

BIOMETRIC EVALUATION COMMON FRAMEWORK

PROGRAMMER'S GUIDE

VERSION 0.1

WAYNE SALAMON
GREGORY FIUMARA

IMAGE GROUP
INFORMATION ACCESS DIVISION
INFORMATION TECHNOLOGY LABORATORY



OCTOBER 31, 2018

Contents

1	Introduction	1
2	Overview	3
3	Framework	5
3.1	Versioning	5
3.2	Enumerations	5
4	Memory	9
4.1	AutoBuffer	9
4.2	AutoArray	10
4.3	IndexedBuffer	11
5	Error Handling	13
5.1	Biometric Evaluation Exceptions	13
5.2	Signal Handling	13
6	Input/Output	17
6.1	Utility	17
6.2	Record Management	17
6.3	Logging	18
6.3.1	FileLogsheet	19
6.3.2	SysLogsheet	20
6.4	Properties	20
6.5	Compressor	21
7	Text	23
8	Time and Timing	25
8.1	Elapsed Time	25
8.2	Limiting Execution Time	25
9	Process Information and Control	27
9.1	Process Statistics	27
9.2	Process Management	29
9.2.1	Manager	29
9.2.2	Worker	29
9.2.3	WorkerController	30
9.2.4	Communications	32
10	System	35

11 Image	37
11.1 The Image Namespace	37
11.2 The Image Class	37
11.3 Raw Image	38
11.4 JPEG	38
11.5 JPEGL	38
11.6 JPEG2000	38
11.7 NetPBM	39
11.8 PNG	39
11.9 TIFF	39
11.10 WSQ	39
11.11 BMP	39
12 Video	41
12.1 Container	41
12.2 Stream	41
13 Device	45
13.1 TLV	45
13.2 Smartcard	45
13.2.1 APDU	45
13.2.2 Smartcard Communication	45
14 Feature	47
14.1 ANSI/NIST Features	47
14.1.1 ANSI/NIST 2011 Extended Feature Sets	47
14.2 ISO/INCITS Features	49
15 View	51
15.1 ANSI/NIST Views	52
16 Finger	53
16.1 ANSI/NIST Minutiae Data Record	53
16.1.1 ANSI/NIST Finger Views	53
16.1.2 ISO/INCITS Finger Views	55
17 Palm	57
17.1 ANSI/NIST Palm Views	57
18 Face	59
18.0.1 ISO/INCITS Face Views	59
19 Data Interchange	63
19.1 ANSI/NIST Data Records	63
19.2 INCITS Data Records	65
20 Messaging	69
20.1 Message Center	69
20.2 Command Center	70

21 Parallel Processing	73
21.1 MPI Parallel Processing Package	73
21.2 Work Package	73
21.3 MPI Resources	75
21.4 Checkpoint Save and Restore	75
21.5 Distributor	76
21.5.1 Record Store Distributor	76
21.5.2 CSV Distributor	76
21.6 Receiver	77
21.7 Work Package Processor	77
21.7.1 Record Processor	78
21.8 MPI Runtime	78
21.9 Logging	79
21.10 MPI Framework Applications	79
References	85
A Building the Framework	87
A.1 Language Features	87
A.2 The Framework Build System	87
A.3 The CMake Build System	87
A.4 External Software Dependencies	88
A.4.1 NIST Biometric Image Software	88
A.4.2 Video and Image Processing	88
A.4.3 Cryptography	88
A.4.4 Sqlite	89
A.4.5 Berkeley Database	89
A.4.6 Message Passing Interface	89
B Running an MPI Job	91
B.1 OpenMPI	91
B.2 Example Shell Script	91
C Namespace Index	93
C.1 Namespace List	93
D Hierarchical Index	95
D.1 Class Hierarchy	95
E Class Index	99
E.1 Class List	99
F Namespace Documentation	107
F.1 BiometricEvaluation Namespace Reference	107
F.1.1 Detailed Description	108
F.2 BiometricEvaluation::Error Namespace Reference	108
F.2.1 Detailed Description	109
F.2.2 Function Documentation	109
F.2.2.1 errorStr()	109
F.3 BiometricEvaluation::Face Namespace Reference	109
F.3.1 Detailed Description	111
F.3.2 Typedef Documentation	111
F.3.2.1 PropertySet	111

F.4	BiometricEvaluation::Feature Namespace Reference	111
F.4.1	Detailed Description	112
F.4.2	Function Documentation	113
F.4.2.1	operator<<()	113
F.5	BiometricEvaluation::Feature::Sort Namespace Reference	113
F.5.1	Detailed Description	114
F.5.2	Enumeration Type Documentation	114
F.5.2.1	Kind	114
F.5.3	Function Documentation	114
F.5.3.1	sort()	114
F.5.3.2	stableSort()	115
F.6	BiometricEvaluation::Finger Namespace Reference	115
F.6.1	Detailed Description	116
F.6.2	Enumeration Type Documentation	117
F.6.2.1	FingerImageCode	117
F.6.2.2	Impression	117
F.6.2.3	PatternClassification	117
F.6.2.4	Position	117
F.7	BiometricEvaluation::Framework Namespace Reference	117
F.7.1	Detailed Description	118
F.7.2	Enumeration Type Documentation	118
F.7.2.1	APICurrentState	118
F.7.3	Function Documentation	118
F.7.3.1	getCompileDate()	118
F.7.3.2	getCompiler()	119
F.7.3.3	getCompilerVersion()	119
F.7.3.4	getCompileTime()	119
F.7.3.5	getMajorVersion()	119
F.7.3.6	getMinorVersion()	119
F.7.3.7	operator<<()	120
F.7.3.8	to_string()	120
F.8	BiometricEvaluation::Image Namespace Reference	120
F.8.1	Detailed Description	122
F.8.2	Enumeration Type Documentation	122
F.8.2.1	CompressionAlgorithm	122
F.8.2.2	PixelFormat	122
F.8.3	Function Documentation	123
F.8.3.1	distance()	123
F.8.3.2	operator<<()	123
F.8.3.3	removeComponents()	123
F.8.3.4	to_string() [1/5]	124
F.8.3.5	to_string() [2/5]	124
F.8.3.6	to_string() [3/5]	125
F.8.3.7	to_string() [4/5]	125
F.8.3.8	to_string() [5/5]	125
F.8.4	Variable Documentation	126
F.8.4.1	CentimetersPerInch	126
F.8.4.2	MillimetersPerInch	126
F.9	BiometricEvaluation::IO Namespace Reference	126
F.9.1	Detailed Description	127
F.9.2	Enumeration Type Documentation	127

F.9.2.1	Mode	127
F.10	BiometricEvaluation::IO::Utility Namespace Reference	127
F.10.1	Detailed Description	128
F.10.2	Function Documentation	129
F.10.2.1	copyDirectoryContents()	129
F.10.2.2	countLines() [1/2]	129
F.10.2.3	countLines() [2/2]	129
F.10.2.4	createTemporaryFile() [1/2]	130
F.10.2.5	createTemporaryFile() [2/2]	130
F.10.2.6	fileExists()	131
F.10.2.7	getFileSize()	131
F.10.2.8	isReadable()	132
F.10.2.9	isWritable()	132
F.10.2.10	makePath()	133
F.10.2.11	readFile()	133
F.10.2.12	readPipe() [1/2]	134
F.10.2.13	readPipe() [2/2]	134
F.10.2.14	removeDirectory() [1/2]	135
F.10.2.15	removeDirectory() [2/2]	135
F.10.2.16	setAsideName()	135
F.10.2.17	sumDirectoryUsage()	136
F.10.2.18	writeFile() [1/2]	136
F.10.2.19	writeFile() [2/2]	137
F.10.2.20	writePipe() [1/2]	137
F.10.2.21	writePipe() [2/2]	138
F.11	BiometricEvaluation::Iris Namespace Reference	138
F.11.1	Detailed Description	139
F.12	BiometricEvaluation::Memory Namespace Reference	139
F.12.1	Detailed Description	140
F.12.2	Function Documentation	140
F.12.2.1	isLittleEndian()	140
F.12.2.2	make_unique() [1/3]	141
F.12.2.3	make_unique() [2/3]	141
F.12.2.4	make_unique() [3/3]	141
F.12.2.5	operator"!=(141
F.12.2.6	operator<()	142
F.12.2.7	operator<=()	142
F.12.2.8	operator==(142
F.12.2.9	operator>()	142
F.12.2.10	operator>=()	142
F.13	BiometricEvaluation::Memory::AutoArrayUtility Namespace Reference	143
F.13.1	Detailed Description	143
F.13.2	Function Documentation	143
F.13.2.1	cstr()	143
F.13.2.2	getString()	144
F.13.2.3	setString() [1/2]	144
F.13.2.4	setString() [2/2]	144
F.14	BiometricEvaluation::MPI Namespace Reference	145
F.14.1	Detailed Description	146
F.14.2	Typedef Documentation	146
F.14.2.1	msgtag_t	146

F.14.2.2	taskcmd_t	146
F.14.2.3	taskstat_t	147
F.14.3	Enumeration Type Documentation	147
F.14.3.1	MessageTag	147
F.14.3.2	TaskCommand	147
F.14.3.3	TaskStatus	147
F.14.4	Function Documentation	148
F.14.4.1	generateUniqueID()	148
F.14.4.2	logEntry()	148
F.14.4.3	logMessage()	148
F.14.4.4	openLogsheet()	148
F.14.4.5	printStatus()	149
F.15	BiometricEvaluation::Palm Namespace Reference	149
F.15.1	Detailed Description	150
F.15.2	Enumeration Type Documentation	150
F.15.2.1	Position	150
F.16	BiometricEvaluation::Plantar Namespace Reference	150
F.16.1	Detailed Description	150
F.16.2	Enumeration Type Documentation	150
F.16.2.1	Position	150
F.17	BiometricEvaluation::Process Namespace Reference	150
F.17.1	Detailed Description	151
F.17.2	Typedef Documentation	151
F.17.2.1	ParameterList	151
F.18	BiometricEvaluation::System Namespace Reference	152
F.18.1	Detailed Description	152
F.18.2	Function Documentation	152
F.18.2.1	getCPUCoreCount()	152
F.18.2.2	getCPUCount()	152
F.18.2.3	getCPUSocketCount()	153
F.18.2.4	getLoadAverage()	153
F.18.2.5	getRealMemorySize()	153
F.19	BiometricEvaluation::Text Namespace Reference	154
F.19.1	Detailed Description	155
F.19.2	Function Documentation	155
F.19.2.1	basename()	155
F.19.2.2	caseInsensitiveCompare()	155
F.19.2.3	decodeBase64()	155
F.19.2.4	digest() [1/2]	156
F.19.2.5	digest() [2/2]	156
F.19.2.6	dirname()	157
F.19.2.7	encodeBase64()	157
F.19.2.8	ltrim()	157
F.19.2.9	ltrimWhitespace()	158
F.19.2.10	rtrim()	158
F.19.2.11	rtrimWhitespace()	158
F.19.2.12	split()	159
F.19.2.13	toLowerCase()	159
F.19.2.14	toUpperCase()	160
F.19.2.15	trim()	160
F.19.2.16	trimWhitespace()	160

