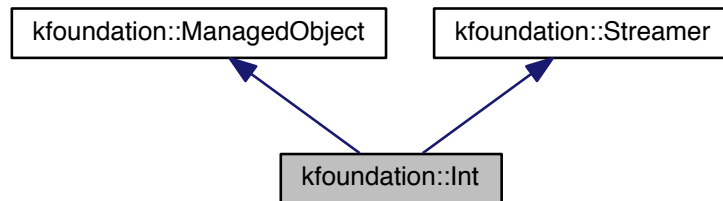
- [InputStream.h](#)

6.20 kfoundation::Int Class Reference

Wrapper for `int` type.

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

Inheritance diagram for `kfoundation::Int`:



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 toHexString (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

<i>str</i>	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

<i>v</i>	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:

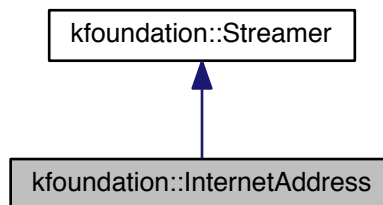
- `Int.h`
- `Int.cpp`

6.21 kfoundation::InternetAddress Class Reference

Encodes an IP address and port number.

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

Inheritance diagram for kfoundation::InternetAddress:



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)`
Constructs 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 * getIp () 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.

- [IPAddress](#) `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::IPAddress::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::IPAddress::IPAddress (const kf_octet_t * ip, kf_int32_t port)`

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

Parameters

<i>ip</i>	An array of <code>kf_octet_t</code> of size 4.
<i>port</i>	A port number.

6.21.3.2 `kfoundation::IPAddress::IPAddress (const string & str)`

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

Parameters

<i>str</i>	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::IPAddress::IPAddress (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

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

<i>port</i>	The port number for the new object.
-------------	-------------------------------------

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

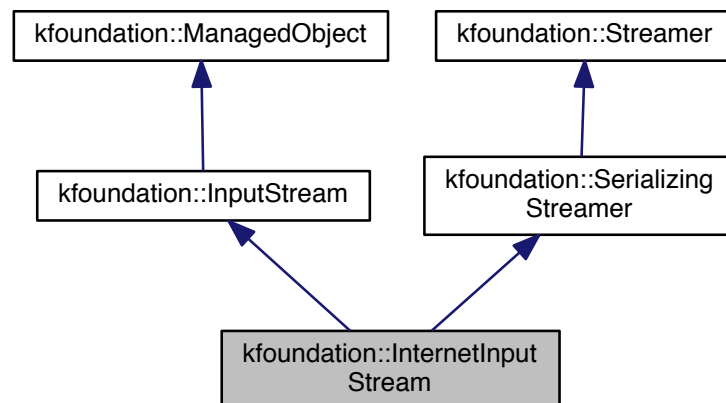
- `InternetAddress.h`
- `InternetAddress.cpp`

6.22 kfoundation::InternetInputStream Class Reference

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

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

Inheritance diagram for `kfoundation::InternetInputStream`:



Public Member Functions

- [InternetInputStream \(\)](#)
Constructor, creates a closed input stream.
- [~InternetInputStream \(\)](#)
Destructor.
- `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 incoming connection is established.*
- 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 established.*
- `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 privileges to bind to the given address and port, or the port is already in use, this function will throw an exception.

Parameters

<i>address</i>	The address to bind this stream to.
----------------	-------------------------------------

Exceptions

<i>Throws</i>	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 incoming connection is established.

Exceptions

<i>Throws</i>	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

<i>buffer</i>	The octets to read.
<i>nOctets</i>	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

<i>nOctets</i>	The desired number of octets to skip.
----------------	---------------------------------------

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:

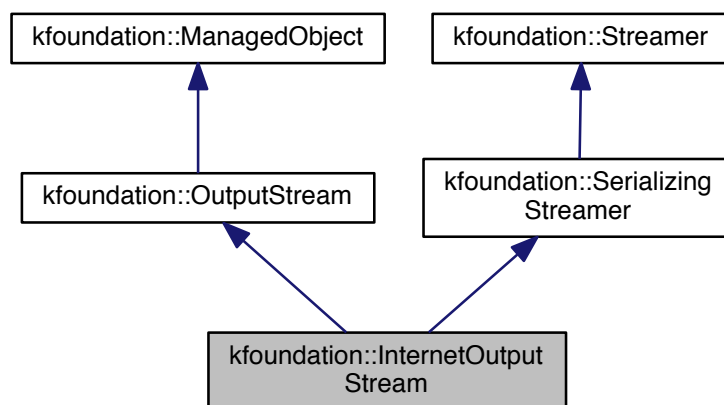
- InternetInputStream.h
- InternetInputStream.cpp

6.23 kfoundation::InternetOutputStream Class Reference

Input stream used to write to TCP/IP socket.

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

Inheritance diagram for kfoundation::InternetOutputStream:



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 assignend 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

<i>Throws</i>	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

<i>buffer</i>	The octets to write.
<i>nOctets</i>	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

<i>octet</i>	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

<i>is</i>	The stream to read from.
-----------	--------------------------

Implements [kfoundation::OutputStream](#).

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

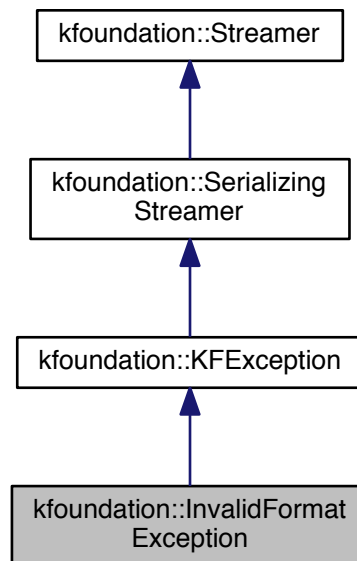
- `InternetOutputStream.h`
- `InternetOutputStream.cpp`

6.24 kfoundation::InvalidFormatException Class Reference

Thrown when an input with an invalid format is encountered.

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

Inheritance diagram for kfoundation::InvalidFormatException:



Public Member Functions

- [InvalidFormatException](#) (string message)

Constructor.

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)

Constructor.

Parameters

<i>message</i>	A message describing the error.
----------------	---------------------------------

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

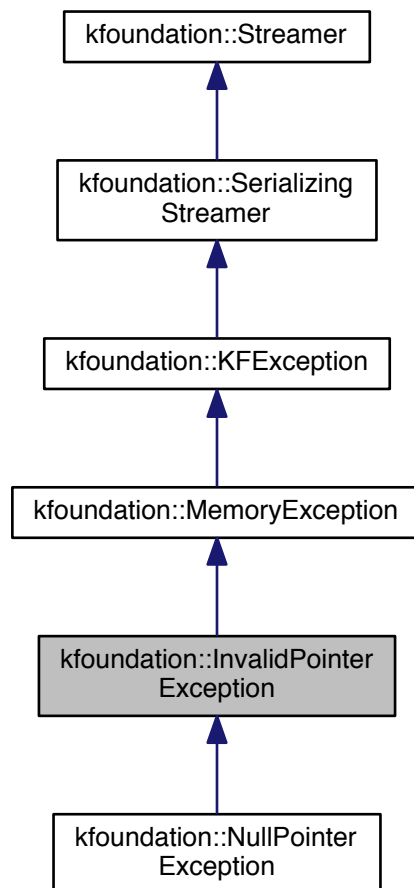
- InvalidFormatException.h
- InvalidFormatException.cpp

6.25 kfoundation::InvalidPointerException Class Reference

Thrown on attempt to access an invalid pointer.

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

Inheritance diagram for kfoundation::InvalidPointerException:



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:

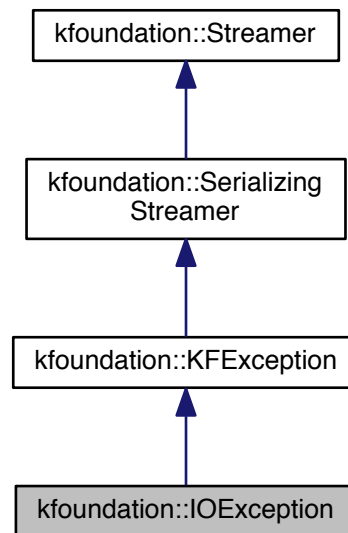
- InvalidPointerException.h
- InvalidPointerException.cpp

6.26 kfoundation::IOException Class Reference

Thrown to signal an IO-related exception.

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

Inheritance diagram for kfoundation::IOException:



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

<i>message</i>	A message explaining the cause of exception.
----------------	--

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

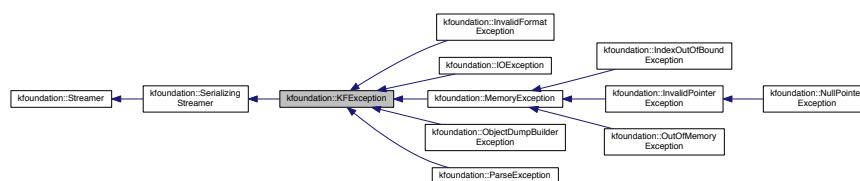
- IOException.h
- IOException.cpp

6.27 kfoundation::KFException Class Reference

Superclass for all exceptions in KFoundation.

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

Inheritance diagram for kfoundation::KFException:



Public Member Functions

- [KFException](#) (string message)
Primary constructor.
- [KFException](#) (const [KFException](#) &other)
Copy constructor.
- [~KFException](#) () throw ()
Destructor.
- 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;
}
```

6.27.2 Constructor & Destructor Documentation

6.27.2.1 kfoundation::KException::KException (string *message*)

Primary constructor.

Once invoked, creates and stores the current stack trace.

Parameters

<i>message</i>	The error message describing the exception.
----------------	---

6.27.2.2 kfoundation::KException::KException (const KException & *other*)

Copy constructor.

Parameters

<i>other</i>	The object to be copied.
--------------	--------------------------

6.27.3 Member Function Documentation

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

Returns

The message assigned to this exception.

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

Serializing method.

Parameters

<i>builder</i>	The ObjectSerializer used to build the output.
----------------	--

Implements [kfoundation::SerializingStreamer](#).

6.27.3.3 void kfoundation::KException::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

<i>name</i>	The name of the exception class.
-------------	----------------------------------

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

- KException.h
- KException.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 reference to the channel with the given name.
- void [removeAllChannels](#) ()
Removes all channels.
- [Stream](#) & [log](#) ([level_t](#) level, const char fileName[], int lineNumber, 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, multi-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,

```
LOG << "Hello" << EL;
```

is equivalent to

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

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);
```

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

<i>name</i>	A name for the new channel.
<i>os</i>	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

<i>name</i>	A name for the new channel.
<i>fileName</i>	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

<i>An</i>	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

<i>level</i>	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

<i>The</i>	level to set.
------------	---------------

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

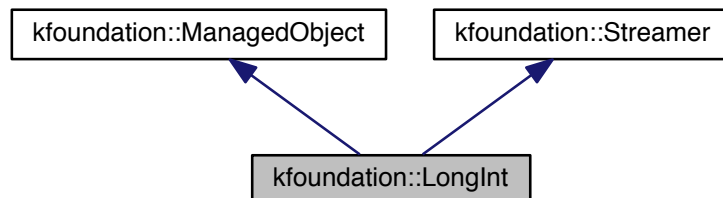
- [Logger.h](#)
- [Logger.cpp](#)

6.29 kfoundation::LongInt Class Reference

Wrapper class for 'long int' type.

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

Inheritance diagram for kfoundation::LongInt:



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 to 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 [toHexString](#) (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 parameter.

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

<i>str</i>	The string to parse.
------------	----------------------

<i>encoding</i>	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

<i>str</i>	The string to parse
<i>encoding</i>	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

<i>v</i>	The value to be set.
----------	----------------------

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

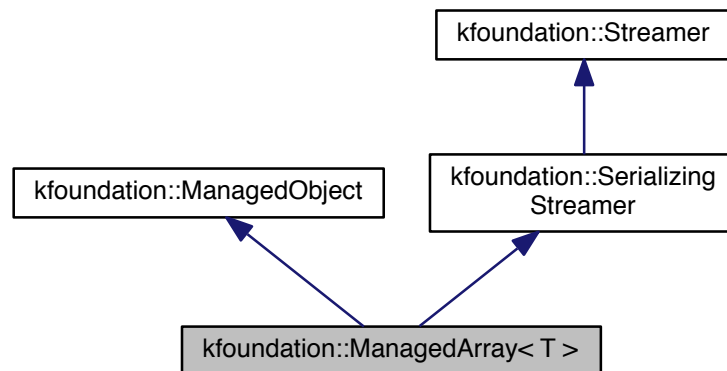
- LongInt.h
- LongInt.cpp

6.30 kfoundation::ManagedArray< T > Class Template Reference

One-dimentional indexed collection of [ManagedObjects](#).