F.20	BiometricEvaluation::Time Namespace Reference	161
F.20.1	Detailed Description	161
F.20.2	Function Documentation	161
F.20.2.1	getCurrentCalendarInformation()	162
F.20.2.2	getCurrentDate()	162
F.20.2.3	getCurrentDateAndTime()	162
F.20.2.4	getCurrentTime()	162
F.20.2.5	operator<<()	162
F.20.2.6	put_time()	163
F.21	BiometricEvaluation::Video Namespace Reference	163
F.21.1	Detailed Description	163
F.21.2	Enumeration Type Documentation	164
F.21.2.1	CodingFormat	164
F.21.2.2	ContainerFormat	164
F.22	BiometricEvaluation::View Namespace Reference	164
F.22.1	Detailed Description	164
F.22.2	Function Documentation	165
F.22.2.1	operator<<() [1/2]	165
F.22.2.2	operator<<() [2/2]	165
G	Class Documentation	167
G.1	BiometricEvaluation::Feature::AN2K7Minutiae Class Reference	167
G.1.1	Detailed Description	168
G.1.2	Member Enumeration Documentation	168
G.1.2.1	EncodingMethod	168
G.1.3	Constructor & Destructor Documentation	169
G.1.3.1	AN2K7Minutiae() [1/2]	169
G.1.3.2	AN2K7Minutiae() [2/2]	169
G.1.4	Member Function Documentation	169
G.1.4.1	convertCoordinate()	170
G.1.4.2	convertEncodingMethod()	170
G.1.4.3	convertPatternClassification() [1/2]	170
G.1.4.4	convertPatternClassification() [2/2]	171
G.1.4.5	getOriginatingFingerprintReadingSystem()	171
G.1.4.6	getPatternClassificationSet()	171
G.2	BiometricEvaluation::Finger::AN2KMinutiaeDataRecord Class Reference	171
G.2.1	Detailed Description	172
G.2.2	Constructor & Destructor Documentation	172
G.2.2.1	AN2KMinutiaeDataRecord() [1/2]	172
G.2.2.2	AN2KMinutiaeDataRecord() [2/2]	173
G.2.3	Member Function Documentation	173
G.2.3.1	getAN2K11EFS()	173
G.2.3.2	getAN2K7Minutiae()	173
G.2.3.3	getImpressionType()	174
G.2.3.4	getRegisteredVendorBlock()	174
G.3	BiometricEvaluation::View::AN2KViewVariableResolution::AN2KQualityMetric Struct Reference	174
G.3.1	Detailed Description	174
G.4	BiometricEvaluation::DataInterchange::AN2KRecord Class Reference	175
G.4.1	Detailed Description	176
G.4.2	Member Typedef Documentation	176
G.4.2.1	CharacterSet	176

G.4.2.2	DomainName	176
G.4.3	Constructor & Destructor Documentation	176
G.4.3.1	AN2KRecord() [1/2]	176
G.4.3.2	AN2KRecord() [2/2]	177
G.4.4	Member Function Documentation	177
G.4.4.1	getDate()	177
G.4.4.2	getDestinationAgency()	177
G.4.4.3	getDirectoryOfCharacterSets()	178
G.4.4.4	getDomainName()	178
G.4.4.5	getFingerCaptureCount()	178
G.4.4.6	getFingerCaptures()	178
G.4.4.7	getFingerLatentCount()	178
G.4.4.8	getFingerLatents()	179
G.4.4.9	getGreenwichMeanTime()	179
G.4.4.10	getMinutiaeDataRecordSet()	179
G.4.4.11	getNativeScanningResolution()	179
G.4.4.12	getNominalTransmittingResolution()	179
G.4.4.13	getOriginatingAgency()	180
G.4.4.14	getPriority()	180
G.4.4.15	getTransactionControlNumber()	180
G.4.4.16	getVersionNumber()	180
G.4.4.17	recordLocations() [1/2]	180
G.4.4.18	recordLocations() [2/2]	181
G.5	BiometricEvaluation::Palm::AN2KView Class Reference	181
G.5.1	Detailed Description	182
G.5.2	Constructor & Destructor Documentation	182
G.5.2.1	AN2KView() [1/2]	182
G.5.2.2	AN2KView() [2/2]	182
G.5.3	Member Function Documentation	182
G.5.3.1	getPalmQualityMetric()	182
G.5.3.2	getPosition()	183
G.6	BiometricEvaluation::View::AN2KView Class Reference	183
G.6.1	Detailed Description	184
G.6.2	Member Enumeration Documentation	184
G.6.2.1	DeviceMonitoringMode	184
G.6.2.2	RecordType	185
G.6.3	Constructor & Destructor Documentation	185
G.6.3.1	AN2KView() [1/2]	185
G.6.3.2	AN2KView() [2/2]	185
G.6.4	Member Function Documentation	185
G.6.4.1	convertCompressionAlgorithm()	185
G.6.4.2	convertDeviceMonitoringMode()	186
G.6.4.3	getAN2KRecord()	186
G.6.4.4	getMinutiaeDataRecordSet()	186
G.6.4.5	getRecordType()	186
G.7	BiometricEvaluation::Latent::AN2KView Class Reference	187
G.7.1	Constructor & Destructor Documentation	187
G.7.1.1	AN2KView() [1/2]	187
G.7.1.2	AN2KView() [2/2]	188
G.7.2	Member Function Documentation	188
G.7.2.1	getLatentQualityMetric()	188

G.7.2.2	getPositions()	188
G.7.2.3	getPrintPositionCoordinates()	188
G.8	BiometricEvaluation::Finger::AN2KView Class Reference	188
G.8.1	Detailed Description	190
G.8.2	Constructor & Destructor Documentation	190
G.8.2.1	AN2KView() [1/2]	190
G.8.2.2	AN2KView() [2/2]	190
G.8.3	Member Function Documentation	191
G.8.3.1	addMinutiaeDataRecord()	191
G.8.3.2	convertFingerImageCode()	191
G.8.3.3	convertPosition()	192
G.8.3.4	getImpressionType()	192
G.8.3.5	getMinutiaeDataRecordSet()	192
G.8.3.6	getPositions()	192
G.8.3.7	populateFGP()	193
G.8.3.8	setImpressionType()	193
G.8.3.9	setPositions()	193
G.9	BiometricEvaluation::Finger::AN2KViewCapture Class Reference	193
G.9.1	Detailed Description	195
G.9.2	Member Enumeration Documentation	195
G.9.2.1	AmputatedBandaged	195
G.9.3	Constructor & Destructor Documentation	195
G.9.3.1	AN2KViewCapture() [1/2]	195
G.9.3.2	AN2KViewCapture() [2/2]	196
G.9.4	Member Function Documentation	196
G.9.4.1	extractNISTQuality()	196
G.9.4.2	getAlternateFingerSegmentPositionSet()	196
G.9.4.3	getAmputatedBandaged()	196
G.9.4.4	getFingerprintQualityMetric()	197
G.9.4.5	getFingerSegmentPositionSet()	197
G.9.4.6	getNISTQualityMetric()	197
G.9.4.7	getPosition()	197
G.9.4.8	getPrintPositionCoordinates()	197
G.9.4.9	getSegmentationQualityMetric()	198
G.10	BiometricEvaluation::Finger::AN2KViewFixedResolution Class Reference	198
G.10.1	Detailed Description	198
G.10.2	Constructor & Destructor Documentation	198
G.10.2.1	AN2KViewFixedResolution() [1/2]	199
G.10.2.2	AN2KViewFixedResolution() [2/2]	199
G.11	BiometricEvaluation::View::AN2KViewVariableResolution Class Reference	200
G.11.1	Detailed Description	201
G.11.2	Constructor & Destructor Documentation	201
G.11.2.1	AN2KViewVariableResolution() [1/2]	201
G.11.2.2	AN2KViewVariableResolution() [2/2]	201
G.11.3	Member Function Documentation	201
G.11.3.1	extractQuality()	202
G.11.3.2	getCaptureDate()	202
G.11.3.3	getComment()	202
G.11.3.4	getImpressionType()	202
G.11.3.5	getPositionDescriptors()	203
G.11.3.6	getPositions()	203

G.11.3.7	getPrintPositionCoordinates()	203
G.11.3.8	getQualityMetric()	203
G.11.3.9	getSourceAgency()	203
G.11.3.10	getUserDefinedField()	204
G.11.3.11	parseUserDefinedField()	204
G.12	BiometricEvaluation::Feature::Sort::Angle Class Reference	205
G.12.1	Detailed Description	205
G.12.2	Member Function Documentation	205
G.12.2.1	operator()	205
G.13	BiometricEvaluation::DataInterchange::ANSI2004Record Class Reference	205
G.13.1	Detailed Description	206
G.13.2	Constructor & Destructor Documentation	206
G.13.2.1	ANSI2004Record() [1/3]	206
G.13.2.2	ANSI2004Record() [2/3]	207
G.13.2.3	ANSI2004Record() [3/3]	207
G.13.3	Member Function Documentation	207
G.13.3.1	getEDBLength()	207
G.13.3.2	getFMR()	207
G.13.3.3	getFMRLength()	208
G.13.3.4	getMinutia() [1/2]	208
G.13.3.5	getMinutia() [2/2]	208
G.13.3.6	getNumFingerViews()	208
G.13.3.7	getView()	209
G.13.3.8	insertView() [1/2]	209
G.13.3.9	insertView() [2/2]	209
G.13.3.10	isolateView()	210
G.13.3.11	removeView()	210
G.13.3.12	setMinutia() [1/2]	211
G.13.3.13	setMinutia() [2/2]	211
G.13.3.14	updateView()	211
G.14	BiometricEvaluation::Finger::ANSI2004View Class Reference	212
G.14.1	Detailed Description	213
G.14.2	Constructor & Destructor Documentation	213
G.14.2.1	ANSI2004View() [1/2]	213
G.14.2.2	ANSI2004View() [2/2]	213
G.14.3	Member Function Documentation	214
G.14.3.1	readCoreDeltaData()	214
G.15	BiometricEvaluation::Finger::ANSI2007View Class Reference	214
G.15.1	Detailed Description	215
G.15.2	Constructor & Destructor Documentation	215
G.15.2.1	ANSI2007View() [1/2]	215
G.15.2.2	ANSI2007View() [2/2]	215
G.15.3	Member Function Documentation	216
G.15.3.1	readCoreDeltaData()	216
G.16	BiometricEvaluation::Device::Smartcard::APDU Class Reference	216
G.16.1	Member Data Documentation	218
G.16.1.1	cla	218
G.16.1.2	FIELD_LC	218
G.16.1.3	FIELD_LE	218
G.16.1.4	field_mask	218
G.16.1.5	ins	218

G.16.1.6	lc	218
G.16.1.7	le	218
G.16.1.8	nc	218
G.16.1.9	pl	218
G.16.1.10	p2	218
G.17	BiometricEvaluation::Device::Smartcard::APDUException Struct Reference	219
G.17.1	Detailed Description	219
G.17.2	Constructor & Destructor Documentation	219
G.17.2.1	APDUException() [1/2]	219
G.17.2.2	APDUException() [2/2]	219
G.17.3	Member Data Documentation	219
G.17.3.1	apdu	219
G.17.3.2	response	220
G.18	BiometricEvaluation::Device::Smartcard::APDUResponse Struct Reference	220
G.18.1	Detailed Description	220
G.18.2	Constructor & Destructor Documentation	220
G.18.2.1	APDUResponse() [1/2]	220
G.18.2.2	APDUResponse() [2/2]	220
G.18.3	Member Data Documentation	221
G.18.3.1	data	221
G.18.3.2	sw1	221
G.18.3.3	sw2	221
G.19	BiometricEvaluation::Framework::API< T > Class Template Reference	221
G.19.1	Detailed Description	221
G.19.2	Constructor & Destructor Documentation	222
G.19.2.1	API()	222
G.19.3	Member Function Documentation	222
G.19.3.1	call()	222
G.19.3.2	getSignalManager()	223
G.19.3.3	getTimer()	223
G.19.3.4	getWatchdog()	223
G.20	BiometricEvaluation::IO::ArchiveRecordStore Class Reference	223
G.20.1	Detailed Description	225
G.20.2	Constructor & Destructor Documentation	225
G.20.2.1	ArchiveRecordStore() [1/2]	225
G.20.2.2	ArchiveRecordStore() [2/2]	225
G.20.2.3	~ArchiveRecordStore()	226
G.20.3	Member Function Documentation	226
G.20.3.1	changeDescription()	226
G.20.3.2	flush()	226
G.20.3.3	getArchiveName()	227
G.20.3.4	getCount()	227
G.20.3.5	getDescription()	227
G.20.3.6	getManifestName()	227
G.20.3.7	getPathname()	227
G.20.3.8	getSpaceUsed()	228
G.20.3.9	insert()	228
G.20.3.10	length()	228
G.20.3.11	move()	229
G.20.3.12	needsVacuum() [1/2]	229
G.20.3.13	needsVacuum() [2/2]	229

G.20.3.14 read()	230
G.20.3.15 remove()	230
G.20.3.16 sequence()	231
G.20.3.17 sequenceKey()	231
G.20.3.18 setCursorAtKey()	232
G.20.3.19 sync()	232
G.20.3.20 vacuum()	232
G.20.4 Member Data Documentation	233
G.20.4.1 ARCHIVE_FILE_NAME	233
G.20.4.2 MANIFEST_FILE_NAME	233
G.20.4.3 OFFSET_RECORD_REMOVED	233
G.21 BiometricEvaluation::Memory::AutoArray< T > Class Template Reference	233
G.21.1 Detailed Description	235
G.21.2 Member Typedef Documentation	235
G.21.2.1 const_iterator	235
G.21.2.2 const_reference	235
G.21.2.3 iterator	235
G.21.2.4 reference	235
G.21.2.5 size_type	235
G.21.2.6 value_type	236
G.21.3 Constructor & Destructor Documentation	236
G.21.3.1 AutoArray() [1/4]	236
G.21.3.2 AutoArray() [2/4]	236
G.21.3.3 AutoArray() [3/4]	236
G.21.3.4 AutoArray() [4/4]	237
G.21.3.5 ~AutoArray()	237
G.21.4 Member Function Documentation	237
G.21.4.1 at() [1/2]	237
G.21.4.2 at() [2/2]	238
G.21.4.3 begin() [1/2]	238
G.21.4.4 begin() [2/2]	238
G.21.4.5 cbegin()	239
G.21.4.6 cend()	239
G.21.4.7 copy() [1/2]	239
G.21.4.8 copy() [2/2]	239
G.21.4.9 end() [1/2]	240
G.21.4.10 end() [2/2]	240
G.21.4.11 operator const T *()	240
G.21.4.12 operator T*()	240
G.21.4.13 operator=() [1/2]	241
G.21.4.14 operator=() [2/2]	241
G.21.4.15 operator[]() [1/2]	241
G.21.4.16 operator[]() [2/2]	242
G.21.4.17 resize()	242
G.21.4.18 size()	243
G.21.4.19 to_vector()	243
G.22 BiometricEvaluation::Memory::AutoArrayIterator< CONST, T > Class Template Reference	243
G.22.1 Detailed Description	244
G.22.2 Member Typedef Documentation	244
G.22.2.1 difference_type	245
G.22.2.2 iterator_category	245

G.22.2.3	pointer	245
G.22.2.4	reference	245
G.22.2.5	value_type	245
G.22.3	Constructor & Destructor Documentation	245
G.22.3.1	AutoArrayIterator() [1/3]	245
G.22.3.2	AutoArrayIterator() [2/3]	246
G.22.3.3	AutoArrayIterator() [3/3]	246
G.22.3.4	~AutoArrayIterator()	246
G.22.4	Member Function Documentation	246
G.22.4.1	operator"!=(246
G.22.4.2	operator*(246
G.22.4.3	operator+(246
G.22.4.4	operator+(247
G.22.4.5	operator++(247
G.22.4.6	operator++(247
G.22.4.7	operator+=(247
G.22.4.8	operator-(247
G.22.4.9	operator-(248
G.22.4.10	operator--(248
G.22.4.11	operator--(248
G.22.4.12	operator==(248
G.22.4.13	operator>(248
G.22.4.14	operator<(249
G.22.4.15	operator<=(249
G.22.4.16	operator=(249
G.22.4.17	operator=(249
G.22.4.18	operator==(249
G.22.4.19	operator>(250
G.22.4.20	operator>=(250
G.22.4.21	operator[]()	250
G.22.5	Friends And Related Function Documentation	250
G.22.5.1	operator+	250
G.22.5.2	operator-	250
G.23	BiometricEvaluation::Memory::AutoBuffer< T > Class Template Reference	251
G.23.1	Member Typedef Documentation	251
G.23.1.1	value_type	251
G.24	BiometricEvaluation::Image::BMP Class Reference	252
G.24.1	Detailed Description	252
G.24.2	Member Function Documentation	252
G.24.2.1	getRawData()	253
G.24.2.2	getRawGrayscaleData()	253
G.24.2.3	isBMP()	253
G.25	BiometricEvaluation::DataInterchange::AN2KRecord::CharacterSet Struct Reference	254
G.25.1	Constructor & Destructor Documentation	254
G.25.1.1	CharacterSet()	254
G.25.2	Member Data Documentation	254
G.25.2.1	commonName	255
G.25.2.2	identifier	255
G.25.2.3	version	255
G.26	BiometricEvaluation::Image::BMP::ColorTableEntry Struct Reference	255
G.26.1	Detailed Description	255

G.26.2 Member Data Documentation	255
G.26.2.1 blue	255
G.26.2.2 green	255
G.26.2.3 red	255
G.26.2.4 reserved	256
G.27 BiometricEvaluation::Process::CommandCenter< T, typename >::Command Class Reference	256
G.27.1 Detailed Description	256
G.27.2 Member Data Documentation	256
G.27.2.1 arguments	256
G.27.2.2 clientID	256
G.27.2.3 command	256
G.28 BiometricEvaluation::Process::CommandCenter< T, typename > Class Template Reference	256
G.28.1 Detailed Description	257
G.28.2 Constructor & Destructor Documentation	257
G.28.2.1 CommandCenter()	257
G.28.2.2 ~CommandCenter()	257
G.28.3 Member Function Documentation	258
G.28.3.1 disconnectClient()	258
G.28.3.2 getNextCommand()	258
G.28.3.3 hasPendingCommands()	258
G.28.3.4 sendResponse()	259
G.29 BiometricEvaluation::Process::CommandParser< T > Class Template Reference	259
G.29.1 Detailed Description	260
G.29.2 Constructor & Destructor Documentation	260
G.29.2.1 CommandParser()	260
G.29.2.2 ~CommandParser()	260
G.29.3 Member Function Documentation	260
G.29.3.1 getNextCommand()	260
G.29.3.2 getUsage()	261
G.29.3.3 parse()	261
G.29.3.4 setUsage()	261
G.30 BiometricEvaluation::IO::CompressedRecordStore Class Reference	261
G.30.1 Detailed Description	262
G.30.2 Constructor & Destructor Documentation	262
G.30.2.1 CompressedRecordStore() [1/4]	262
G.30.2.2 CompressedRecordStore() [2/4]	263
G.30.2.3 CompressedRecordStore() [3/4]	263
G.30.2.4 CompressedRecordStore() [4/4]	264
G.30.3 Member Function Documentation	264
G.30.3.1 changeDescription()	264
G.30.3.2 flush()	264
G.30.3.3 getCount()	265
G.30.3.4 getDescription()	265
G.30.3.5 getPathname()	265
G.30.3.6 getSpaceUsed()	265
G.30.3.7 insert()	266
G.30.3.8 length()	266
G.30.3.9 move()	267
G.30.3.10 operator=()	267
G.30.3.11 read()	267
G.30.3.12 remove()	268

G.30.3.13 sequence()	268
G.30.3.14 sequenceKey()	269
G.30.3.15 setCursorAtKey()	269
G.30.3.16 sync()	270
G.31 BiometricEvaluation::IO::Compressor Class Reference	270
G.31.1 Detailed Description	271
G.31.2 Member Enumeration Documentation	271
G.31.2.1 Kind	272
G.31.3 Constructor & Destructor Documentation	272
G.31.3.1 Compressor() [1/2]	272
G.31.3.2 ~Compressor()	272
G.31.3.3 Compressor() [2/2]	272
G.31.4 Member Function Documentation	272
G.31.4.1 compress() [1/6]	272
G.31.4.2 compress() [2/6]	273
G.31.4.3 compress() [3/6]	273
G.31.4.4 compress() [4/6]	274
G.31.4.5 compress() [5/6]	274
G.31.4.6 compress() [6/6]	274
G.31.4.7 createCompressor()	275
G.31.4.8 decompress() [1/6]	275
G.31.4.9 decompress() [2/6]	276
G.31.4.10 decompress() [3/6]	276
G.31.4.11 decompress() [4/6]	277
G.31.4.12 decompress() [5/6]	277
G.31.4.13 decompress() [6/6]	278
G.31.4.14 getOption()	278
G.31.4.15 getOptionAsInteger()	278
G.31.4.16 operator=()	280
G.31.4.17 removeOption()	280
G.31.4.18 setOption() [1/2]	280
G.31.4.19 setOption() [2/2]	281
G.32 BiometricEvaluation::Video::Container Class Reference	281
G.32.1 Detailed Description	282
G.32.2 Constructor & Destructor Documentation	282
G.32.2.1 Container() [1/3]	282
G.32.2.2 Container() [2/3]	282
G.32.2.3 Container() [3/3]	282
G.32.3 Member Function Documentation	283
G.32.3.1 getVideoStream()	283
G.33 BiometricEvaluation::Error::ConversionError Class Reference	283
G.33.1 Detailed Description	283
G.33.2 Constructor & Destructor Documentation	284
G.33.2.1 ConversionError() [1/2]	284
G.33.2.2 ConversionError() [2/2]	284
G.34 BiometricEvaluation::Image::Coordinate Struct Reference	284
G.34.1 Detailed Description	284
G.34.2 Constructor & Destructor Documentation	284
G.34.2.1 Coordinate()	284
G.34.3 Member Data Documentation	285
G.34.3.1 x	285