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

Inheritance diagram for kfoundation::ManagedArray< T >:



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 occurrence of the given pointer in the array.
- [kf_int32_t](#) [indexOf](#) ([kf_int32_t](#) offset, [PPtr](#)< T > value) const
Searches for the occurrence of the first occurrence 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-dimensional 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

<i>initialCapacity</i>	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<typename 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

<i>index</i>	The index of the element to be accessed.
--------------	--

Exceptions

<i>Throws</i>	IndexOutOfBoundsException 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 occurrence of the given pointer in the array.

Note

This method only compares pointers not the pointed values.

Parameters

<i>value</i>	The pointer to search for.
--------------	----------------------------

Returns

The index of the first occurrence 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 occurrence of the first occurrence of the given pointer after the given offset in the array.

Note

This method only compares pointers not the pointed values.

Parameters

<i>value</i>	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, PPtr< 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

<i>index</i>	The index to be inserted at.
<i>value</i>	The pointer to be inserted.

6.30.3.5 `template<typename T> PPtr< 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

<i>Throws</i>	IndexOutOfBoundsException 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

<i>value</i>	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

<i>index</i>	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

<i>size</i>	The array's new size.
-------------	-----------------------

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

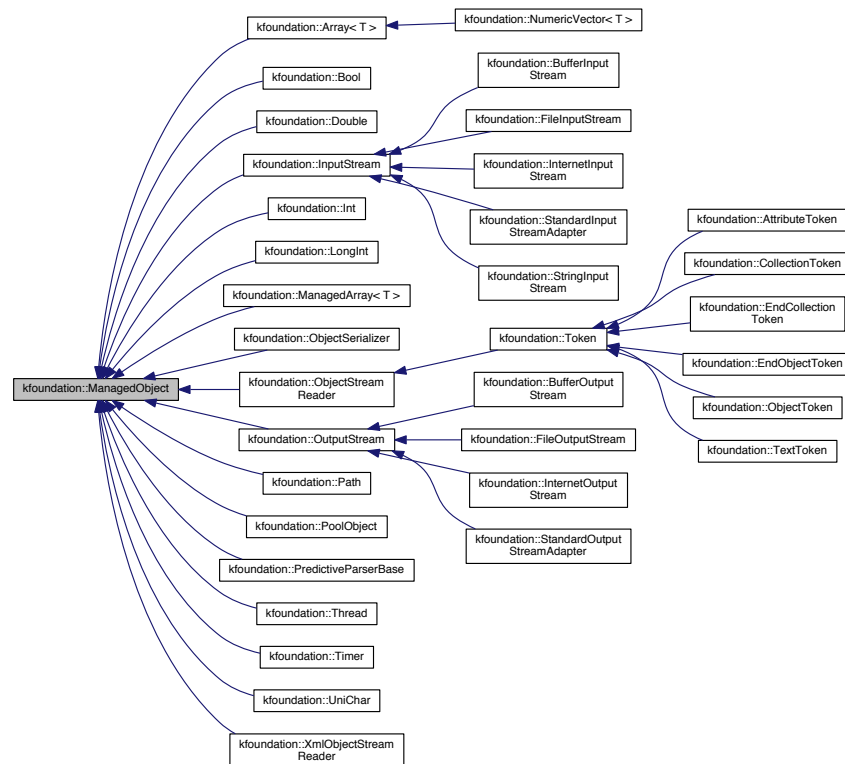
- ManagedArrayDecl.h
- ManagedArray.h

6.31 kfoundation::ManagedObject Class Reference

The root class for all classes using KFoundation framework.

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

Inheritance diagram for kfoundation::ManagedObject:



Public Member Functions

- [ManagedObject](#) ()
Default constructor.
- virtual [~ManagedObject](#) ()
Destructor.
- [PPtr< ManagedObject > getPtr](#) () const
Returns a managed pointer 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]

Destructor.

Upon destruction makes sure this object is unregistered 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.
- [~MasterMemoryManager \(\)](#)
Deconstructor.
- `const ObjectRecord & registerObject (ManagedObject *ptr)`
Registers a new object to the default manager.
- `int registerManager (MemoryManager *manager)`
Registers a new memory manager and assigns 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 updateTable (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::getMasterMemoryManager\(\)](#).

This object always owns an instance of [RefCountMemoryManager](#) as its default manager. To access use `getManagerAtIndex (0)`.

6.32.2 Member Function Documentation

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

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

Parameters

<i>manager</i>	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

<i>index</i>	The index of the memory manager to be unregistered.
--------------	---

6.32.2.3 void kfoundation::MasterMemoryManager::updateTable (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:

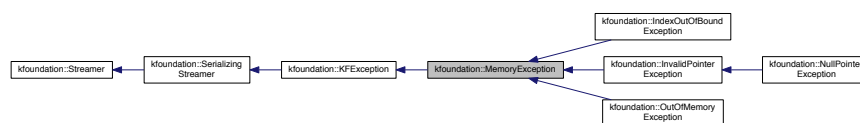
- MasterMemoryManager.h
- MasterMemoryManager.cpp

6.33 kfoundation::MemoryException Class Reference

Used to throw exeptions related to memory.

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

Inheritance diagram for kfoundation::MemoryException:

**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:

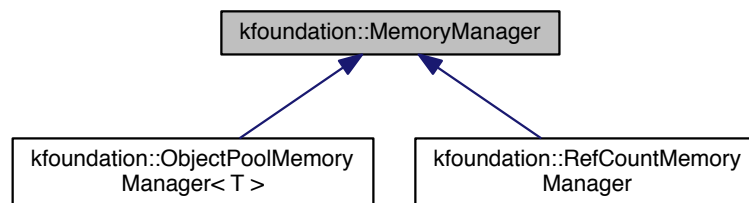
- [MemoryException.h](#)
- [MemoryException.cpp](#)

6.34 kfoundation::MemoryManager Class Reference

Abstract interface to be implemented by all memory managers.

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

Inheritance diagram for kfoundation::MemoryManager:



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 matches, 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](#) ()
Destructor.
- 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

<i>isShared</i>	Should be set true if the mutex is shared between multiple dynamic libraries (shared objects). Default value is <code>false</code> .
-----------------	---

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

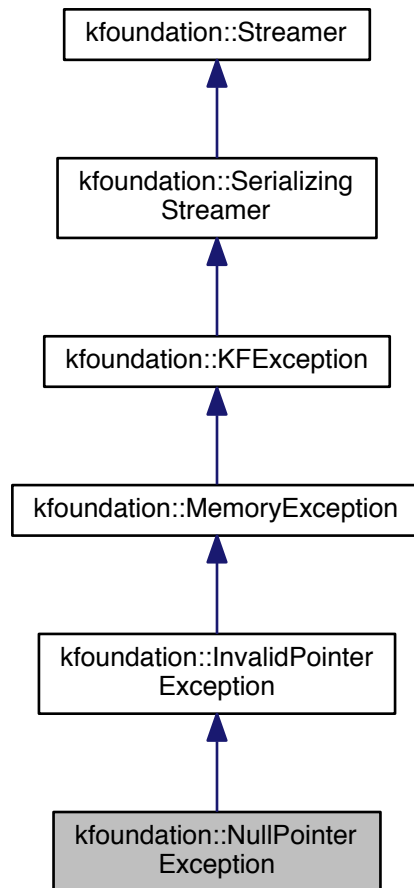
- `Mutex.h`
- `Mutex.cpp`

6.36 kfoundation::NullPointerException Class Reference

Thrown on attempt to access to a null pointer.

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

Inheritance diagram for kfoundation::NullPointerException:



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:

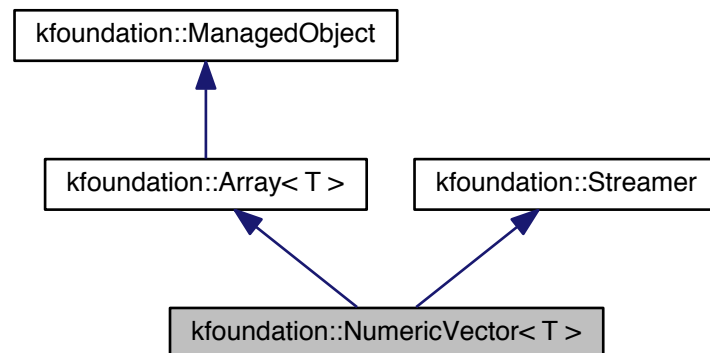
- `NullPointerException.h`
- `NullPointerException.cpp`

6.37 kfoundation::NumericVector< T > Class Template Reference

A subclass of [Array](#), adds numeric operations.

```
#include <kfounadtion/NumericVector.h>
```

Inheritance diagram for kfoundation::NumericVector< T >:



Public Member Functions

- [NumericVector](#) ()
Default constructor, creates an empty array.
- [NumericVector](#) (T *values, [kf_int32_t](#) size)
Constructor, creates a new [NumericVector](#) 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 [NumericVector](#) 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 [NumericVector](#) 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, creates a new [NumericVector](#) copying the specified elements from a C-style array.

Parameters

<i>values</i>	The beginning of the C-style array containing values to be copied.
<i>size</i>	The number of elements to be copied.

6.37.3 Member Function Documentation

6.37.3.1 `template<typename T> Ptr< NumericVector< T > > kfoundation::NumericVector< T >::add (const Ptr< NumericVector< T > > & other) const`

Returns the pointer to a new [NumericVector](#) representing the summation of this vector with the given one.

Parameters

<i>other</i>	The vector to add this one to.
--------------	--------------------------------

6.37.3.2 `template<typename T> Ptr< NumericVector< T > > kfoundation::NumericVector< T >::mul (const T & coef) const`

Resturns the pointer to a new [NumericVector](#) resulted from multiplying this vector to the given scalar value.

Parameters

<i>coef</i>	The scalar value to multiply this vector to.
-------------	--

6.37.3.3 `template<typename T> Ptr< NumericVector< T > > kfoundation::NumericVector< 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

<i>str</i>	The string to parse.
------------	----------------------

6.37.3.4 `template<typename T> Ptr< NumericVector< T > > kfoundation::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.

Parameters

<i>other</i>	The vector to subtract from this one.
--------------	---------------------------------------

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

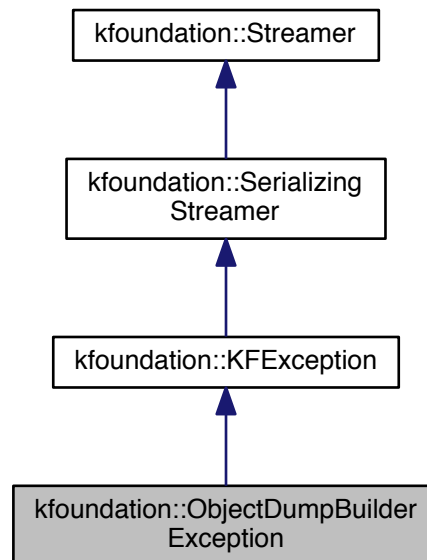
- NumericVectorDecl.h
- NumericVector.h

6.38 kfoundation::ObjectDumpBuilderException Class Reference

Thrown when [ObjectSerializer](#) is used in an invalid way.

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

Inheritance diagram for kfoundation::ObjectDumpBuilderException:



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

<i>message</i>	A message describing the problem.
----------------	-----------------------------------

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

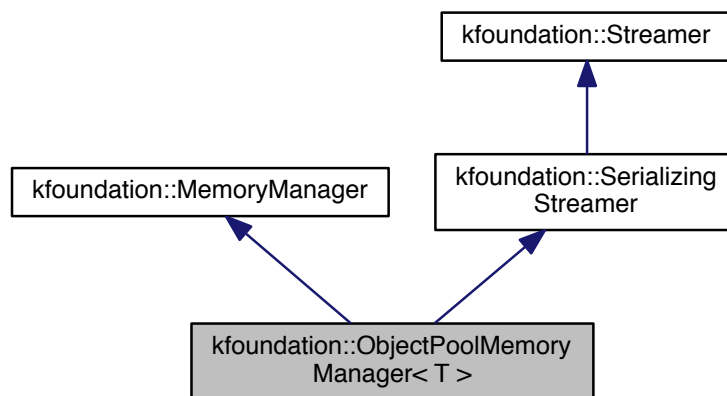
- ObjectSerializer.h
- ObjectSerializer.cpp

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>
```

Inheritance diagram for kfoundation::ObjectPoolMemoryManager< T >:



Public Member Functions

- [ObjectPoolMemoryManager](#) (const int initialCapacity, const int growthRate)
Constructor.
- [~ObjectPoolMemoryManager](#) ()
Destructor.
- [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 matches, 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

<i>initialCapacity</i>	Initial capacity.
<i>growthRate</i>	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 >::~ObjectPoolMemoryManager ()`

Deconstructor.

Deconstructs all objects in the pool internally.

6.39.3 Member Function Documentation

6.39.3.1 `template<typename T > ObjectRecord * kfoundation::ObjectPoolMemoryManager< T >::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](#).

6.39.3.2 `template<typename T > kf_int32_t kfoundation::ObjectPoolMemoryManager< T >::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](#).

6.39.3.3 `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:

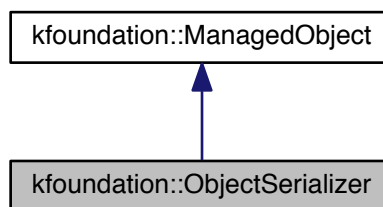
- `MemoryManager.h`

6.41 kfoundation::ObjectSerializer Class Reference

Provides APIs to serialize an object.

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

Inheritance diagram for `kfoundation::ObjectSerializer`:



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` attribute.
- `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 conjunction 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 exception 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 should be an `endObject()` corresponding to each `object()` and `endCollection()` corresponding to each `collection()`.

An exception 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 syntax. 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_t *outputType*, int *indentUnit*)

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

Parameters

<i>stream</i>	The stream to print the output to.
<i>outputType</i>	The output format.
<i>indentUnit</i>	Number of spaces for each indentation 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

<i>stream</i>	The stream to print the output to.
<i>outputType</i>	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

<i>name</i>	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

<i>className</i>	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

<i>ref</i>	The field to be printed.
------------	--------------------------

Returns

Pointer to self.

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

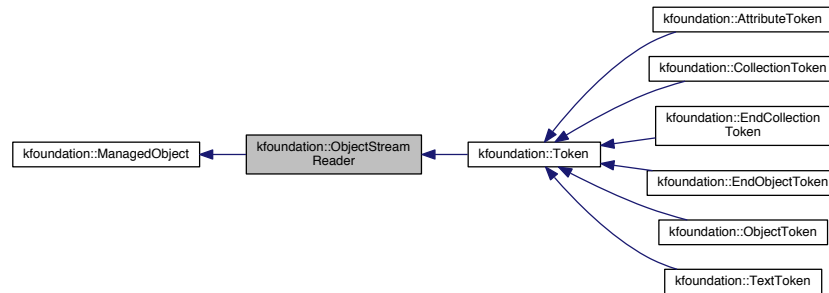
- ObjectSerializer.h
- ObjectSerializer.cpp

6.42 kfoundation::ObjectStreamReader Class Reference

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

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

Inheritance diagram for kfoundation::ObjectStreamReader:



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 objects from a 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:

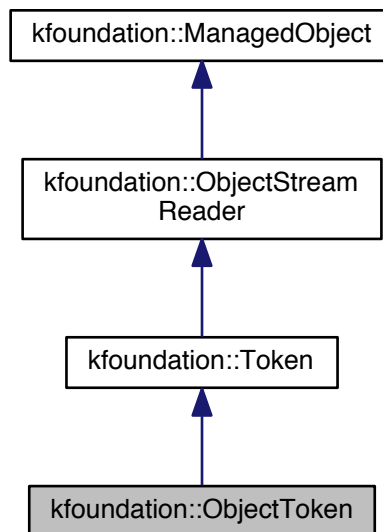
- ObjectStreamReader.h
- ObjectStreamReader.cpp

6.43 kfoundation::ObjectToken Class Reference

Represents beginning of an object in the parsed stream.

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


Inheritance diagram for kfoundation::ObjectToken:



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 [throwInvalidClass](#) () 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

<i>name</i>	The name to check against.
-------------	----------------------------

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

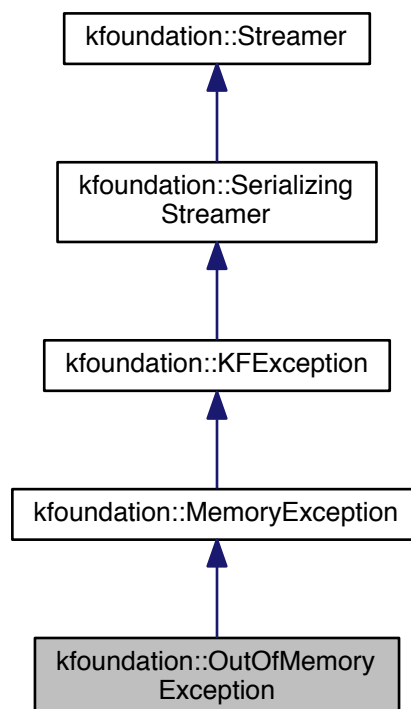
- ObjectStreamReader.h
- ObjectStreamReader.cpp

6.44 kfoundation::OutOfMemoryException Class Reference

Thrown on out of memory.

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

Inheritance diagram for kfoundation::OutOfMemoryException:



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

<i>message</i>	The message to be displayed once the exception is thrown.
----------------	---

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

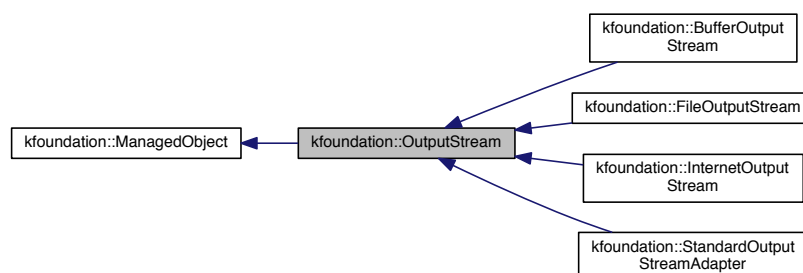
- OutOfMemoryException.h
- OutOfMemoryException.cpp

6.45 kfoundation::OutputStream Class Reference

Abstract interface for all output streams.

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

Inheritance diagram for kfoundation::OutputStream:



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 interface 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::StandardOutputStreamAdapter](#), 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

<i>buffer</i>	The octets to write.
<i>nOctets</i>	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

<i>octet</i>	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

<i>is</i>	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:

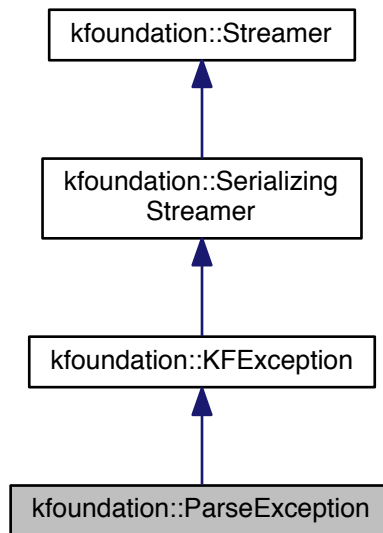
- [OutputStream.h](#)

6.46 kfoundation::ParseException Class Reference

Thrown when a parsing error happens.

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

Inheritance diagram for kfoundation::ParseException:



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 [CodeRange](#) to report when thrown.
- [~ParseException](#) () throw ()
Deconstructor.
- 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 beginning 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

<i>message</i>	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

<i>message</i>	A message describing the error.
<i>The</i>	location of the parsed stream in which the error is observed.

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

Constructor, stores the given string message and [CodeRange](#) to report when thrown.

Parameters

<i>message</i>	A message describing the error.
<i>range</i>	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:

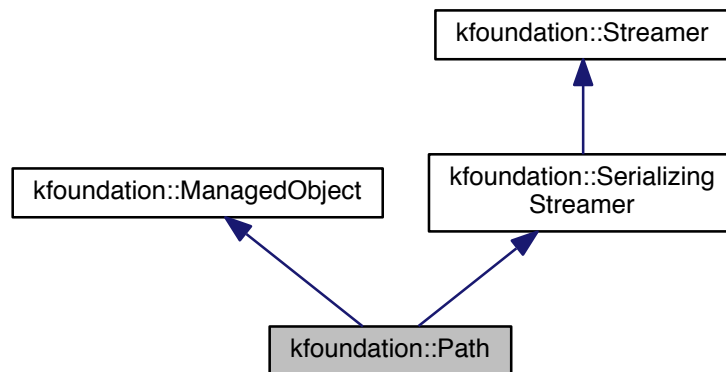
- `ParseException.h`
- `ParseException.cpp`

6.47 kfoundation::Path Class Reference

Used to represent and manipulate file and directory pathnames.

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

Inheritance diagram for kfoundation::Path:



Public Member Functions

- [Path](#) (const string &str)
Constructor.
- [~Path](#) ()
Destructor.
- 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 [getExtension](#) () 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 extension.
- [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

<i>str</i>	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

<code>os</code>	The stream to print to.
-----------------	-------------------------

Reimplemented from [kfoundation::SerializingStreamer](#).

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

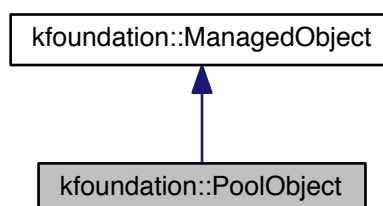
- Path.h
- Path.cpp

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>
```

Inheritance diagram for kfoundation::PoolObject:



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]

Cleanup 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:

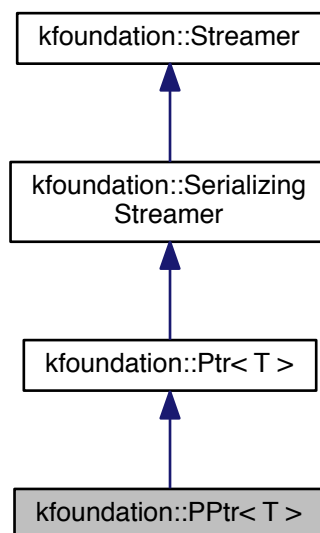
- `ManagedObject.h`
- `ManagedObject.cpp`

6.49 `kfoundation::PPtr< T >` Class Template Reference

Passive pointer, will not release and retain automatically.

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

Inheritance diagram for `kfoundation::PPtr< T >`:



Public Member Functions

- `PPtr ()`

Default constructor, creates a passive NULL-pointer.

- **PPtr** (const **PPtr**< T > &pptr)
Copy constructor.
- **PPtr** (const **P**tr< T > &aptr)
Copy constructor from nonpassive pointer.
- **PPtr** (const **S**Ptr< 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 beginning and the end of the scope of definition 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

<i>obj</i>	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

<i>obj</i>	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

<i>obj</i>	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

<i>obj</i>	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:

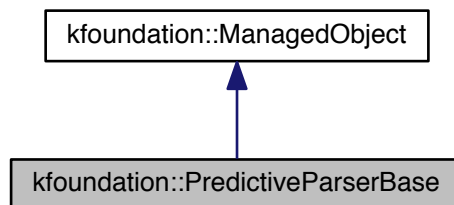
- ObjectPoolMemoryManagerDecl.h
- PtrDecl.h
- Ptr.h

6.50 kfoundation::PredictiveParserBase Class Reference

Packs ample of basic functionalities to implement any predictive parser.

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

Inheritance diagram for kfoundation::PredictiveParserBase:



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)
Thes if any of the given characters is next in the stream.
- bool [testAlphabet](#) ()
Thes 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 equivalent 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 equivalent 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 beginning 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 its equivalent integer value.
- size_t [readNumber](#) (double &output)
Checks if the next character(s) represents a real number and if so, reads and converts it to its equivalent 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 immediately after the given escape 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 beginning 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 overridden 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 failed 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

<i>input</i>	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

<i>ch</i>	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 'isAphabet(ch) || ch == ' _ '.

Parameters

<i>ch</i>	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

<i>ch</i>	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

<i>storage</i>	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

<i>storage</i>	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

<i>t</i>	The character to read until.
<i>storage</i>	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 immediately after the given escape character.