G.34.3.2	xDistance	285
G.34.3.3	y	285
G.34.3.4	yDistance	285
G.35	BiometricEvaluation::Feature::AN2K11EFS::CorePoint Struct Reference	285
G.36	BiometricEvaluation::Feature::CorePoint Struct Reference	285
G.36.1	Detailed Description	286
G.37	BiometricEvaluation::MPI::CSVDistributor Class Reference	286
G.37.1	Detailed Description	287
G.37.2	Constructor & Destructor Documentation	287
G.37.2.1	CSVDistributor()	287
G.37.3	Member Function Documentation	287
G.37.3.1	checkpointRestore()	287
G.37.3.2	checkpointSave()	287
G.37.3.3	createWorkPackage()	288
G.37.4	Member Data Documentation	288
G.37.4.1	CHECKPOINTLINECOUNT	288
G.37.4.2	CHECKPOINTRANDOMSEED	288
G.38	BiometricEvaluation::MPI::CSVProcessor Class Reference	288
G.38.1	Detailed Description	289
G.38.2	Constructor & Destructor Documentation	289
G.38.2.1	CSVProcessor()	289
G.38.3	Member Function Documentation	289
G.38.3.1	newProcessor()	289
G.38.3.2	performInitialization()	290
G.38.3.3	processLine()	290
G.38.3.4	processWorkPackage()	291
G.39	BiometricEvaluation::MPI::CSVResources Class Reference	291
G.39.1	Member Function Documentation	292
G.39.1.1	getDelimiter()	292
G.39.1.2	getNumLines()	292
G.39.1.3	getNumRemainingLines()	293
G.39.1.4	getRandomSeed()	293
G.39.1.5	randomizeLines()	293
G.39.1.6	readLine()	293
G.39.1.7	useBuffer()	294
G.39.2	Member Data Documentation	294
G.39.2.1	CHUNKSIZEPROPERTY	294
G.39.2.2	DELIMITERPROPERTY	294
G.39.2.3	INPUTCSVPROPERTY	294
G.39.2.4	RANDOMIZEPROPERTY	294
G.39.2.5	RANDOMSEEDPROPERTY	294
G.39.2.6	TRIMPROPERTY	294
G.39.2.7	USEBUFFERPROPERTY	294
G.40	BiometricEvaluation::Error::DataError Class Reference	295
G.40.1	Detailed Description	295
G.40.2	Constructor & Destructor Documentation	295
G.40.2.1	DataError() [1/2]	295
G.40.2.2	DataError() [2/2]	295
G.41	BiometricEvaluation::IO::DBRecordStore Class Reference	295
G.41.1	Detailed Description	296
G.41.2	Constructor & Destructor Documentation	296

G.41.2.1 DBRecordStore() [1/2]	296
G.41.2.2 DBRecordStore() [2/2]	297
G.41.3 Member Function Documentation	297
G.41.3.1 changeDescription()	297
G.41.3.2 flush()	298
G.41.3.3 getCount()	298
G.41.3.4 getDescription()	298
G.41.3.5 getPathname()	298
G.41.3.6 getSpaceUsed()	299
G.41.3.7 insert()	299
G.41.3.8 length()	299
G.41.3.9 move()	300
G.41.3.10 read()	300
G.41.3.11 remove()	301
G.41.3.12 sequence()	301
G.41.3.13 sequenceKey()	302
G.41.3.14 setCursorAtKey()	302
G.41.3.15 sync()	302
G.42 BiometricEvaluation::Feature::AN2K11EFS::DeltaPoint Struct Reference	304
G.42.1 Detailed Description	304
G.43 BiometricEvaluation::Feature::DeltaPoint Struct Reference	304
G.43.1 Detailed Description	305
G.44 BiometricEvaluation::MPI::Distributor Class Reference	305
G.44.1 Detailed Description	306
G.44.2 Constructor & Destructor Documentation	306
G.44.2.1 Distributor()	306
G.44.3 Member Function Documentation	306
G.44.3.1 checkpointRestore()	306
G.44.3.2 checkpointSave()	307
G.44.3.3 createWorkPackage()	307
G.44.3.4 getCheckpointData()	307
G.44.3.5 getLogsheet()	307
G.44.3.6 start()	308
G.44.4 Member Data Documentation	308
G.44.4.1 CHECKPOINTFILENAME	308
G.44.4.2 CHECKPOINTPID	308
G.44.4.3 CHECKPOINTREASON	308
G.45 BiometricEvaluation::DataInterchange::AN2KRecord::DomainName Struct Reference	308
G.45.1 Detailed Description	308
G.45.2 Constructor & Destructor Documentation	308
G.45.2.1 DomainName()	309
G.45.3 Member Data Documentation	309
G.45.3.1 identifier	309
G.45.3.2 version	309
G.46 BiometricEvaluation::Feature::AN2K7Minutiae::PatternClassification::Entry Struct Reference	309
G.46.1 Constructor & Destructor Documentation	309
G.46.1.1 Entry()	309
G.46.2 Member Data Documentation	310
G.46.2.1 code	310
G.46.2.2 standard	310
G.47 BiometricEvaluation::Error::Exception Class Reference	310

G.47.1 Detailed Description	311
G.47.2 Constructor & Destructor Documentation	311
G.47.2.1 Exception() [1/2]	311
G.47.2.2 Exception() [2/2]	312
G.47.3 Member Function Documentation	312
G.47.3.1 what()	312
G.47.3.2 whatString()	312
G.48 BiometricEvaluation::MPI::Exception Class Reference	312
G.48.1 Constructor & Destructor Documentation	313
G.48.1.1 Exception() [1/2]	313
G.48.1.2 Exception() [2/2]	313
G.48.1.3 ~Exception()	313
G.49 BiometricEvaluation::Feature::AN2K11EFS::ExtendedFeatureSet Class Reference	313
G.49.1 Detailed Description	314
G.49.2 Constructor & Destructor Documentation	314
G.49.2.1 ExtendedFeatureSet() [1/2]	314
G.49.2.2 ExtendedFeatureSet() [2/2]	314
G.49.3 Member Function Documentation	315
G.49.3.1 getCPS()	315
G.49.3.2 getDPS()	315
G.49.3.3 getImageInfo()	315
G.49.3.4 getMPS()	316
G.49.3.5 getMRCI()	316
G.49.3.6 getNFP()	316
G.50 BiometricEvaluation::Error::FileError Class Reference	316
G.50.1 Detailed Description	317
G.50.2 Constructor & Destructor Documentation	317
G.50.2.1 FileError() [1/2]	317
G.50.2.2 FileError() [2/2]	317
G.51 BiometricEvaluation::IO::FileLogCabinet Class Reference	317
G.51.1 Detailed Description	317
G.51.2 Constructor & Destructor Documentation	317
G.51.2.1 FileLogCabinet() [1/2]	317
G.51.2.2 FileLogCabinet() [2/2]	318
G.51.3 Member Function Documentation	318
G.51.3.1 getCount()	318
G.51.3.2 getDescription()	318
G.51.3.3 getPathname()	318
G.51.3.4 newLogsheet()	318
G.52 BiometricEvaluation::IO::FileLogsheet Class Reference	319
G.52.1 Detailed Description	320
G.52.2 Constructor & Destructor Documentation	320
G.52.2.1 FileLogsheet() [1/3]	320
G.52.2.2 FileLogsheet() [2/3]	321
G.52.2.3 ~FileLogsheet()	321
G.52.2.4 FileLogsheet() [3/3]	321
G.52.3 Member Function Documentation	322
G.52.3.1 mergeLogsheets()	322
G.52.3.2 operator=()	322
G.52.3.3 sequence()	322
G.52.3.4 sync()	323

G.52.3.5	trim()	323
G.52.3.6	updateCursor()	323
G.52.3.7	write()	324
G.52.3.8	writeComment()	324
G.52.3.9	writeDebug()	324
G.52.4	Member Data Documentation	325
G.52.4.1	_cursor	325
G.52.4.2	_sequenceFile	325
G.52.4.3	_theLogFile	325
G.52.4.4	BE_FILELOGSHEET_SEQ_NEXT	325
G.52.4.5	BE_FILELOGSHEET_SEQ_START	325
G.53	BiometricEvaluation::IO::FileRecordStore Class Reference	325
G.53.1	Detailed Description	326
G.53.2	Constructor & Destructor Documentation	326
G.53.2.1	FileRecordStore() [1/2]	326
G.53.2.2	FileRecordStore() [2/2]	327
G.53.3	Member Function Documentation	327
G.53.3.1	changeDescription()	327
G.53.3.2	flush()	328
G.53.3.3	getCount()	328
G.53.3.4	getDescription()	328
G.53.3.5	getPathname()	328
G.53.3.6	getSpaceUsed()	329
G.53.3.7	insert()	329
G.53.3.8	length()	329
G.53.3.9	move()	330
G.53.3.10	read()	330
G.53.3.11	remove()	331
G.53.3.12	replace()	331
G.53.3.13	sequence()	331
G.53.3.14	sequenceKey()	332
G.53.3.15	setCursorAtKey()	333
G.53.3.16	sync()	333
G.54	BiometricEvaluation::Feature::AN2K7Minutiae::FingerprintReadingSystem Struct Reference	333
G.54.1	Detailed Description	333
G.54.2	Member Data Documentation	333
G.54.2.1	equipment	334
G.54.2.2	method	334
G.54.2.3	name	334
G.55	BiometricEvaluation::Finger::AN2KViewCapture::FingerSegmentPosition Struct Reference	334
G.55.1	Detailed Description	334
G.55.2	Constructor & Destructor Documentation	334
G.55.2.1	FingerSegmentPosition()	334
G.55.3	Member Data Documentation	335
G.55.3.1	coordinates	335
G.55.3.2	fingerPosition	335
G.56	BiometricEvaluation::Process::ForkManager Class Reference	335
G.56.1	Detailed Description	336
G.56.2	Constructor & Destructor Documentation	336
G.56.2.1	ForkManager()	336
G.56.3	Member Function Documentation	336

G.56.3.1	addWorker()	336
G.56.3.2	broadcastSignal()	337
G.56.3.3	defaultExitCallback()	337
G.56.3.4	getIsWorkingStatus()	337
G.56.3.5	responsibleFor()	338
G.56.3.6	setExitCallback()	338
G.56.3.7	setExitStatus()	338
G.56.3.8	setNotWorking()	339
G.56.3.9	startWorker()	339
G.56.3.10	startWorkers()	339
G.56.3.11	stopWorker()	340
G.56.3.12	waitForWorkerExit()	340
G.56.4	Member Data Documentation	341
G.56.4.1	FORKMANAGERS	341
G.57	BiometricEvaluation::Process::ForkWorkerController Class Reference	341
G.57.1	Detailed Description	342
G.57.2	Member Function Documentation	342
G.57.2.1	_stop()	342
G.57.2.2	everWorked()	342
G.57.2.3	getPID()	342
G.57.2.4	isWorking()	343
G.57.2.5	reset()	343
G.57.3	Friends And Related Function Documentation	343
G.57.3.1	ForkManager::addWorker	343
G.57.3.2	ForkManager::setExitStatus	344
G.57.3.3	ForkManager::startWorker	344
G.57.3.4	ForkManager::startWorkers	345
G.57.3.5	ForkManager::stopWorker	345
G.58	BiometricEvaluation::Feature::AN2K11EFS::FPPPosition Struct Reference	345
G.58.1	Detailed Description	346
G.58.2	Member Data Documentation	346
G.58.2.1	fgp	346
G.58.2.2	fsm	346
G.58.2.3	ocf	346
G.58.2.4	sgp	346
G.59	BiometricEvaluation::Video::Frame Struct Reference	346
G.60	BiometricEvaluation::Feature::FrictionRidgeGeneralizedPosition Struct Reference	346
G.60.1	Detailed Description	347
G.61	BiometricEvaluation::IO::GZip Class Reference	347
G.61.1	Detailed Description	348
G.61.2	Constructor & Destructor Documentation	348
G.61.2.1	GZip()	348
G.61.3	Member Function Documentation	349
G.61.3.1	compress() [1/6]	349
G.61.3.2	compress() [2/6]	349
G.61.3.3	compress() [3/6]	350
G.61.3.4	compress() [4/6]	350
G.61.3.5	compress() [5/6]	350
G.61.3.6	compress() [6/6]	351
G.61.3.7	decompress() [1/6]	351
G.61.3.8	decompress() [2/6]	352