Parameters

<i>t</i>	The character to read until.
<i>escape</i>	The escape character.
<i>storage</i>	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

<i>storage</i>	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

<i>str</i>	The sequence of characters to read until.
<i>storage</i>	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

<i>storage</i>	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

<i>storage</i>	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

<i>storage</i>	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

<i>ch</i>	Output, will be assigned with the read character.
<i>octets</i>	If not <code>NULL</code> , the read octets are written on the buffer pointed by this argument. Default value is <code>NULL</code> .

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 equivalent numeric value.

Parameters

<i>ch</i>	Output, will be assigned to the read character.
<i>octets</i>	If not <code>NULL</code> , the read octets are written on the buffer pointed by this argument. Default value is <code>NULL</code> .

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

<i>ch</i>	Output, will be assigned to the character read.
<i>octets</i>	If not <code>NULL</code> , the read octets are written on the buffer pointed by this argument. Default value is <code>NULL</code> .

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

<i>t</i>	The character to be expected next in the stream.
<i>octets</i>	If not <code>NULL</code> , the read octets are written on the buffer pointed by this argument. Default value is <code>NULL</code> .

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

<i>chars</i>	The list of character to check against.
<i>n</i>	The number of characters in the given list.
<i>octets</i>	If not <code>NULL</code> , the read octets are written on the buffer pointed by this argument. Default value is <code>NULL</code> .

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 [isValidIdentifierBeginChar\(\)](#) and [isValidIdentifierChar\(\)](#) to customize the behavior of this method.

Parameters

<i>storage</i>	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 beginning character, and if so, reads it.

Override [isValidIdentifierBeginChar\(\)](#) to customize the behavior of this method.

Parameters

<i>ch</i>	Output, will be assigned to the read character.
<i>octets</i>	If not <code>NULL</code> , the read octets are written on the buffer pointed by this argument. Default value is <code>NULL</code> .

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 [isValidIdentifierBeginChar\(\)](#) to customize the behavior of this method.

Parameters

<i>ch</i>	Output, will be assigned to the read character.
<i>octets</i>	If not <code>NULL</code> , the read octets are written on the buffer pointed by this argument. Default value is <code>NULL</code> .

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

<i>storage</i>	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 equivalent integer value.

Parameters

<i>output</i>	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

<i>output</i>	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

<i>ch</i>	Output, will be assigned with the read character.
<i>octets</i>	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 equivalent numeric value.

Parameters

<i>digit</i>	Output, will be assigned to the numeric equivalent 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

<i>str</i>	The sequence of characters to check against.
------------	--

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 customize the behavior of this function.

Returns

The number of read octets.

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

<i>ch</i>	The character to test.
-----------	------------------------

6.50.3.32 `bool kfoundation::PredictiveParserBase::testChar (const wchar_t * chars, const int & n)`

Test if any of the given characters is next in the stream.

Parameters

<i>chars</i>	The list of characters to test against.
<i>n</i>	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

<i>str</i>	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:

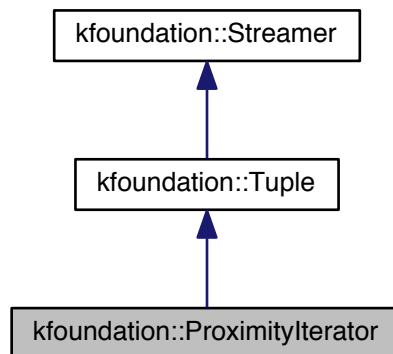
- PredictiveParserBase.h
- PredictiveParserBase.cpp

6.51 kfoundation::ProximityIterator Class Reference

Iterates the proximity of a desired point.

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

Inheritance diagram for `kfoundation::ProximityIterator`:



Public Member Functions

- [ProximityIterator](#) (int radius)
Constructor, sets the rectangular radius around the point to be iterated.
- [ProximityIterator](#) (int radius, const [Tuple](#) ¢er)
Constructor, sets the point and the rectangular radius around it to be iterated.
- [ProximityIterator](#) & [centerAt](#) (const [Tuple](#) ¢er)
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:

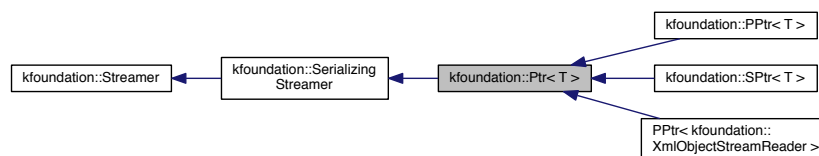
- ProximityIterator.h
- ProximityIterator.cpp

6.52 kfoundation::Ptr< T > Class Template Reference

Managed pointer to a class of given template type.

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

Inheritance diagram for kfoundation::Ptr< T >:



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 ()`
Destructor.
- `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 pointer 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 pointer.

```
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);
```

Similiarly, there is an elegant way to check the type of a managed pointer.

```
if(myObject.ISA(MySuperClass)) {
    LOG << "I Love KFoundation <3" << EL;
}
```

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 certain 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 registry on each access. To make it fast, a novel fast algorithm with O(1) time complexity is developed to do the task.

On rare occasions 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

<i>obj</i>	Pointer to a ManagedObject or one of its subclasses, or NULL.
<i>trace</i>	Used for debugging. Causes a lot of information to be printed everytime the pointer is modified.

6.52.2.2 `template<typename T> kfoundation::Ptr< T >::Ptr (const Ptr< T > & other) [inline]`

Copy constructor.

Parameters

<i>other</i>	The other pointer to be copied.
--------------	---------------------------------

6.52.2.3 `template<typename T> kfoundation::Ptr< T >::~~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 invalid (or NULL) the returned value will be negative.

6.52.3.4 `template<typename T> template<typename K> bool kfoundation::Ptr< T >::isa () const [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++ equivalent.

6.52.3.8 `template<typename T> Ptr< T > & kfoundation::Ptr< T >::operator= (const Ptr< T > & other) [inline]`

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 pointer 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

<i>replacement</i>	The new object to point to. It should be an instance of ManagedObject or one of its subclasses, 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

<i>replacement</i>	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 to 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:

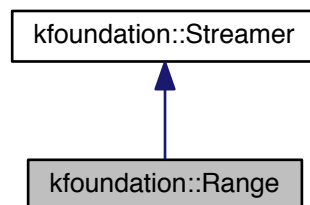
- PtrDecl.h
- Ptr.h

6.53 kfoundation::Range Class Reference

Represents a range in n-dimensional space.

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

Inheritance diagram for kfoundation::Range:



Public Member Functions

- [Range](#) ()
Default constructor.
- [Range](#) (const [Tuple](#) &begin, const [Tuple](#) &end)
Constructor, creates a range with the given beginning and end.
- [Range](#) (const [Tuple](#) &end)
Constructor, creates a range with its beginning 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 beginning 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 [isAdjacentTo](#) (const [Range](#) &other) const
Checks if this range is adgecent 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 getlterator](#) () 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 subtracting 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

<i>d</i>	The direction of the desired broder.
<i>s</i>	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:

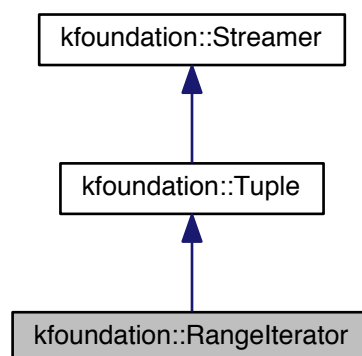
- Range.h
- Range.cpp

6.54 kfoundation::RangeIterator Class Reference

Used to iterate over all points in a given range.

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

Inheritance diagram for kfoundation::RangeIterator:



Public Member Functions

- [RangeIterator](#) (const [Tuple](#) &upperBound)

Constructor, the resulting object iterates points from origin to the given upper bound (exclusive).

- [RangeIterator](#) (const [Tuple](#) &lowerBound, const [Tuple](#) &upperBound)

Constructor, the resulting object iterates points from the given lower bound (inclusive) to the given upper bound (exclusive).

- [RangeIterator](#) (const [Range](#) &range)

Copy constructor.

- [~RangeIterator](#) ()

Destructor.

- const [RangeIterator](#) & first ()

Resets this iterator.

- const [RangeIterator](#) & 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 beginning of the range is inclusive and the end of it is exclusive. Usage:

```
Range r(Tuple1D(10, 10), Tuple2D(20, 30));
for(RangeIterator(r); r.hasMore(); r.next()) {
    Tuple point = r;
    ... do somethingf with r ...
}
```

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

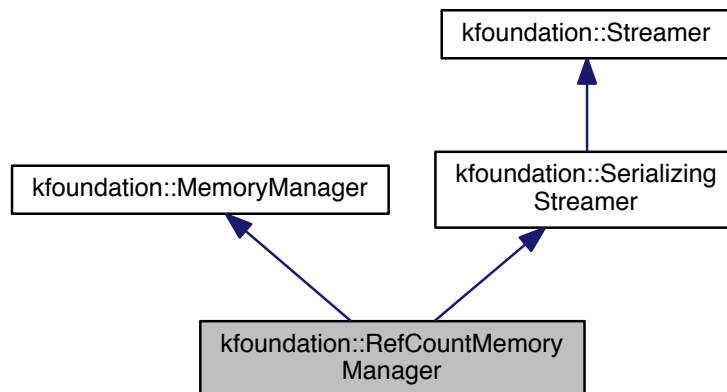
- [RangeIterator.h](#)
- [RangeIterator.cpp](#)

6.55 kfoundation::RefCountMemoryManager Class Reference

Reference counting memory manager.

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

Inheritance diagram for kfoundation::RefCountMemoryManager:



Public Member Functions

- [RefCountMemoryManager](#) ([MasterMemoryManager](#) *master)
Constructor.
- [~RefCountMemoryManager](#) ()
Destructor.
- 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 matches, 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](#) > serializer) 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:

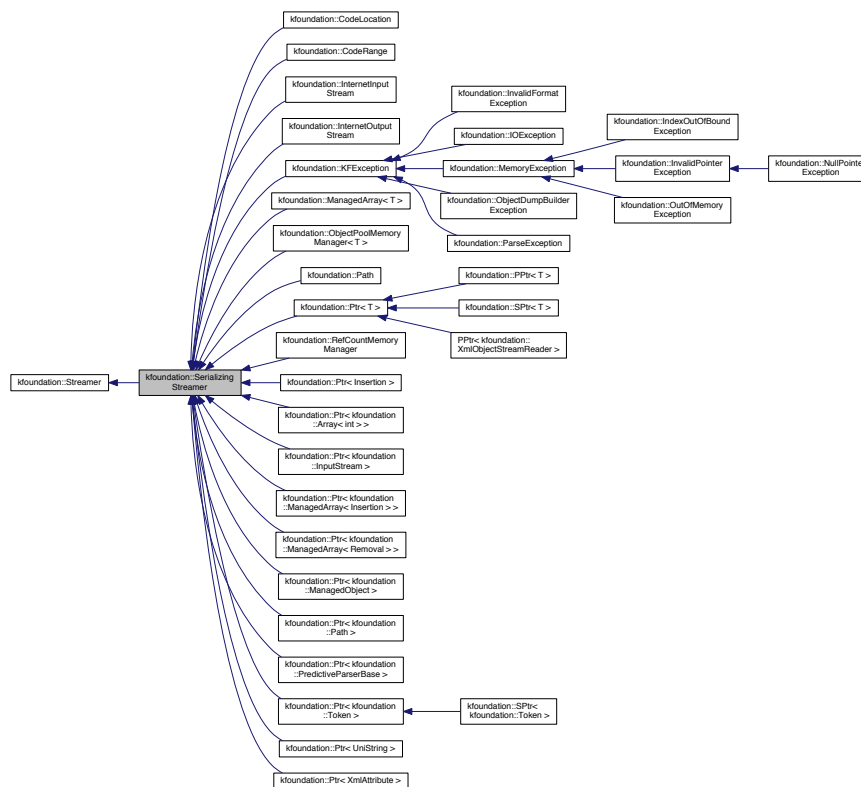
- `RefCountMemoryManager.h`
- `RefCountMemoryManager.cpp`

6.56 kfoundation::SerializingStreamer Class Reference

Objects implementing this class can be serialized into any format allowed by [ObjectSerializer](#).