G.61.3.9 decompress() [3/6]	352
G.61.3.10 decompress() [4/6]	353
G.61.3.11 decompress() [5/6]	353
G.61.3.12 decompress() [6/6]	354
G.61.3.13 operator=()	354
G.61.4 Member Data Documentation	354
G.61.4.1 CHUNK_SIZE	354
G.61.4.2 COMPRESSION_LEVEL	354
G.61.4.3 COMPRESSION_METHOD	355
G.61.4.4 COMPRESSION_STRATEGY	355
G.61.4.5 INPUT_DATA_TYPE	355
G.61.4.6 MEMORY_LEVEL	355
G.61.4.7 WINDOW_BITS	355
G.62 BiometricEvaluation::Image::Image Class Reference	355
G.62.1 Detailed Description	357
G.62.2 Constructor & Destructor Documentation	358
G.62.2.1 Image() [1/2]	358
G.62.2.2 Image() [2/2]	358
G.62.3 Member Function Documentation	359
G.62.3.1 getBitDepth()	359
G.62.3.2 getColorDepth()	359
G.62.3.3 getCompressionAlgorithm() [1/4]	359
G.62.3.4 getCompressionAlgorithm() [2/4]	359
G.62.3.5 getCompressionAlgorithm() [3/4]	360
G.62.3.6 getCompressionAlgorithm() [4/4]	360
G.62.3.7 getData()	361
G.62.3.8 getDataPointer()	361
G.62.3.9 getDataSize()	361
G.62.3.10 getDimensions()	361
G.62.3.11 getRawData() [1/2]	361
G.62.3.12 getRawData() [2/2]	362
G.62.3.13 getRawGrayscaleData()	362
G.62.3.14 getRawImage()	363
G.62.3.15 getResolution()	363
G.62.3.16 hasAlphaChannel()	364
G.62.3.17 openImage() [1/3]	364
G.62.3.18 openImage() [2/3]	364
G.62.3.19 openImage() [3/3]	365
G.62.3.20 setBitDepth()	365
G.62.3.21 setColorDepth()	365
G.62.3.22 setDimensions()	366
G.62.3.23 setHasAlphaChannel()	366
G.62.3.24 setResolution()	366
G.62.3.25 valueInColorspace()	366
G.63 BiometricEvaluation::Feature::AN2K11EFS::ImageInfo Struct Reference	367
G.63.1 Detailed Description	367
G.63.2 Member Data Documentation	367
G.63.2.1 fpp	367
G.63.2.2 ort	367
G.63.2.3 plr	367
G.63.2.4 roi	368

G.63.2.5	trv	368
G.64	BiometricEvaluation::Feature::INCITSMinutiae Class Reference	368
G.64.1	Detailed Description	369
G.64.2	Constructor & Destructor Documentation	370
G.64.2.1	INCITSMinutiae()	370
G.64.3	Member Function Documentation	370
G.64.3.1	setCorePointSet()	370
G.64.3.2	setDeltaPointSet()	370
G.64.3.3	setMinutiaPoints()	371
G.64.3.4	setRidgeCountItems()	371
G.65	BiometricEvaluation::Face::INCITSView Class Reference	371
G.65.1	Detailed Description	372
G.65.2	Constructor & Destructor Documentation	372
G.65.2.1	INCITSView() [1/2]	373
G.65.2.2	INCITSView() [2/2]	373
G.65.3	Member Function Documentation	373
G.65.3.1	getColorSpace()	373
G.65.3.2	getDeviceType()	374
G.65.3.3	getEyeColor()	374
G.65.3.4	getFeaturePointSet()	374
G.65.3.5	getFIDData()	374
G.65.3.6	getGender()	374
G.65.3.7	getHairColor()	375
G.65.3.8	getImageDataType()	375
G.65.3.9	getImageType()	375
G.65.3.10	getPoseAngle()	375
G.65.3.11	getPropertySet()	375
G.65.3.12	getSourceType()	376
G.65.3.13	propertiesConsidered()	376
G.65.3.14	readFaceView()	376
G.65.3.15	readHeader()	376
G.66	BiometricEvaluation::Iris::INCITSView Class Reference	377
G.66.1	Detailed Description	379
G.66.2	Constructor & Destructor Documentation	379
G.66.2.1	INCITSView() [1/2]	379
G.66.2.2	INCITSView() [2/2]	379
G.66.3	Member Function Documentation	380
G.66.3.1	getCameraRange()	380
G.66.3.2	getCaptureDateString()	380
G.66.3.3	getCaptureDeviceTechnology()	380
G.66.3.4	getCaptureDeviceType()	380
G.66.3.5	getCaptureDeviceVendor()	380
G.66.3.6	getCertificationFlag()	381
G.66.3.7	getEyeLabel()	381
G.66.3.8	getIIRData()	381
G.66.3.9	getImageProperties()	381
G.66.3.10	getImageType()	381
G.66.3.11	getIrisCenterInfo()	382
G.66.3.12	getQualitySet()	382
G.66.3.13	getRollAngleInfo()	382
G.66.3.14	readHeader()	383

G.66.3.15 readIrisView()	383
G.67 BiometricEvaluation::Finger::INCITSView Class Reference	383
G.67.1 Detailed Description	386
G.67.2 Constructor & Destructor Documentation	386
G.67.2.1 INCITSView() [1/2]	386
G.67.2.2 INCITSView() [2/2]	386
G.67.3 Member Function Documentation	387
G.67.3.1 convertImpression()	387
G.67.3.2 convertPosition()	387
G.67.3.3 getCaptureEquipmentID()	388
G.67.3.4 getEDBLength()	388
G.67.3.5 getFIRData()	388
G.67.3.6 getFMRData()	388
G.67.3.7 getFMRReservedByte()	388
G.67.3.8 getImpressionType()	388
G.67.3.9 getMinutiaeReservedData()	389
G.67.3.10 getNumFingerViews()	389
G.67.3.11 getPosition()	389
G.67.3.12 getProductIDOwner()	389
G.67.3.13 getProductIDType()	389
G.67.3.14 getQuality()	390
G.67.3.15 getRecordLength()	390
G.67.3.16 getViewNumber()	390
G.67.3.17 isAppendixFCompliant()	390
G.67.3.18 readCoreDeltaData()	390
G.67.3.19 readExtendedDataBlock()	391
G.67.3.20 readFMRHeader()	391
G.67.3.21 readFVMR()	391
G.67.3.22 readMinutiaeDataPoints()	392
G.67.3.23 readRidgeCountData()	392
G.67.3.24 setAppendixFCompliance()	393
G.67.3.25 setCaptureEquipmentID()	393
G.67.3.26 setCBEFFProductIDs()	393
G.67.3.27 setImpressionType()	393
G.67.3.28 setMinutiaeData()	394
G.67.3.29 setMinutiaeReservedData()	394
G.67.3.30 setPosition()	394
G.67.3.31 setQuality()	394
G.67.3.32 setViewNumber()	395
G.68 BiometricEvaluation::Memory::IndexedBuffer Class Reference	395
G.68.1 Detailed Description	396
G.68.2 Constructor & Destructor Documentation	396
G.68.2.1 IndexedBuffer() [1/4]	396
G.68.2.2 IndexedBuffer() [2/4]	396
G.68.2.3 IndexedBuffer() [3/4]	396
G.68.2.4 IndexedBuffer() [4/4]	397
G.68.2.5 ~IndexedBuffer()	397
G.68.3 Member Function Documentation	397
G.68.3.1 get()	397
G.68.3.2 getIndex()	397
G.68.3.3 getSize()	397

G.68.3.4	scan()	398
G.68.3.5	scanBeU16Val()	398
G.68.3.6	scanBeU32Val()	398
G.68.3.7	scanU16Val()	399
G.68.3.8	scanU32Val()	399
G.68.3.9	scanU64Val()	399
G.68.3.10	scanU8Val()	399
G.68.3.11	setIndex()	400
G.69	BiometricEvaluation::Face::ISO2005View Class Reference	400
G.69.1	Detailed Description	401
G.69.2	Constructor & Destructor Documentation	401
G.69.2.1	ISO2005View() [1/2]	401
G.69.2.2	ISO2005View() [2/2]	401
G.69.3	Member Function Documentation	402
G.69.3.1	readISOHeader()	402
G.70	BiometricEvaluation::Finger::ISO2005View Class Reference	402
G.70.1	Detailed Description	403
G.70.2	Constructor & Destructor Documentation	403
G.70.2.1	ISO2005View() [1/2]	403
G.70.2.2	ISO2005View() [2/2]	403
G.70.3	Member Function Documentation	404
G.70.3.1	readCoreDeltaData()	404
G.71	BiometricEvaluation::Iris::ISO2011View Class Reference	404
G.71.1	Detailed Description	405
G.71.2	Constructor & Destructor Documentation	405
G.71.2.1	ISO2011View() [1/2]	405
G.71.2.2	ISO2011View() [2/2]	406
G.72	BiometricEvaluation::Image::JPEG Class Reference	406
G.72.1	Detailed Description	407
G.72.2	Member Function Documentation	407
G.72.2.1	getRawData()	407
G.72.2.2	getRawGrayscaleData()	407
G.72.2.3	isJPEG()	408
G.73	BiometricEvaluation::Image::JPEG2000 Class Reference	408
G.73.1	Detailed Description	409
G.73.2	Constructor & Destructor Documentation	409
G.73.2.1	JPEG2000()	409
G.73.3	Member Function Documentation	409
G.73.3.1	getRawData()	409
G.73.3.2	getRawGrayscaleData()	410
G.73.3.3	isJPEG2000()	410
G.74	BiometricEvaluation::Image::JPEGL Class Reference	411
G.74.1	Detailed Description	411
G.74.2	Member Function Documentation	411
G.74.2.1	getRawData()	411
G.74.2.2	getRawGrayscaleData()	412
G.74.2.3	isJPEGL()	412
G.75	BiometricEvaluation::IO::ListRecordStore Class Reference	413
G.75.1	Detailed Description	414
G.75.2	Constructor & Destructor Documentation	414
G.75.2.1	ListRecordStore()	414

G.75.2.2	~ListRecordStore()	414
G.75.3	Member Function Documentation	414
G.75.3.1	changeDescription()	414
G.75.3.2	flush()	415
G.75.3.3	getCount()	415
G.75.3.4	getDescription()	415
G.75.3.5	getPathname()	415
G.75.3.6	getSpaceUsed()	416
G.75.3.7	insert()	416
G.75.3.8	length()	416
G.75.3.9	move()	417
G.75.3.10	read()	417
G.75.3.11	remove()	418
G.75.3.12	replace()	418
G.75.3.13	sequence()	418
G.75.3.14	sequenceKey()	419
G.75.3.15	setCursorAtKey()	420
G.75.3.16	sync()	420
G.76	BiometricEvaluation::IO::Logsheet Class Reference	420
G.76.1	Detailed Description	422
G.76.2	Member Enumeration Documentation	422
G.76.2.1	Kind	422
G.76.3	Constructor & Destructor Documentation	423
G.76.3.1	~Logsheet()	423
G.76.4	Member Function Documentation	423
G.76.4.1	getAutoSync()	423
G.76.4.2	getCommentCommit()	423
G.76.4.3	getCommit()	423
G.76.4.4	getCurrentEntry()	423
G.76.4.5	getCurrentEntryNumber()	424
G.76.4.6	getCurrentEntryNumberAsString()	424
G.76.4.7	getDebugCommit()	424
G.76.4.8	getTypeFromURL()	424
G.76.4.9	lineIsComment()	425
G.76.4.10	lineIsDebug()	425
G.76.4.11	lineIsEntry()	425
G.76.4.12	newEntry()	425
G.76.4.13	resetCurrentEntry()	426
G.76.4.14	setAutoSync()	426
G.76.4.15	setCommentCommit()	426
G.76.4.16	setCommit()	426
G.76.4.17	setDebugCommit()	427
G.76.4.18	sync()	427
G.76.4.19	trim()	427
G.76.4.20	write()	428
G.76.4.21	writeComment()	428
G.76.4.22	writeDebug()	428
G.76.5	Member Data Documentation	429
G.76.5.1	CommentDelimiter	429
G.76.5.2	DebugDelimiter	429
G.76.5.3	DescriptionTag	429