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

Inheritance diagram for kfoundation::SerializingStreamer:



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

<i>os</i>	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

<i>os</i>	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

<i>os</i>	The stream to print to.
-----------	-------------------------

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

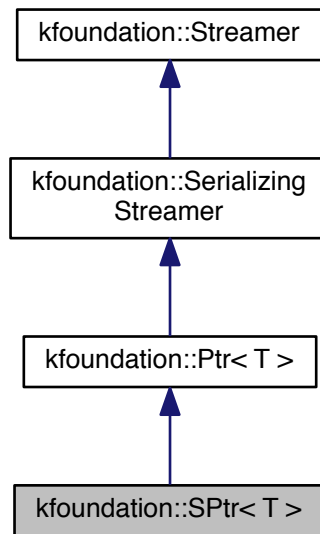
- [SerializingStreamer.h](#)
- [SerializingStreamer.cpp](#)

6.57 kfoundation::SPtr< T > Class Template Reference

Static pointer, makes the pointed object immortal.

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

Inheritance diagram for `kfoundation::SPtr< T >`:



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

<i>obj</i>	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:

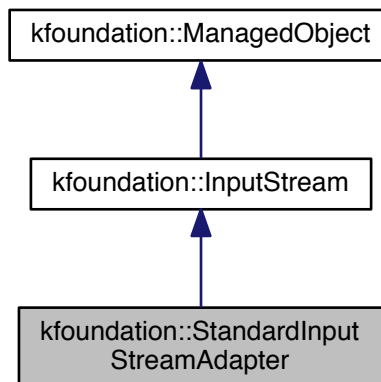
- `PtrDecl.h`
- `Ptr.h`

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>
```

Inheritance diagram for `kfoundation::StandardInputStreamAdapter`:



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 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

<i>buffer</i>	The octets to read.
<i>nOctets</i>	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

<i>nOctets</i>	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:

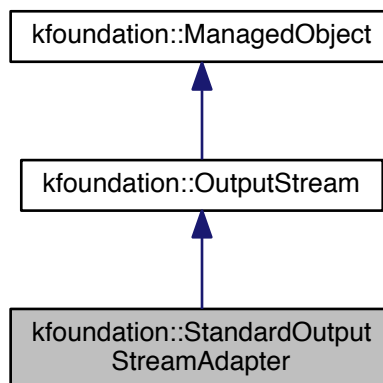
- StandardInputStreamAdapter.h
- StandardInputStreamAdapter.cpp

6.59 kfoundation::StandardOutputStreamAdapter Class Reference

KFoundation wrapper for C++ `ostream`.

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

Inheritance diagram for `kfoundation::StandardOutputStreamAdapter`:



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

<i>buffer</i>	The octets to write.
<i>nOctets</i>	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

<i>octet</i>	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

<i>is</i>	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 destroyed 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\(\)](#) instead.

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

<i>stream</i>	The XML stream to read the object from.
---------------	---

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

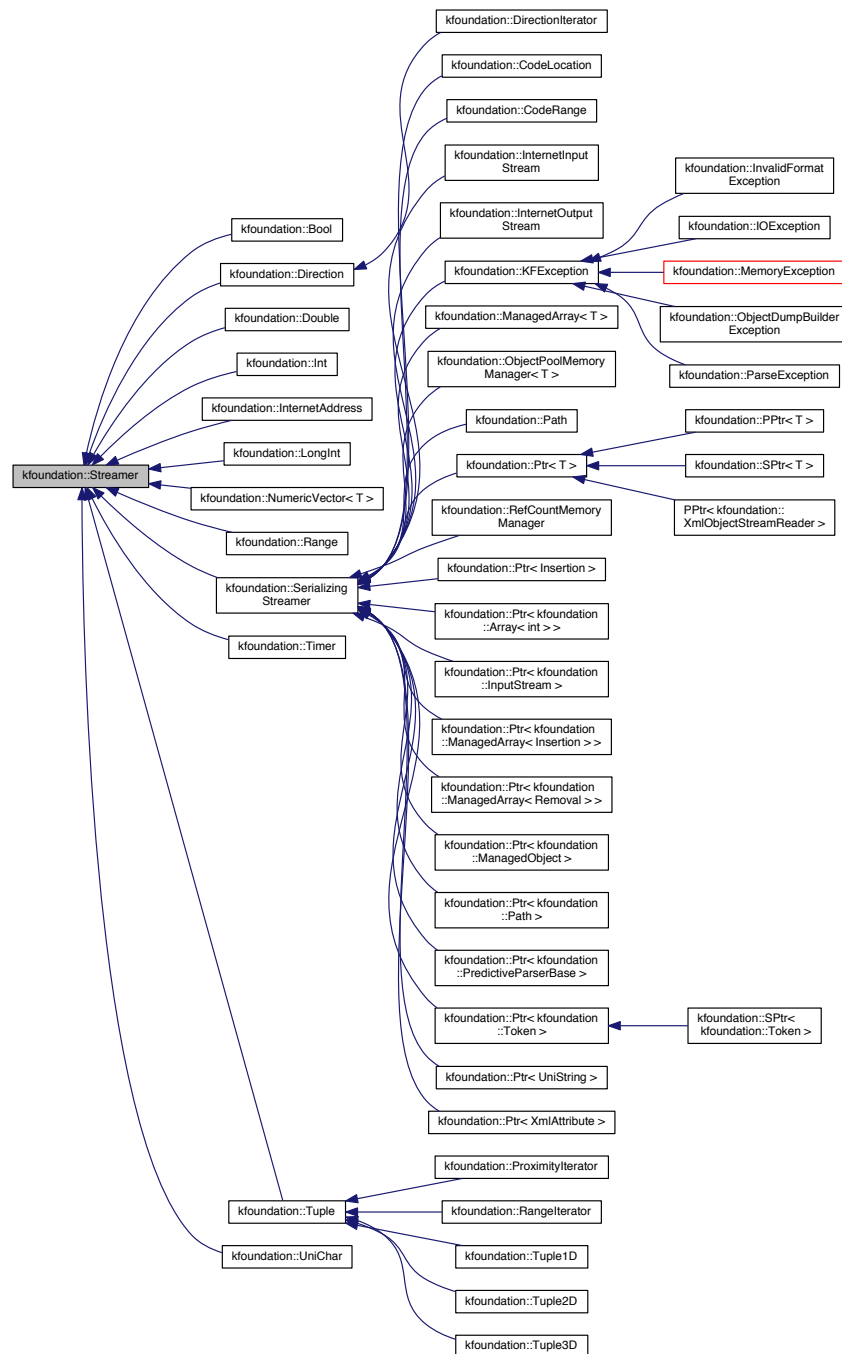
- [StreamDeserializer.h](#)
- [StreamDeserializer.cpp](#)

6.62 kfoundation::Streamer Class Reference

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

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

Inheritance diagram for kfoundation::Streamer:



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 themselves 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:

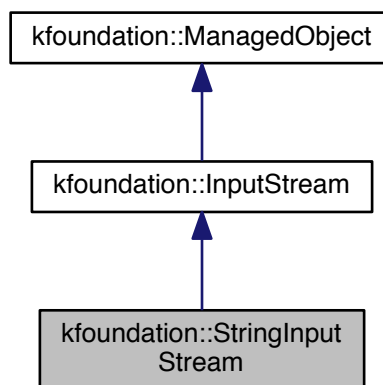
- Streamer.h
- Streamer.cpp

6.63 kfoundation::StringInputStream Class Reference

Input stream to read from string.

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

Inheritance diagram for `kfoundation::StringInputStream`:



Public Member Functions

- [StringInputStream](#) (const string &str)
Constructor, uses the given string as input.
- [~StringInputStream](#) ()
Destructor.
- [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

<i>buffer</i>	The octets to read.
<i>nOctets</i>	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

<i>nOctets</i>	The desired number of octets to skip.
----------------	---------------------------------------

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 system 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 underlying operating system.
- static [Logger](#) & [getLogger](#) ()
Returns the default system logger.
- static void [sleep](#) (const int msec)
Causes the current thread to sleep for the given number of milliseconds.
- 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 getCurrentTimeInMilliseconds ()`
Returns the current time in milliseconds 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

<i>command</i>	The command to be executed.
<i>args</i>	An array of C strings, each containing a signle argument.
<i>argc</i>	The number of elements in args.

6.64.3.2 `kf_int64_t kfoundation::System::getCurrentTimeInMilliseconds () [static]`

Returns the current time in milliseconds from the standars origin time.

The origin time is usually midnight 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 thread to sleep for the given number of milliseconds.

Most platforms provide [sleep\(\)](#) which works with 1-second resolution, and [usleep\(\)](#) which works with 1-microsecond resolution. [System::sleep\(\)](#) provides millisecond resolution which makes sense in most devices, and has the same interface as `Java.System.sleep()`. Can be used in conjunction with [System::getCurrentTimeInMilliseconds\(\)](#) without the fuss of `timeval` struct.

Parameters

<i>msecs</i>	The number of milliseconds to sleep.
--------------	--------------------------------------

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

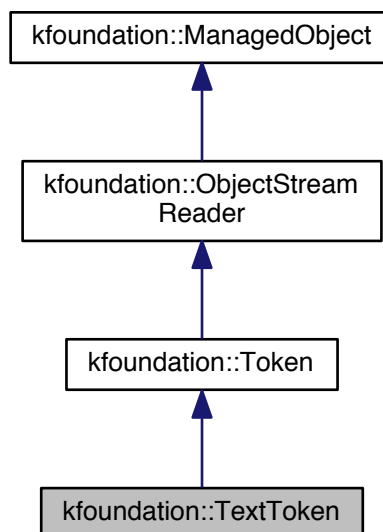
- `System.h`
- `System.cpp`

6.65 kfoundation::TextToken Class Reference

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

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

Inheritance diagram for `kfoundation::TextToken`:



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:

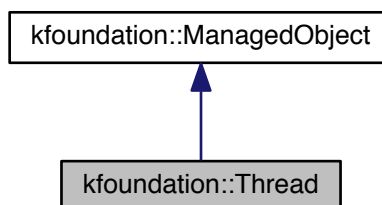
- ObjectStreamReader.h
- ObjectStreamReader.cpp

6.66 kfoundation::Thread Class Reference

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

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

Inheritance diagram for kfoundation::Thread:



Public Member Functions

- [Thread](#) ()
Default constructor.
- [Thread](#) (const string &name)
Constructs a named thread.
- [~Thread](#) ()
Destructor.
- 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 thread 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:

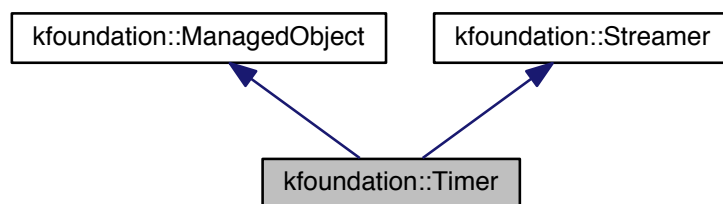
- Thread.h
- Thread.cpp

6.67 kfoundation::Timer Class Reference

Utility class to measure execution time of a code fragment.

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

Inheritance diagram for kfoundation::Timer:



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;
```

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:

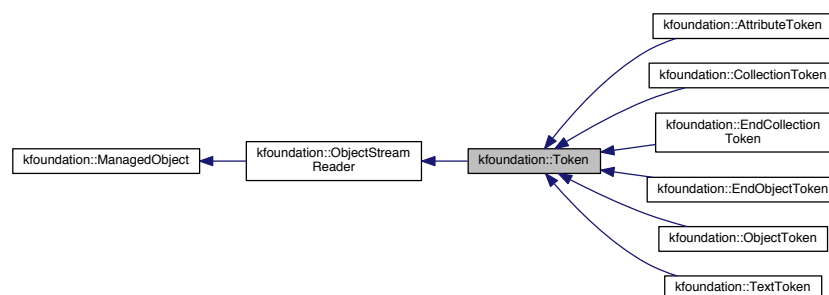
- Timer.h
- Timer.cpp

6.68 kfoundation::Token Class Reference

Represents a token in a stream.

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

Inheritance diagram for kfoundation::Token:



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 if the type of this token matches the given argument, and if not, produces a [ParseException](#) explaining 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 beginning 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 beginning 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:

```

void deserialize(PPtr<ObjectToken> headToken) {
    headToken->validateClass("MyClass");
    Ptr<Token> token = headToken->next();
    if(token.is(Token::ATTRIBUTE)) {
        PPtr<AttributeToken> attrib = token.AS(Attribute);
        if(attrib->checkName("attrib1")) {
            _attrib1 = attrib->getValue();
        } else if(attrib->checkName("attrib2")) {
            _attrib2 = attrib->getValue();
        } else {
            attrib->throwInvalidName();
        }
    } else {
        token->validateType(Token::OBJECT);
        ...
    }
    token->next()->validateType(END_OBJECT);
}

```

validateXXX() methods cause an exception 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

<i>cr</i>	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 beginning and end of this token.

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

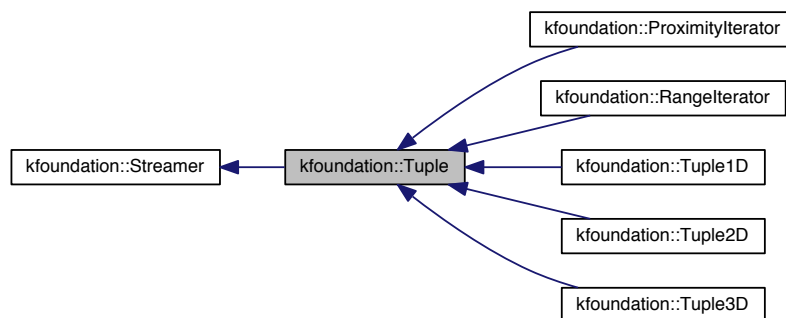
- ObjectStreamReader.h
- ObjectStreamReader.cpp

6.69 kfoundation::Tuple Class Reference

Represents a point in n-dimensional space.

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

Inheritance diagram for kfoundation::Tuple:



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`
Subtraction operator.
 - `Tuple operator+ (const int n) const`
Adds the given scalar to all elements of this tuple.
 - `Tuple operator- (const int n) const`
Subtracts 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.

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 given tuple.

Parameters

<i>other</i>	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

<i>other</i>	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

<i>other</i>	The scalar to multiply.
--------------	-------------------------

Returns

The resulting tuple.

6.69.2.5 Tuple kfoundation::Tuple::operator+ (const Tuple & *other*) const

Addition operator.

Parameters

<i>other</i>	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

<i>other</i>	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

<i>other</i>	The tuple to subtract.
--------------	------------------------

Returns

The resulting tuple.

6.69.2.9 Tuple kfoundation::Tuple::operator- (const int *n*) const

Subtracts the given scalar from all elements of this tuple.

Parameters

<i>other</i>	The scalar to subtract.
--------------	-------------------------

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

<i>other</i>	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

<i>other</i>	The tuple to divide to.
--------------	-------------------------

Returns

The resulting tuple.

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

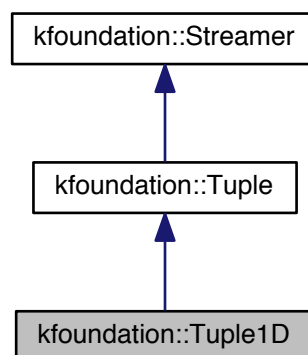
- Tuple.h
- Tuple.cpp

6.70 kfoundation::Tuple1D Class Reference

1-dimensional specialization of [Tuple](#).

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

Inheritance diagram for kfoundation::Tuple1D:



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:

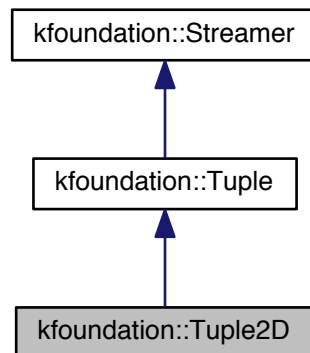
- [Tuple.h](#)
- [Tuple.cpp](#)

6.71 [kfoundation::Tuple2D](#) Class Reference

2-dimensional specialization of [Tuple](#).

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

Inheritance diagram for kfoundation::Tuple2D:



Public Member Functions

- [Tuple2D](#) ()
Default 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

<i>x</i>	Value for the first element.
<i>y</i>	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

<i>x</i>	Value for the first element.
<i>y</i>	Value for the second element.

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

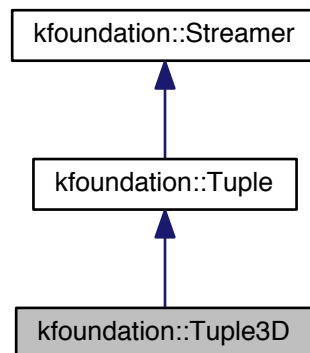
- Tuple.h
- Tuple.cpp

6.72 kfoundation::Tuple3D Class Reference

3-dimensional specialization of [Tuple](#).

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

Inheritance diagram for kfoundation::Tuple3D:



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

<i>x</i>	The value for the first element.
<i>y</i>	The value for the second element.
<i>z</i>	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

<i>x</i>	The value for the first element.
<i>y</i>	The value for the second element.
<i>z</i>	The value for the third element.

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

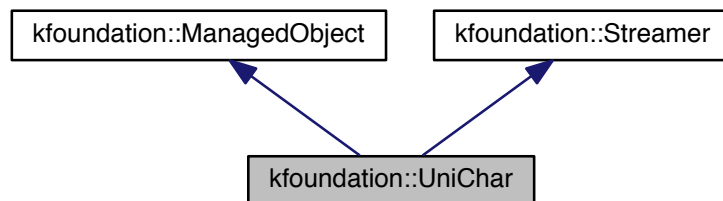
- [Tuple.h](#)
- [Tuple.cpp](#)

6.73 kfoundation::UniChar Class Reference

Wrapper class for unicode character.

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

Inheritance diagram for kfoundation::UniChar:



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 character 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` object representing the lowercase equivalent of this object.
- `Ptr< UniChar > duplicateToUpperCase () const`
Creates a new `UniChar` object representing the uppercase equivalent 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 representation.

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

<i>value</i>	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

<i>value</i>	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 character encoded in UTF-8 from the given buffer.

Parameters

<i>buffer</i>	The beginning 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

<i>stream</i>	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

<i>input</i>	The beginning of the buffer to read from.
--------------	---

<i>output</i>	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

<i>stream</i>	The stream to read from.
<i>output</i>	The output to write to.
<i>readBytes</i>	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

<i>value</i>	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

<i>ch</i>	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

<i>ch</i>	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

<i>buffer</i>	The beginning of the buffer to write 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

<i>ch</i>	The character to write.
<i>buffer</i>	The beginning of the buffer to write on.

Returns

The number of octets written.

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

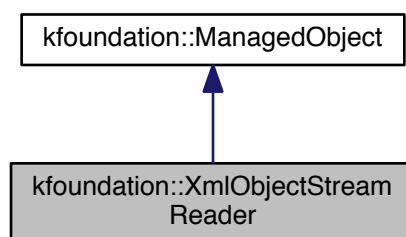
- UniChar.h
- UniChar.cpp

6.74 kfoundation::XmlObjectStreamReader Class Reference

[ObjectStreamReader](#) to deserialize XML streams.

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

Inheritance diagram for kfoundation::XmlObjectStreamReader:



Public Member Functions

- [XmlObjectStreamReader](#) ([PPtr](#)< [InputStream](#) > input)
Constructor.
- [~XmlObjectStreamReader](#) ()
Destructor.
- [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

6.74.2.1 kfoundation::XmlObjectStreamReader::XmlObjectStreamReader (*PPtr*< *InputStream* > *input*)

Constructor.

Parameters

<i>input</i>	A stream containing XML representation of an object.
--------------	--

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

- [XmlObjectStreamReader.h](#)
- [XmlObjectStreamReader.cpp](#)

Index

- ~ManagedArray
 - kfoundation::ManagedArray, [89](#)
- ~ManagedObject
 - kfoundation::ManagedObject, [92](#)
- ~ObjectPoolMemoryManager
 - kfoundation::ObjectPoolMemoryManager, [104](#)
- ~Ptr
 - kfoundation::Ptr, [138](#)

- A
 - kfoundation::IPAddress, [68](#)
- APPLE_OS_X
 - kfoundation::System, [162](#)
- AS
 - Memory Management, [18](#)
- add
 - kfoundation::NumericVector, [100](#)
- addChannel
 - kfoundation::Logger, [84](#)
- Array
 - kfoundation::Array, [26](#)
- asAttribute
 - kfoundation::Token, [169](#)
- asCollection
 - kfoundation::Token, [170](#)
- asEndCollection
 - kfoundation::Token, [170](#)
- asEndObject
 - kfoundation::Token, [170](#)
- asObject
 - kfoundation::Token, [170](#)
- at
 - kfoundation::Array, [27](#)
 - kfoundation::ManagedArray, [89](#)

- B
 - kfoundation::IPAddress, [68](#)
- BACK
 - kfoundation::Direction, [48](#)
- bind
 - kfoundation::InternetInputStream, [71](#)
- block
 - kfoundation::Condition, [46](#)
- Bool
 - kfoundation::Bool, [31](#)
- border
 - kfoundation::Range, [143](#)
- BufferInputStream
 - kfoundation::BufferInputStream, [33](#)
- BufferOutputStream

- kfoundation::BufferOutputStream, [36](#)

- C
 - kfoundation::IPAddress, [68](#)
- cast
 - kfoundation::Ptr, [138](#)
- changeExtension
 - kfoundation::Path, [118](#)
- Channel
 - kfoundation::Logger::Channel, [38](#)
- checkLevel
 - kfoundation::Logger::Channel, [39](#)
- checkName
 - kfoundation::Logger::Channel, [40](#)
- class_t
 - kfoundation::IPAddress, [68](#)
- close
 - kfoundation::BufferOutputStream, [37](#)
 - kfoundation::FileOutputStream, [58](#)
 - kfoundation::InternetOutputStream, [75](#)
 - kfoundation::OutputStream, [114](#)
 - kfoundation::StandardOutputStreamAdapter, [154](#)
- CodeRange
 - kfoundation::CodeRange, [44](#)
- codeRange
 - kfoundation::Token, [171](#)
- collection
 - kfoundation::ObjectSerializer, [108](#)
- component_t
 - kfoundation::Direction, [48](#)
- connect
 - kfoundation::InternetOutputStream, [75](#)
- contains
 - kfoundation::Array, [27](#)
- copyWithPort
 - kfoundation::IPAddress, [69](#)
- D
 - kfoundation::IPAddress, [68](#)
- DECIMAL
 - kfoundation::LongInt, [86](#)
- DUMP
 - kfoundation::ObjectSerializer, [108](#)
- del
 - kfoundation::Ptr, [138](#)
- DirectionIterator
 - kfoundation::DirectionIterator, [49](#)
- Double
 - kfoundation::Double, [51](#)