G.76.5.4	EntryDelimiter	429
G.76.5.5	FILEURLSCHEME	429
G.76.5.6	SYSLOGURLSCHEME	429
G.77	BiometricEvaluation::Process::Manager Class Reference	430
G.77.1	Detailed Description	431
G.77.2	Member Function Documentation	431
G.77.2.1	addWorker()	431
G.77.2.2	broadcastMessage()	431
G.77.2.3	getNextMessage()	432
G.77.2.4	getNumActiveWorkers()	432
G.77.2.5	getNumCompletedWorkers()	432
G.77.2.6	getTotalWorkers()	433
G.77.2.7	reset()	433
G.77.2.8	startWorker()	433
G.77.2.9	startWorkers()	434
G.77.2.10	stopWorker()	434
G.77.2.11	waitForMessage()	435
G.77.2.12	waitForWorkerExit()	435
G.77.3	Member Data Documentation	435
G.77.3.1	_pendingExit	435
G.77.3.2	_workers	435
G.78	BiometricEvaluation::Error::MemoryError Class Reference	436
G.78.1	Detailed Description	436
G.78.2	Constructor & Destructor Documentation	436
G.78.2.1	MemoryError() [1/2]	436
G.78.2.2	MemoryError() [2/2]	436
G.79	BiometricEvaluation::Process::MessageCenter Class Reference	436
G.79.1	Detailed Description	437
G.79.2	Constructor & Destructor Documentation	437
G.79.2.1	MessageCenter()	437
G.79.3	Member Function Documentation	437
G.79.3.1	disconnectClient()	437
G.79.3.2	getNextMessage()	437
G.79.3.3	hasUnseenMessages()	438
G.79.3.4	sendResponse()	438
G.79.4	Member Data Documentation	438
G.79.4.1	CONNECTION_BACKLOG	438
G.79.4.2	DEFAULT_PORT	439
G.79.4.3	DEFAULT_TIMEOUT	439
G.79.4.4	MAX_MESSAGE_LENGTH	439
G.80	BiometricEvaluation::Process::MessageCenterListener Class Reference	439
G.80.1	Detailed Description	439
G.80.2	Member Function Documentation	439
G.80.2.1	workerMain()	440
G.80.3	Member Data Documentation	440
G.80.3.1	PARAM_PORT	440
G.81	BiometricEvaluation::Process::MessageCenterReceiver Class Reference	440
G.81.1	Detailed Description	441
G.81.2	Constructor & Destructor Documentation	441
G.81.2.1	MessageCenterReceiver()	441
G.81.2.2	~MessageCenterReceiver()	441

G.81.3 Member Function Documentation	441
G.81.3.1 workerMain()	441
G.81.4 Member Data Documentation	441
G.81.4.1 MSG_DISCONNECT	441
G.81.4.2 PARAM_CLIENT_ID	441
G.81.4.3 PARAM_CLIENT_SOCKET	441
G.82 BiometricEvaluation::Feature::Minutiae Class Reference	442
G.82.1 Detailed Description	442
G.83 BiometricEvaluation::Feature::AN2K11EFS::MinutiaeRidgeCount Struct Reference	442
G.83.1 Detailed Description	443
G.83.2 Member Data Documentation	443
G.83.2.1 mia	443
G.83.2.2 mib	443
G.83.2.3 mir	443
G.83.2.4 mrn	443
G.83.2.5 mrs	443
G.84 BiometricEvaluation::Feature::AN2K11EFS::MinutiaeRidgeCountConfidence Struct Reference	443
G.84.1 Detailed Description	443
G.85 BiometricEvaluation::Feature::AN2K11EFS::MinutiaeRidgeCountInfo Struct Reference	444
G.85.1 Detailed Description	444
G.86 BiometricEvaluation::Feature::MinutiaPoint Struct Reference	444
G.86.1 Detailed Description	444
G.87 BiometricEvaluation::Feature::AN2K11EFS::MinutiaPoint Struct Reference	445
G.87.1 Detailed Description	445
G.87.2 Member Data Documentation	445
G.87.2.1 mdu	445
G.87.2.2 mru	445
G.88 BiometricEvaluation::Feature::MPEGFacePoint Struct Reference	445
G.88.1 Detailed Description	446
G.89 BiometricEvaluation::Memory::MutableIndexedBuffer Class Reference	446
G.89.1 Detailed Description	446
G.89.2 Constructor & Destructor Documentation	446
G.89.2.1 MutableIndexedBuffer() [1/3]	447
G.89.2.2 MutableIndexedBuffer() [2/3]	447
G.89.2.3 MutableIndexedBuffer() [3/3]	447
G.89.2.4 ~MutableIndexedBuffer()	447
G.89.3 Member Function Documentation	447
G.89.3.1 get()	447
G.89.3.2 push()	448
G.89.3.3 pushBeU16Val()	448
G.89.3.4 pushBeU32Val()	448
G.89.3.5 pushU16Val()	449
G.89.3.6 pushU32Val()	449
G.89.3.7 pushU64Val()	450
G.89.3.8 pushU8Val()	450
G.90 BiometricEvaluation::Image::NetPBM Class Reference	450
G.90.1 Detailed Description	451
G.90.2 Member Function Documentation	452
G.90.2.1 ASCIIBitmapTo8Bit()	452
G.90.2.2 ASCIIPixmapToBinaryPixmap()	452
G.90.2.3 BinaryBitmapTo8Bit()	453

G.90.2.4	getNextValue()	453
G.90.2.5	getRawData()	454
G.90.2.6	getRawGrayscaleData()	454
G.90.2.7	isNetPBM()	455
G.90.2.8	skipComment()	455
G.90.2.9	skipLine()	456
G.91	BiometricEvaluation::Feature::AN2K11EFS::NoFeaturesPresent Struct Reference	456
G.91.1	Detailed Description	456
G.92	BiometricEvaluation::Error::NotImplemented Class Reference	456
G.92.1	Detailed Description	457
G.92.2	Constructor & Destructor Documentation	457
G.92.2.1	NotImplemented() [1/2]	457
G.92.2.2	NotImplemented() [2/2]	457
G.93	BiometricEvaluation::Error::ObjectDoesNotExist Class Reference	457
G.93.1	Detailed Description	458
G.93.2	Constructor & Destructor Documentation	458
G.93.2.1	ObjectDoesNotExist() [1/2]	458
G.93.2.2	ObjectDoesNotExist() [2/2]	458
G.94	BiometricEvaluation::Error::ObjectExists Class Reference	458
G.94.1	Detailed Description	458
G.94.2	Constructor & Destructor Documentation	458
G.94.2.1	ObjectExists() [1/2]	459
G.94.2.2	ObjectExists() [2/2]	459
G.95	BiometricEvaluation::Error::ObjectIsClosed Class Reference	459
G.95.1	Detailed Description	459
G.95.2	Constructor & Destructor Documentation	459
G.95.2.1	ObjectIsClosed() [1/2]	459
G.95.2.2	ObjectIsClosed() [2/2]	459
G.96	BiometricEvaluation::Error::ObjectIsOpen Class Reference	460
G.96.1	Detailed Description	460
G.96.2	Constructor & Destructor Documentation	460
G.96.2.1	ObjectIsOpen() [1/2]	460
G.96.2.2	ObjectIsOpen() [2/2]	460
G.97	BiometricEvaluation::Memory::OrderedMap< Key, T > Class Template Reference	460
G.97.1	Detailed Description	461
G.97.2	Constructor & Destructor Documentation	461
G.97.2.1	OrderedMap()	462
G.97.2.2	~OrderedMap()	462
G.97.3	Member Function Documentation	462
G.97.3.1	begin() [1/2]	462
G.97.3.2	begin() [2/2]	462
G.97.3.3	cbegin()	462
G.97.3.4	cend()	463
G.97.3.5	end() [1/2]	463
G.97.3.6	end() [2/2]	463
G.97.3.7	erase() [1/2]	463
G.97.3.8	erase() [2/2]	463
G.97.3.9	find()	464
G.97.3.10	key_eq()	464
G.97.3.11	keyExists()	464
G.97.3.12	operator[]()	465

G.97.3.13 push_back()	465
G.97.3.14 size()	465
G.98 BiometricEvaluation::Memory::OrderedMapConstIterator< Key, T > Class Template Reference	465
G.98.1 Detailed Description	466
G.98.2 Member Typedef Documentation	466
G.98.2.1 difference_type	466
G.98.2.2 iterator_category	466
G.98.2.3 pointer	467
G.98.2.4 reference	467
G.98.2.5 value_type	467
G.98.3 Constructor & Destructor Documentation	467
G.98.3.1 OrderedMapConstIterator() [1/2]	467
G.98.3.2 OrderedMapConstIterator() [2/2]	467
G.98.3.3 ~OrderedMapConstIterator()	467
G.98.4 Member Function Documentation	467
G.98.4.1 operator"!=(468
G.98.4.2 operator*(468
G.98.4.3 operator++() [1/2]	468
G.98.4.4 operator++() [2/2]	468
G.98.4.5 operator--() [1/2]	468
G.98.4.6 operator--() [2/2]	469
G.98.4.7 operator->()	469
G.98.4.8 operator==(469
G.99 BiometricEvaluation::Memory::OrderedMapIterator< Key, T > Class Template Reference	469
G.99.1 Detailed Description	470
G.99.2 Member Typedef Documentation	470
G.99.2.1 difference_type	470
G.99.2.2 iterator_category	470
G.99.2.3 pointer	470
G.99.2.4 reference	470
G.99.2.5 value_type	471
G.99.3 Constructor & Destructor Documentation	471
G.99.3.1 OrderedMapIterator()	471
G.99.3.2 ~OrderedMapIterator()	471
G.99.4 Member Function Documentation	471
G.99.4.1 operator"!=(471
G.99.4.2 operator*(471
G.99.4.3 operator++() [1/2]	472
G.99.4.4 operator++() [2/2]	472
G.99.4.5 operator--() [1/2]	472
G.99.4.6 operator--() [2/2]	472
G.99.4.7 operator->()	472
G.99.4.8 operator==(472
G.100 BiometricEvaluation::Feature::AN2K11EFS::Orientation Struct Reference	473
G.100.1 Detailed Description	473
G.100.2 Member Data Documentation	473
G.100.2.1 eod	473
G.100.2.2 EODDefault	473
G.100.2.3 euc	473
G.100.2.4 EUCDefault	473
G.100.2.5 is_default	474

G.101BiometricEvaluation::Error::ParameterError Class Reference	474
G.101.1Detailed Description	474
G.101.2Constructor & Destructor Documentation	474
G.101.2.1 ParameterError() [1/2]	474
G.101.2.2 ParameterError() [2/2]	474
G.102BiometricEvaluation::Feature::AN2K7Minutiae::PatternClassification Class Reference	474
G.102.1Detailed Description	475
G.103BiometricEvaluation::IO::PersistentRecordStoreUnion Class Reference	475
G.103.1Constructor & Destructor Documentation	475
G.103.1.1 PersistentRecordStoreUnion() [1/3]	475
G.103.1.2 PersistentRecordStoreUnion() [2/3]	476
G.103.1.3 PersistentRecordStoreUnion() [3/3]	476
G.103.1.4 ~PersistentRecordStoreUnion()	476
G.104BiometricEvaluation::Image::PNG Class Reference	476
G.104.1Detailed Description	477
G.104.2Member Function Documentation	477
G.104.2.1 getRawData()	477
G.104.2.2 getRawGrayscaleData()	477
G.104.2.3 isPNG()	478
G.105BiometricEvaluation::Feature::Sort::Polar Class Reference	478
G.105.1Detailed Description	479
G.105.2Constructor & Destructor Documentation	479
G.105.2.1 Polar()	479
G.105.3Member Function Documentation	479
G.105.3.1 centerOfImage()	479
G.105.3.2 centerOfMinutiaeMass()	479
G.105.3.3 operator>()	480
G.106BiometricEvaluation::Face::PoseAngle Struct Reference	480
G.106.1Detailed Description	480
G.107BiometricEvaluation::Process::POSIXThreadManager Class Reference	480
G.107.1Detailed Description	481
G.107.2Constructor & Destructor Documentation	481
G.107.2.1 POSIXThreadManager()	481
G.107.3Member Function Documentation	481
G.107.3.1 addWorker()	481
G.107.3.2 startWorker()	482
G.107.3.3 startWorkers()	482
G.107.3.4 stopWorker()	483
G.107.3.5 waitForWorkerExit()	483
G.108BiometricEvaluation::Process::POSIXThreadWorkerController Class Reference	483
G.108.1Detailed Description	484
G.108.2Member Function Documentation	484
G.108.2.1 everWorked()	484
G.108.2.2 isWorking()	484
G.108.2.3 reset()	485
G.109BiometricEvaluation::View::AN2KViewVariableResolution::PrintPositionCoordinate Struct Reference	486
G.109.1Detailed Description	486
G.109.2Member Data Documentation	486
G.109.2.1 coordinates	486
G.109.2.2 fingerView	486

G.109.2.3 segment	486
G.110 BiometricEvaluation::IO::Properties Class Reference	486
G.110.1 Detailed Description	488
G.110.2 Constructor & Destructor Documentation	488
G.110.2.1 Properties() [1/2]	488
G.110.2.2 Properties() [2/2]	488
G.110.2.3 ~Properties()	488
G.110.3 Member Function Documentation	489
G.110.3.1 getMode()	489
G.110.3.2 getProperty()	489
G.110.3.3 getPropertyAsDouble()	489
G.110.3.4 getPropertyAsInteger()	490
G.110.3.5 getPropertyKeys()	490
G.110.3.6 initWithBuffer() [1/2]	490
G.110.3.7 initWithBuffer() [2/2]	491
G.110.3.8 removeProperty()	491
G.110.3.9 setProperty()	491
G.110.3.10 getPropertyFromBoolean()	492
G.110.3.11 getPropertyFromDouble()	492
G.110.3.12 getPropertyFromInteger()	493
G.111 BiometricEvaluation::IO::PropertiesFile Class Reference	493
G.111.1 Detailed Description	494
G.111.2 Constructor & Destructor Documentation	494
G.111.2.1 PropertiesFile() [1/2]	494
G.111.2.2 ~PropertiesFile()	495
G.111.2.3 PropertiesFile() [2/2]	495
G.111.3 Member Function Documentation	495
G.111.3.1 changeName()	495
G.111.3.2 operator=()	495
G.111.3.3 sync()	496
G.112 BiometricEvaluation::Feature::Sort::Quality Class Reference	496
G.112.1 Detailed Description	496
G.113 BiometricEvaluation::Iris::INCITSView::QualitySubBlock Struct Reference	496
G.113.1 Detailed Description	497
G.114 BiometricEvaluation::Image::Raw Class Reference	497
G.114.1 Detailed Description	497
G.114.2 Member Function Documentation	497
G.114.2.1 getRawData()	497
G.114.2.2 getRawGrayscaleData()	498
G.115 BiometricEvaluation::MPI::Receiver Class Reference	498
G.115.1 Detailed Description	499
G.115.2 Constructor & Destructor Documentation	499
G.115.2.1 Receiver()	499
G.115.3 Member Function Documentation	499
G.115.3.1 start()	499
G.116 BiometricEvaluation::IO::RecordStore::Record Struct Reference	500
G.116.1 Constructor & Destructor Documentation	500
G.116.1.1 Record() [1/2]	500
G.116.1.2 Record() [2/2]	500
G.117 BiometricEvaluation::MPI::RecordProcessor Class Reference	500
G.117.1 Detailed Description	501

G.117.2	Constructor & Destructor Documentation	501
G.117.2.1	RecordProcessor()	501
G.117.3	Member Function Documentation	502
G.117.3.1	newProcessor()	502
G.117.3.2	performInitialization()	502
G.117.3.3	processRecord() [1/2]	503
G.117.3.4	processRecord() [2/2]	503
G.117.3.5	processWorkPackage()	504
G.118	BiometricEvaluation::IO::RecordStore Class Reference	504
G.118.1	Detailed Description	505
G.118.2	Member Enumeration Documentation	506
G.118.2.1	Kind	506
G.118.3	Member Function Documentation	506
G.118.3.1	begin()	506
G.118.3.2	changeDescription()	506
G.118.3.3	containsKey()	507
G.118.3.4	createRecordStore()	507
G.118.3.5	end()	508
G.118.3.6	flush()	508
G.118.3.7	getCount()	508
G.118.3.8	getDescription()	508
G.118.3.9	getPathname()	509
G.118.3.10	getSpaceUsed()	509
G.118.3.11	insert() [1/2]	509
G.118.3.12	insert() [2/2]	510
G.118.3.13	length()	510
G.118.3.14	mergeRecordStores()	511
G.118.3.15	move()	511
G.118.3.16	openRecordStore()	512
G.118.3.17	read()	512
G.118.3.18	remove()	513
G.118.3.19	removeRecordStore()	513
G.118.3.20	replace() [1/2]	514
G.118.3.21	replace() [2/2]	514
G.118.3.22	sequence()	515
G.118.3.23	sequenceKey()	515
G.118.3.24	setCursorAtKey()	516
G.118.3.25	sync()	516
G.118.4	Member Data Documentation	517
G.118.4.1	BE_RECSTORE_SEQ_NEXT	517
G.118.4.2	BE_RECSTORE_SEQ_START	517
G.118.4.3	INVALIDKEYCHARS	517
G.119	BiometricEvaluation::MPI::RecordStoreDistributor Class Reference	517
G.119.1	Detailed Description	518
G.119.2	Constructor & Destructor Documentation	518
G.119.2.1	RecordStoreDistributor()	518
G.119.3	Member Function Documentation	519
G.119.3.1	checkpointRestore()	519
G.119.3.2	checkpointSave()	519
G.119.3.3	createWorkPackage()	519
G.119.4	Member Data Documentation	519

G.119.4.1 CHECKPOINTLASTKEY	519
G.119.4.2 CHECKPOINTNUMKEYS	520
G.120 BiometricEvaluation::IO::RecordStoreIterator Class Reference	520
G.120.1 Detailed Description	520
G.120.2 Member Typedef Documentation	521
G.120.2.1 difference_type	521
G.120.2.2 iterator_category	521
G.120.2.3 pointer	521
G.120.2.4 reference	521
G.120.2.5 value_type	521
G.120.3 Constructor & Destructor Documentation	521
G.120.3.1 RecordStoreIterator() [1/4]	521
G.120.3.2 RecordStoreIterator() [2/4]	521
G.120.3.3 RecordStoreIterator() [3/4]	522
G.120.3.4 RecordStoreIterator() [4/4]	522
G.120.3.5 ~RecordStoreIterator()	522
G.120.4 Member Function Documentation	522
G.120.4.1 operator"!=()	522
G.120.4.2 operator*()	522
G.120.4.3 operator+()	523
G.120.4.4 operator++() [1/2]	523
G.120.4.5 operator++() [2/2]	523
G.120.4.6 operator+=()	523
G.120.4.7 operator->()	523
G.120.4.8 operator=()	524
G.120.4.9 operator==()	524
G.121 BiometricEvaluation::MPI::RecordStoreResources Class Reference	524
G.121.1 Detailed Description	525
G.121.2 Constructor & Destructor Documentation	525
G.121.2.1 RecordStoreResources()	525
G.121.3 Member Function Documentation	525
G.121.3.1 getOptionalProperties()	525
G.121.3.2 getRecordStore()	526
G.121.3.3 getRequiredProperties()	526
G.121.3.4 haveRecordStore()	526
G.122 BiometricEvaluation::IO::RecordStoreUnion Class Reference	526
G.122.1 Detailed Description	527
G.122.2 Constructor & Destructor Documentation	527
G.122.2.1 RecordStoreUnion() [1/7]	527
G.122.2.2 RecordStoreUnion() [2/7]	528
G.122.2.3 RecordStoreUnion() [3/7]	528
G.122.2.4 RecordStoreUnion() [4/7]	528
G.122.2.5 RecordStoreUnion() [5/7]	528
G.122.2.6 RecordStoreUnion() [6/7]	529
G.122.2.7 ~RecordStoreUnion()	529
G.122.2.8 RecordStoreUnion() [7/7]	529
G.122.3 Member Function Documentation	529
G.122.3.1 getNames()	529
G.122.3.2 getRecordStore()	530
G.122.3.3 length()	530
G.122.3.4 read()	531

G.122.3.5 setImpl()	531
G.123 BiometricEvaluation::Image::Resolution Struct Reference	531
G.123.1 Detailed Description	532
G.123.2 Member Enumeration Documentation	532
G.123.2.1 Units	532
G.123.3 Constructor & Destructor Documentation	532
G.123.3.1 Resolution()	532
G.123.4 Member Function Documentation	533
G.123.4.1 toUnits()	533
G.123.5 Member Data Documentation	533
G.123.5.1 units	533
G.123.5.2 xRes	533
G.123.5.3 yRes	533
G.124 BiometricEvaluation::MPI::Resources Class Reference	533
G.124.1 Detailed Description	534
G.124.2 Constructor & Destructor Documentation	535
G.124.2.1 Resources()	535
G.124.3 Member Function Documentation	535
G.124.3.1 getCheckpointPath()	535
G.124.3.2 getLogsheetURL()	535
G.124.3.3 getOptionalProperties()	536
G.124.3.4 getPropertiesFileName()	536
G.124.3.5 getRequiredProperties()	536
G.124.4 Member Data Documentation	536
G.124.4.1 NUMCORES	536
G.124.4.2 NUMCPUS	536
G.124.4.3 NUMSOCKETS	536
G.124.4.4 WORKERSPERNODEPROPERTY	537
G.125 BiometricEvaluation::Framework::API< T >::Result Class Reference	537
G.125.1 Detailed Description	537
G.125.2 Constructor & Destructor Documentation	537
G.125.2.1 Result()	537
G.125.3 Member Function Documentation	537
G.125.3.1 operator bool()	538
G.125.3.2 operator"!()	538
G.125.4 Member Data Documentation	538
G.125.4.1 elapsed	538
G.125.4.2 status	538
G.126 BiometricEvaluation::Feature::RidgeCountItem Struct Reference	538
G.126.1 Detailed Description	539
G.127 BiometricEvaluation::Image::ROI Struct Reference	539
G.127.1 Detailed Description	539
G.127.2 Constructor & Destructor Documentation	539
G.127.2.1 ROI() [1/2]	539
G.127.2.2 ROI() [2/2]	539
G.128 BiometricEvaluation::MPI::Runtime Class Reference	540
G.128.1 Detailed Description	540
G.128.2 Constructor & Destructor Documentation	540
G.128.2.1 Runtime()	540
G.128.3 Member Function Documentation	541
G.128.3.1 abort()	541