E

kfoundation::IPAddress, 68

EQUAL

Macros and Typedefs, 16

ERR

kfoundation::Logger, 84

encoding_t

kfoundation::LongInt, 86

endCollection

kfoundation::ObjectSerializer, 108

endObject

kfoundation::ObjectSerializer, 108

Exceptions, 17

exec

kfoundation::System, 162

FORTH

kfoundation::Direction, 48

FREE_BSD

kfoundation::System, 162

FileInputStream

kfoundation::FileInputStream, 55

FileOutputStream

kfoundation::FileOutputStream, 58

finalize

kfoundation::PoolObject, 120

GREATER

Macros and Typedefs, 16

get

kfoundation::Bool, 31

kfoundation::Double, 51

kfoundation::Int, 66

kfoundation::LongInt, 87

kfoundation::UniChar, 183

getBegin

kfoundation::ParseException, 116

getChannelByName

kfoundation::Logger, 84

getCpuTime

kfoundation::Timer, 167

getCurrentTimeInMilliseconds

kfoundation::System, 162

getData

kfoundation::BufferOutputStream, 37

getEnd

kfoundation::ParseException, 116

getExePath

kfoundation::System, 162

getMessage

kfoundation::KFException, 81

getRetainCount

kfoundation::Ptr, 138

getTable

kfoundation::MemoryManager, 95

kfoundation::ObjectPoolMemoryManager, 104

kfoundation::RefCountMemoryManager, 145

getTableSize

kfoundation::MemoryManager, 96

kfoundation::ObjectPoolMemoryManager, 104

kfoundation::RefCountMemoryManager, 146

HEXADECIMAL

kfoundation::LongInt, 86

hasExtention

kfoundation::Path, 118

IOException

kfoundation::IOException, 79

IS_NULL

Macros and Typedefs, 16

ISA

Memory Management, 18

indexOf

kfoundation::Array, 27

kfoundation::ManagedArray, 89, 90

IndexOutOfBoundsException

kfoundation::IndexOutOfBoundsException, 61

Input/Output, 19

insert

kfoundation::Array, 28

kfoundation::ManagedArray, 90

Int

kfoundation::Int, 66

IPAddress

kfoundation::IPAddress, 68

InternetInputStream

kfoundation::InternetInputStream, 70

InternetOutputStream

kfoundation::InternetOutputStream, 75

InvalidFormatException

kfoundation::InvalidFormatException, 77

isAbsolute

kfoundation::Path, 118

isBound

kfoundation::InternetInputStream, 71

isEmpty

kfoundation::Range, 143

isLocked

kfoundation::FileOutputStream, 58

isMarkSupported

kfoundation::BufferInputStream, 34

kfoundation::FileInputStream, 55

kfoundation::InputStream, 63

kfoundation::InternetInputStream, 71

kfoundation::StandardInputStreamAdapter, 152

kfoundation::StringInputStream, 159

isRunning

kfoundation::Thread, 165

isSilent

kfoundation::Logger::Channel, 40

isSpace

kfoundation::PredictiveParserBase, 124

isTheCurrentThread

kfoundation::Thread, 165

isValid

kfoundation::Ptr, 138

isValidIdentifierBeginChar

- kfoundation::PredictiveParserBase, 125
- isValidIdentifierChar
 - kfoundation::PredictiveParserBase, 125
- isa
 - kfoundation::Ptr, 138
- JSON
 - kfoundation::ObjectSerializer, 108
- KF_EXPORT
 - Macros and Typedefs, 16
- KF_NOP
 - Macros and Typedefs, 16
- KF_SYNCHRONIZED
 - Mutithreading, 20
- KFException
 - kfoundation::KFException, 81
- kf_comparison_t
 - Macros and Typedefs, 16
- kfoundation::Direction
 - BACK, 48
 - FORTH, 48
 - NEUTRAL, 48
- kfoundation::InternetAddress
 - A, 68
 - B, 68
 - C, 68
 - D, 68
 - E, 68
- kfoundation::Logger
 - ERR, 84
 - L1, 84
 - L2, 84
 - L3, 84
 - WRN, 84
- kfoundation::LongInt
 - DECIMAL, 86
 - HEXADECIMAL, 86
- kfoundation::ObjectSerializer
 - DUMP, 108
 - JSON, 108
 - XML, 108
- kfoundation::System
 - APPLE_OS_X, 162
 - FREE_BSD, 162
 - LINUX, 162
 - MACH, 162
 - SOLARIS, 162
 - UNSUPPORTED, 162
- kfoundation::Array
 - Array, 26
 - at, 27
 - contains, 27
 - indexOf, 27
 - insert, 28
 - pop, 28
 - push, 28
 - remove, 28
- kfoundation::Array< T >, 25
- kfoundation::AttributeToken, 29
- kfoundation::Bool, 30
 - Bool, 31
 - get, 31
 - set, 31
 - toString, 31
- kfoundation::BufferInputStream, 31
 - BufferInputStream, 33
 - isMarkSupported, 34
 - mark, 34
 - peek, 34
 - read, 34
 - reset, 35
 - skip, 35
- kfoundation::BufferOutputStream, 35
 - BufferOutputStream, 36
 - close, 37
 - getData, 37
 - write, 37
- kfoundation::CodeLocation, 41
 - set, 42
 - setByteIndex, 42
 - setCharIndex, 42
 - setCol, 43
 - setLine, 43
- kfoundation::CodeRange, 43
 - CodeRange, 44
- kfoundation::CollectionToken, 45
- kfoundation::Condition, 46
 - block, 46
 - release, 46
- kfoundation::Direction, 47
 - component_t, 48
 - set, 48
- kfoundation::DirectionIterator, 48
 - DirectionIterator, 49
- kfoundation::Double, 50
 - Double, 51
 - get, 51
 - parse, 51
 - set, 51
- kfoundation::EndCollectionToken, 51
- kfoundation::EndObjectToken, 53
- kfoundation::FileInputStream, 54
 - FileInputStream, 55
 - isMarkSupported, 55
 - mark, 55
 - peek, 56
 - read, 56
 - reset, 56
 - skip, 56
- kfoundation::FileOutputStream, 57
 - close, 58
 - FileOutputStream, 58
 - isLocked, 58
 - lock, 59
 - truncate, 59
 - unlock, 59

- write, [59](#), [60](#)
- kfoundation::IOException, [79](#)
 - IOException, [79](#)
- kfoundation::IndexOutOfBoundsException, [60](#)
 - IndexOutOfBoundsException, [61](#)
- kfoundation::InputStream, [62](#)
 - isMarkSupported, [63](#)
 - mark, [63](#)
 - peek, [63](#)
 - read, [63](#), [64](#)
 - reset, [64](#)
 - skip, [64](#)
- kfoundation::Int, [65](#)
 - get, [66](#)
 - Int, [66](#)
 - parse, [66](#)
 - set, [66](#)
 - toHexString, [66](#)
- kfoundation::InternetAddress, [67](#)
 - class_t, [68](#)
 - copyWithPort, [69](#)
 - InternetAddress, [68](#)
- kfoundation::InternetInputStream, [69](#)
 - bind, [71](#)
 - InternetInputStream, [70](#)
 - isBound, [71](#)
 - isMarkSupported, [71](#)
 - listen, [71](#)
 - mark, [71](#)
 - peek, [72](#)
 - read, [72](#)
 - reset, [72](#)
 - skip, [72](#)
 - unbind, [74](#)
- kfoundation::InternetOutputStream, [74](#)
 - close, [75](#)
 - connect, [75](#)
 - InternetOutputStream, [75](#)
 - write, [76](#)
- kfoundation::InvalidFormatException, [76](#)
 - InvalidFormatException, [77](#)
- kfoundation::InvalidPointerException, [78](#)
- kfoundation::KFException, [80](#)
 - getMessage, [81](#)
 - KFException, [81](#)
 - serialize, [81](#)
 - setName, [81](#)
- kfoundation::Logger, [81](#)
 - addChannel, [84](#)
 - getChannelByName, [84](#)
 - level_t, [84](#)
 - log, [84](#)
 - setLevel, [85](#)
- kfoundation::Logger::Channel, [38](#)
 - Channel, [38](#)
 - checkLevel, [39](#)
 - checkName, [40](#)
 - isSilent, [40](#)
 - setFormat, [40](#)
 - setLevel, [40](#)
 - setSilent, [40](#)
- kfoundation::Logger::Stream, [155](#)
 - Stream, [155](#)
- kfoundation::LongInt, [85](#)
 - encoding_t, [86](#)
 - get, [87](#)
 - LongInt, [86](#)
 - parse, [87](#)
 - set, [87](#)
- kfoundation::ManagedArray
 - ~ManagedArray, [89](#)
 - at, [89](#)
 - indexOf, [89](#), [90](#)
 - insert, [90](#)
 - ManagedArray, [89](#)
 - pop, [90](#)
 - push, [91](#)
 - remove, [91](#)
 - setSize, [91](#)
- kfoundation::ManagedArray< T >, [87](#)
- kfoundation::ManagedObject, [91](#)
 - ~ManagedObject, [92](#)
- kfoundation::MasterMemoryManager, [93](#)
 - registerManager, [93](#)
 - unregisterManager, [94](#)
 - updateTable, [94](#)
- kfoundation::MemoryException, [94](#)
- kfoundation::MemoryManager, [95](#)
 - getTable, [95](#)
 - getTableSize, [96](#)
 - release, [96](#)
 - remove, [96](#)
- kfoundation::Mutex, [96](#)
 - Mutex, [97](#)
- kfoundation::NullPointerException, [97](#)
- kfoundation::NumericVector
 - add, [100](#)
 - mul, [100](#)
 - NumericVector, [100](#)
 - parseInt, [100](#)
 - sub, [100](#)
- kfoundation::NumericVector< T >, [99](#)
- kfoundation::ObjectDumpBuilderException, [101](#)
 - ObjectDumpBuilderException, [102](#)
- kfoundation::ObjectPoolMemoryManager
 - ~ObjectPoolMemoryManager, [104](#)
 - getTable, [104](#)
 - getTableSize, [104](#)
 - ObjectPoolMemoryManager, [104](#)
 - release, [105](#)
 - remove, [105](#)
- kfoundation::ObjectPoolMemoryManager< T >, [103](#)
- kfoundation::ObjectRecord, [105](#)
- kfoundation::ObjectSerializer, [106](#)
 - collection, [108](#)
 - endCollection, [108](#)

- endObject, 108
- member, 109
- null, 109
- object, 109
- ObjectSerializer, 108
- output_type_t, 108
- kfoundation::ObjectStreamReader, 110
- kfoundation::ObjectToken, 110
 - validateClass, 112
- kfoundation::OutOfMemoryException, 112
 - OutOfMemoryException, 113
- kfoundation::OutputStream, 113
 - close, 114
 - write, 114
- kfoundation::PPtr
 - PPtr, 121
- kfoundation::PPtr< T >, 120
- kfoundation::ParseException, 115
 - getBegin, 116
 - getEnd, 116
 - ParseException, 116
- kfoundation::Path, 117
 - changeExtension, 118
 - hasExtention, 118
 - isAbsolute, 118
 - parent, 119
 - Path, 118
 - printToStream, 119
- kfoundation::PoolObject, 119
 - finalize, 120
- kfoundation::PredictiveParserBase, 122
 - isSpace, 124
 - isValidIdentifierBeginChar, 125
 - isValidIdentifierChar, 125
 - PredictiveParserBase, 124
 - readAllAlphabet, 125
 - readAllAlphanumeric, 125
 - readAllBeforeChar, 125
 - readAllBeforeCharSkipEscaped, 126
 - readAllBeforeNewLine, 126
 - readAllBeforeSequence, 126
 - readAllBeforeSpace, 126
 - readAllBeforeSpaceOrNewLine, 127
 - readAllNumeric, 127
 - readAlphabet, 127
 - readAlphanumeric, 127
 - readAny, 128
 - readChar, 128
 - readIdentifier, 128
 - readIdentifierBeginChar, 130
 - readIdentifierChar, 130
 - readNewLine, 130
 - readNumber, 130, 131
 - readNumeric, 131
 - readSequence, 132
 - readSpace, 132
 - skipSpaces, 132
 - skipSpacesAndNewLines, 132
 - testChar, 132, 133
 - testSequence, 133
 - testSpace, 133
- kfoundation::ProximityIterator, 133
 - next, 135
 - ProximityIterator, 134
- kfoundation::Ptr
 - ~Ptr, 138
 - cast, 138
 - del, 138
 - getRetainCount, 138
 - isValid, 138
 - isa, 138
 - operator*, 139
 - operator->, 139
 - operator=, 139
 - Ptr, 137, 138
 - release, 139
 - replace, 139, 140
 - retain, 140
 - trace, 140
 - untrace, 140
- kfoundation::Ptr< T >, 135
- kfoundation::Range, 141
 - border, 143
 - isEmpty, 143
 - Range, 143
- kfoundation::Rangelterator, 143
- kfoundation::RefCountMemoryManager, 144
 - getTable, 145
 - getTableSize, 146
 - release, 146
 - remove, 146
- kfoundation::SPtr
 - SPtr, 150, 151
- kfoundation::SPtr< T >, 149
- kfoundation::SerializingStreamer, 146
 - printToJsonStream, 148
 - printToStream, 149
 - printToXmlStream, 149
- kfoundation::StandardInputStreamAdapter, 151
 - isMarkSupported, 152
 - mark, 152
 - peek, 152
 - read, 152, 153
 - reset, 153
 - skip, 153
- kfoundation::StandardOutputStreamAdapter, 154
 - close, 154
 - write, 154, 155
- kfoundation::StreamDeserializer, 156
 - readFromXmlStream, 156
- kfoundation::Streamer, 156
- kfoundation::StringInputStream, 158
 - isMarkSupported, 159
 - mark, 159
 - peek, 159
 - read, 160

- reset, [160](#)
- skip, [160](#)
- StringInputStream, [159](#)
- kfoundation::System, [161](#)
 - exec, [162](#)
 - getCurrentTimeInMilliseconds, [162](#)
 - getExePath, [162](#)
 - operating_system_t, [162](#)
 - sleep, [162](#)
- kfoundation::TextToken, [163](#)
- kfoundation::Thread, [164](#)
 - isRunning, [165](#)
 - isTheCurrentThread, [165](#)
 - run, [165](#)
 - start, [166](#)
 - Thread, [165](#)
- kfoundation::Timer, [166](#)
 - getCpuTime, [167](#)
- kfoundation::Token, [167](#)
 - asAttribute, [169](#)
 - asCollection, [170](#)
 - asEndCollection, [170](#)
 - asEndObject, [170](#)
 - asObject, [170](#)
 - codeRange, [171](#)
 - Token, [169](#)
- kfoundation::Tuple, [171](#)
 - negate, [173](#)
 - operator*, [173](#)
 - operator+, [173](#), [174](#)
 - operator-, [174](#)
 - operator/, [174](#), [175](#)
 - operator%, [173](#)
- kfoundation::Tuple1D, [175](#)
 - Tuple1D, [176](#)
- kfoundation::Tuple2D, [176](#)
 - set, [179](#)
 - Tuple2D, [178](#)
- kfoundation::Tuple3D, [179](#)
 - set, [180](#)
 - Tuple3D, [180](#)
- kfoundation::UniChar, [181](#)
 - get, [183](#)
 - readUtf8, [183](#), [184](#)
 - set, [184](#)
 - toLowerCase, [184](#)
 - toUpperCase, [184](#)
 - UniChar, [182](#), [183](#)
 - writeUtf8, [184](#), [186](#)
- kfoundation::XmlObjectStreamReader, [186](#)
 - XmlObjectStreamReader, [187](#)
- L1
 - kfoundation::Logger, [84](#)
- L2
 - kfoundation::Logger, [84](#)
- L3
 - kfoundation::Logger, [84](#)
- LINUX
 - kfoundation::System, [162](#)
- level_t
 - kfoundation::Logger, [84](#)
- listen
 - kfoundation::InternetInputStream, [71](#)
- lock
 - kfoundation::FileOutputStream, [59](#)
- log
 - kfoundation::Logger, [84](#)
- LongInt
 - kfoundation::LongInt, [86](#)
- MACH
 - kfoundation::System, [162](#)
- Macros and Typedefs, [15](#)
 - EQUAL, [16](#)
 - GREATER, [16](#)
 - IS_NULL, [16](#)
 - KF_EXPORT, [16](#)
 - KF_NOP, [16](#)
 - kf_comparison_t, [16](#)
 - NOT_NULL, [16](#)
 - SMALLER, [16](#)
- ManagedArray
 - kfoundation::ManagedArray, [89](#)
- mark
 - kfoundation::BufferInputStream, [34](#)
 - kfoundation::FileInputStream, [55](#)
 - kfoundation::InputStream, [63](#)
 - kfoundation::InternetInputStream, [71](#)
 - kfoundation::StandardInputStreamAdapter, [152](#)
 - kfoundation::StringInputStream, [159](#)
- member
 - kfoundation::ObjectSerializer, [109](#)
- Memory Management, [18](#)
 - AS, [18](#)
 - ISA, [18](#)
- mul
 - kfoundation::NumericVector, [100](#)
- Mutex
 - kfoundation::Mutex, [97](#)
- Mutithreading, [20](#)
 - KF_SYNCHRONIZED, [20](#)
- NEUTRAL
 - kfoundation::Direction, [48](#)
- NOT_NULL
 - Macros and Typedefs, [16](#)
- negate
 - kfoundation::Tuple, [173](#)
- next
 - kfoundation::ProximityIterator, [135](#)
- null
 - kfoundation::ObjectSerializer, [109](#)
- NumericVector
 - kfoundation::NumericVector, [100](#)
- object
 - kfoundation::ObjectSerializer, [109](#)

- Object Serialization and Deserialization, 23
- ObjectDumpBuilderException
 - kfoundation::ObjectDumpBuilderException, 102
- ObjectPoolMemoryManager
 - kfoundation::ObjectPoolMemoryManager, 104
- ObjectSerializer
 - kfoundation::ObjectSerializer, 108
- operating_system_t
 - kfoundation::System, 162
- operator*
 - kfoundation::Ptr, 139
 - kfoundation::Tuple, 173
- operator+
 - kfoundation::Tuple, 173, 174
- operator-
 - kfoundation::Tuple, 174
- operator->
 - kfoundation::Ptr, 139
- operator/
 - kfoundation::Tuple, 174, 175
- operator=
 - kfoundation::Ptr, 139
- operator%
 - kfoundation::Tuple, 173
- OutOfMemoryException
 - kfoundation::OutOfMemoryException, 113
- output_type_t
 - kfoundation::ObjectSerializer, 108
- PPtr
 - kfoundation::PPtr, 121
- parent
 - kfoundation::Path, 119
- parse
 - kfoundation::Double, 51
 - kfoundation::Int, 66
 - kfoundation::LongInt, 87
- ParseException
 - kfoundation::ParseException, 116
- parseInt
 - kfoundation::NumericVector, 100
- Path
 - kfoundation::Path, 118
- peek
 - kfoundation::BufferInputStream, 34
 - kfoundation::FileInputStream, 56
 - kfoundation::InputStream, 63
 - kfoundation::InternetInputStream, 72
 - kfoundation::StandardInputStreamAdapter, 152
 - kfoundation::StringInputStream, 159
- pop
 - kfoundation::Array, 28
 - kfoundation::ManagedArray, 90
- PredictiveParserBase
 - kfoundation::PredictiveParserBase, 124
- printToJsonStream
 - kfoundation::SerializingStreamer, 148
- printToStream
 - kfoundation::Path, 119
- kfoundation::SerializingStreamer, 149
- printToXmlStream
 - kfoundation::SerializingStreamer, 149
- ProximityIterator
 - kfoundation::ProximityIterator, 134
- Ptr
 - kfoundation::Ptr, 137, 138
- push
 - kfoundation::Array, 28
 - kfoundation::ManagedArray, 91
- Range
 - kfoundation::Range, 143
- Range Arithmetics, 24
- read
 - kfoundation::BufferInputStream, 34
 - kfoundation::FileInputStream, 56
 - kfoundation::InputStream, 63, 64
 - kfoundation::InternetInputStream, 72
 - kfoundation::StandardInputStreamAdapter, 152, 153
 - kfoundation::StringInputStream, 160
- readAllAlphabet
 - kfoundation::PredictiveParserBase, 125
- readAllAlphanumeric
 - kfoundation::PredictiveParserBase, 125
- readAllBeforeChar
 - kfoundation::PredictiveParserBase, 125
- readAllBeforeCharSkipEscaped
 - kfoundation::PredictiveParserBase, 126
- readAllBeforeNewLine
 - kfoundation::PredictiveParserBase, 126
- readAllBeforeSequence
 - kfoundation::PredictiveParserBase, 126
- readAllBeforeSpace
 - kfoundation::PredictiveParserBase, 126
- readAllBeforeSpaceOrNewLine
 - kfoundation::PredictiveParserBase, 127
- readAllNumeric
 - kfoundation::PredictiveParserBase, 127
- readAlphabet
 - kfoundation::PredictiveParserBase, 127
- readAlphanumeric
 - kfoundation::PredictiveParserBase, 127
- readAny
 - kfoundation::PredictiveParserBase, 128
- readChar
 - kfoundation::PredictiveParserBase, 128
- readFromXmlStream
 - kfoundation::StreamDeserializer, 156
- readIdentifier
 - kfoundation::PredictiveParserBase, 128
- readIdentifierBeginChar
 - kfoundation::PredictiveParserBase, 130
- readIdentifierChar
 - kfoundation::PredictiveParserBase, 130
- readNewLine
 - kfoundation::PredictiveParserBase, 130
- readNumber

- kfoundation::PredictiveParserBase, 130, 131
- readNumeric
 - kfoundation::PredictiveParserBase, 131
- readSequence
 - kfoundation::PredictiveParserBase, 132
- readSpace
 - kfoundation::PredictiveParserBase, 132
- readUtf8
 - kfoundation::UniChar, 183, 184
- registerManager
 - kfoundation::MasterMemoryManager, 93
- release
 - kfoundation::Condition, 46
 - kfoundation::MemoryManager, 96
 - kfoundation::ObjectPoolMemoryManager, 105
 - kfoundation::Ptr, 139
 - kfoundation::RefCountMemoryManager, 146
- remove
 - kfoundation::Array, 28
 - kfoundation::ManagedArray, 91
 - kfoundation::MemoryManager, 96
 - kfoundation::ObjectPoolMemoryManager, 105
 - kfoundation::RefCountMemoryManager, 146
- replace
 - kfoundation::Ptr, 139, 140
- reset
 - kfoundation::BufferInputStream, 35
 - kfoundation::FileInputStream, 56
 - kfoundation::InputStream, 64
 - kfoundation::InternetInputStream, 72
 - kfoundation::StandardInputStreamAdapter, 153
 - kfoundation::StringInputStream, 160
- retain
 - kfoundation::Ptr, 140
- run
 - kfoundation::Thread, 165
- SMALLER
 - Macros and Typedefs, 16
- SOLARIS
 - kfoundation::System, 162
- SPtr
 - kfoundation::SPtr, 150, 151
- serialize
 - kfoundation::KFException, 81
- set
 - kfoundation::Bool, 31
 - kfoundation::CodeLocation, 42
 - kfoundation::Direction, 48
 - kfoundation::Double, 51
 - kfoundation::Int, 66
 - kfoundation::LongInt, 87
 - kfoundation::Tuple2D, 179
 - kfoundation::Tuple3D, 180
 - kfoundation::UniChar, 184
- setByteIndex
 - kfoundation::CodeLocation, 42
- setCharIndex
 - kfoundation::CodeLocation, 42
- setCol
 - kfoundation::CodeLocation, 43
- setFormat
 - kfoundation::Logger::Channel, 40
- setLevel
 - kfoundation::Logger, 85
 - kfoundation::Logger::Channel, 40
- setLine
 - kfoundation::CodeLocation, 43
- setName
 - kfoundation::KFException, 81
- setSilent
 - kfoundation::Logger::Channel, 40
- setSize
 - kfoundation::ManagedArray, 91
- skip
 - kfoundation::BufferInputStream, 35
 - kfoundation::FileInputStream, 56
 - kfoundation::InputStream, 64
 - kfoundation::InternetInputStream, 72
 - kfoundation::StandardInputStreamAdapter, 153
 - kfoundation::StringInputStream, 160
- skipSpaces
 - kfoundation::PredictiveParserBase, 132
- skipSpacesAndNewLines
 - kfoundation::PredictiveParserBase, 132
- sleep
 - kfoundation::System, 162
- start
 - kfoundation::Thread, 166
- Stream
 - kfoundation::Logger::Stream, 155
- StringInputStream
 - kfoundation::StringInputStream, 159
- sub
 - kfoundation::NumericVector, 100
- testChar
 - kfoundation::PredictiveParserBase, 132, 133
- testSequence
 - kfoundation::PredictiveParserBase, 133
- testSpace
 - kfoundation::PredictiveParserBase, 133
- Thread
 - kfoundation::Thread, 165
- toHexString
 - kfoundation::Int, 66
- toLowerCase
 - kfoundation::UniChar, 184
- toString
 - kfoundation::Bool, 31
- toUpperCase
 - kfoundation::UniChar, 184
- Token
 - kfoundation::Token, 169
- trace
 - kfoundation::Ptr, 140
- truncate
 - kfoundation::FileOutputStream, 59

- Tuple1D
 - kfoundation::Tuple1D, [176](#)
- Tuple2D
 - kfoundation::Tuple2D, [178](#)
- Tuple3D
 - kfoundation::Tuple3D, [180](#)
- UNSUPPORTED
 - kfoundation::System, [162](#)
- unbind
 - kfoundation::InternetInputStream, [74](#)
- UniChar
 - kfoundation::UniChar, [182](#), [183](#)
- unlock
 - kfoundation::FileOutputStream, [59](#)
- unregisterManager
 - kfoundation::MasterMemoryManager, [94](#)
- untrace
 - kfoundation::Ptr, [140](#)
- updateTable
 - kfoundation::MasterMemoryManager, [94](#)
- Utilities, [22](#)
- validateClass
 - kfoundation::ObjectToken, [112](#)
- WRN
 - kfoundation::Logger, [84](#)
- Wrappers and Containers, [21](#)
- write
 - kfoundation::BufferOutputStream, [37](#)
 - kfoundation::FileOutputStream, [59](#), [60](#)
 - kfoundation::InternetOutputStream, [76](#)
 - kfoundation::OutputStream, [114](#)
 - kfoundation::StandardOutputStreamAdapter, [154](#), [155](#)
- writeUtf8
 - kfoundation::UniChar, [184](#), [186](#)
- XML
 - kfoundation::ObjectSerializer, [108](#)
- XmlObjectStreamReader
 - kfoundation::XmlObjectStreamReader, [187](#)