G.128.3.2 shutdown()	541
G.128.3.3 start()	541
G.129 BiometricEvaluation::Process::Semaphore Class Reference	541
G.129.1 Detailed Description	542
G.129.2 Constructor & Destructor Documentation	542
G.129.2.1 Semaphore() [1/2]	542
G.129.2.2 Semaphore() [2/2]	543
G.129.3 Member Function Documentation	543
G.129.3.1 getName()	543
G.129.3.2 post()	543
G.129.3.3 timedwait()	544
G.129.3.4 trywait()	544
G.129.3.5 wait()	545
G.130 BiometricEvaluation::Error::SignalManager Class Reference	545
G.130.1 Detailed Description	545
G.130.2 Constructor & Destructor Documentation	546
G.130.2.1 SignalManager() [1/2]	546
G.130.2.2 SignalManager() [2/2]	546
G.130.3 Member Function Documentation	547
G.130.3.1 clearSigHandled()	547
G.130.3.2 clearSignalSet()	547
G.130.3.3 setDefaultSignalSet()	547
G.130.3.4 setSigHandled()	547
G.130.3.5 setSignalSet()	547
G.130.3.6 sigHandled()	547
G.130.3.7 start()	548
G.130.3.8 stop()	548
G.130.4 Member Data Documentation	548
G.130.4.1 _canSigJump	548
G.130.4.2 _sigJumpBuf	548
G.131 BiometricEvaluation::Image::Size Struct Reference	549
G.131.1 Detailed Description	549
G.131.2 Constructor & Destructor Documentation	549
G.131.2.1 Size()	549
G.131.3 Member Data Documentation	549
G.131.3.1 xSize	549
G.131.3.2 ySize	549
G.132 BiometricEvaluation::Device::Smartcard Class Reference	550
G.132.1 Detailed Description	550
G.132.2 Constructor & Destructor Documentation	550
G.132.2.1 Smartcard() [1/3]	550
G.132.2.2 Smartcard() [2/3]	551
G.132.2.3 ~Smartcard()	551
G.132.2.4 Smartcard() [3/3]	551
G.132.3 Member Function Documentation	552
G.132.3.1 getDedicatedFileObject()	552
G.132.3.2 getLastAPDU()	552
G.132.3.3 getLastResponseData()	552
G.132.3.4 getReaderID()	553
G.132.3.5 operator=()	553
G.132.3.6 sendAPDU()	553

G.132.3.7 setDryrun()	553
G.133 BiometricEvaluation::IO::SQLiteRecordStore Class Reference	554
G.133.1 Detailed Description	555
G.133.2 Member Function Documentation	555
G.133.2.1 changeDescription()	555
G.133.2.2 flush()	555
G.133.2.3 getCount()	555
G.133.2.4 getDescription()	556
G.133.2.5 getPathname()	556
G.133.2.6 getSpaceUsed()	556
G.133.2.7 insert()	556
G.133.2.8 length()	557
G.133.2.9 move()	557
G.133.2.10 read()	558
G.133.2.11 remove()	558
G.133.2.12 sequence()	559
G.133.2.13 sequenceKey()	559
G.133.2.14 setCursorAtKey()	560
G.133.2.15 sync()	560
G.134 BiometricEvaluation::Process::Statistics Class Reference	560
G.134.1 Detailed Description	561
G.134.2 Constructor & Destructor Documentation	561
G.134.2.1 Statistics() [1/3]	561
G.134.2.2 Statistics() [2/3]	561
G.134.2.3 Statistics() [3/3]	562
G.134.3 Member Function Documentation	562
G.134.3.1 callStatistics_logStats()	562
G.134.3.2 getCPUTimes()	562
G.134.3.3 getMemorySizes()	563
G.134.3.4 getNumThreads()	564
G.134.3.5 logStats()	564
G.134.3.6 startAutoLogging()	564
G.134.3.7 stopAutoLogging()	565
G.135 BiometricEvaluation::Framework::Status Class Reference	565
G.135.1 Detailed Description	565
G.135.2 Constructor & Destructor Documentation	565
G.135.2.1 Status()	566
G.135.3 Member Function Documentation	566
G.135.3.1 getCode()	566
G.135.3.2 getMessage()	566
G.135.4 Member Data Documentation	566
G.135.4.1 OK	566
G.136 BiometricEvaluation::Error::StrategyError Class Reference	567
G.136.1 Detailed Description	567
G.136.2 Constructor & Destructor Documentation	567
G.136.2.1 StrategyError() [1/2]	567
G.136.2.2 StrategyError() [2/2]	567
G.137 BiometricEvaluation::Video::Stream Class Reference	567
G.137.1 Member Function Documentation	568
G.137.1.1 getFPS()	568
G.137.1.2 getFrame()	568

G.137.1.3 getFrameCount()	568
G.137.1.4 getFrameSequence()	569
G.137.1.5 setFramePixelFormat()	569
G.137.1.6 setFrameScale()	569
G.138 BiometricEvaluation::IO::SysLogsheet Class Reference	569
G.138.1 Detailed Description	571
G.138.2 Constructor & Destructor Documentation	571
G.138.2.1 SysLogsheet() [1/3]	571
G.138.2.2 SysLogsheet() [2/3]	571
G.138.2.3 ~SysLogsheet()	572
G.138.2.4 SysLogsheet() [3/3]	572
G.138.3 Member Function Documentation	572
G.138.3.1 operator=()	572
G.138.3.2 setup()	572
G.138.3.3 sync()	573
G.138.3.4 write()	573
G.138.3.5 writeComment()	573
G.138.3.6 writeDebug()	574
G.138.3.7 writeToLogger()	574
G.138.4 Member Data Documentation	574
G.138.4.1 _operational	574
G.138.4.2 _sequenced	574
G.138.4.3 _sockFD	574
G.138.4.4 _utc	575
G.139 BiometricEvaluation::MPI::TerminateJob Class Reference	575
G.139.1 Detailed Description	575
G.139.2 Constructor & Destructor Documentation	575
G.139.2.1 TerminateJob() [1/2]	575
G.139.2.2 TerminateJob() [2/2]	575
G.140 BiometricEvaluation::Image::TIFF Class Reference	576
G.140.1 Detailed Description	576
G.140.2 Member Function Documentation	576
G.140.2.1 getRawData()	576
G.140.2.2 getRawGrayscaleData()	577
G.140.2.3 isTIFF() [1/2]	577
G.140.2.4 isTIFF() [2/2]	578
G.141 BiometricEvaluation::Time::Timer Class Reference	578
G.141.1 Detailed Description	579
G.141.2 Member Typedef Documentation	579
G.141.2.1 BE_CLOCK_TYPE	579
G.141.3 Constructor & Destructor Documentation	579
G.141.3.1 Timer() [1/2]	579
G.141.3.2 Timer() [2/2]	579
G.141.4 Member Function Documentation	579
G.141.4.1 elapsed()	579
G.141.4.2 elapsedStr()	580
G.141.4.3 start()	580
G.141.4.4 stop()	581
G.141.4.5 time()	581
G.142 BiometricEvaluation::Device::TLV Class Reference	581
G.142.1 Detailed Description	582

G.142.2	Constructor & Destructor Documentation	582
G.142.2.1	TLV() [1/4]	582
G.142.2.2	TLV() [2/4]	582
G.142.2.3	TLV() [3/4]	583
G.142.2.4	TLV() [4/4]	583
G.142.3	Member Function Documentation	583
G.142.3.1	addChild()	583
G.142.3.2	getChildren()	583
G.142.3.3	getPrimitive()	584
G.142.3.4	getRawTLV()	584
G.142.3.5	getTagClass()	584
G.142.3.6	getTagNum()	584
G.142.3.7	isPrimitive()	585
G.142.3.8	setPrimitive()	585
G.142.3.9	setTag()	585
G.142.3.10	stringFromTLV()	585
G.143	BiometricEvaluation::Memory::unique_if< T > Struct Template Reference	586
G.143.1	Detailed Description	586
G.143.2	Member Typedef Documentation	586
G.143.2.1	unique_single	586
G.144	BiometricEvaluation::Memory::unique_if< T[]> Struct Template Reference	586
G.144.1	Detailed Description	586
G.144.2	Member Typedef Documentation	587
G.144.2.1	unique_array_unknown_bound	587
G.145	BiometricEvaluation::Memory::unique_if< T[S]> Struct Template Reference	587
G.145.1	Detailed Description	587
G.145.2	Member Typedef Documentation	587
G.145.2.1	unique_array_known_bound	587
G.146	BiometricEvaluation::View::View Class Reference	587
G.146.1	Detailed Description	588
G.146.2	Member Function Documentation	588
G.146.2.1	getCompressionAlgorithm()	588
G.146.2.2	getImage()	589
G.146.2.3	getImageColorDepth()	589
G.146.2.4	getImageResolution()	589
G.146.2.5	getImageSize()	589
G.146.2.6	getScanResolution()	590
G.146.2.7	setImageColorDepth()	590
G.146.2.8	setImageData()	590
G.146.2.9	setImageResolution()	590
G.146.2.10	setImageSize()	591
G.146.2.11	setScanResolution()	591
G.147	BiometricEvaluation::Time::Watchdog Class Reference	591
G.147.1	Detailed Description	592
G.147.2	Constructor & Destructor Documentation	592
G.147.2.1	Watchdog()	592
G.147.3	Member Function Documentation	593
G.147.3.1	clearCanSigJump()	593
G.147.3.2	clearExpired()	593
G.147.3.3	expired()	593
G.147.3.4	setCanSigJump()	593

G.147.3.5 setExpired()	593
G.147.3.6 setInterval()	594
G.147.3.7 start()	594
G.147.3.8 stop()	594
G.147.4 Member Data Documentation	594
G.147.4.1 PROCESSTIME	594
G.147.4.2 REALTIME	594
G.148 BiometricEvaluation::Process::Worker Class Reference	595
G.148.1 Detailed Description	596
G.148.2 Member Function Documentation	596
G.148.2.1 _initCommunication()	596
G.148.2.2 closeManagerPipeEnds()	596
G.148.2.3 closeWorkerPipeEnds()	596
G.148.2.4 getParameter()	597
G.148.2.5 getParameterAsDouble()	597
G.148.2.6 getParameterAsInteger()	597
G.148.2.7 getParameterAsString()	598
G.148.2.8 getReceivingPipe()	598
G.148.2.9 getSendingPipe()	599
G.148.2.10 receiveMessageFromManager()	599
G.148.2.11 sendMessageToManager()	599
G.148.2.12 setParameter()	600
G.148.2.13 stopRequested()	600
G.148.2.14 waitForMessage()	600
G.148.2.15 workerMain()	600
G.149 BiometricEvaluation::Process::WorkerController Class Reference	601
G.149.1 Detailed Description	602
G.149.2 Constructor & Destructor Documentation	602
G.149.2.1 WorkerController()	602
G.149.3 Member Function Documentation	602
G.149.3.1 everWorked()	602
G.149.3.2 finishedWorking()	603
G.149.3.3 getExitStatus()	603
G.149.3.4 getWorker()	603
G.149.3.5 isWorking()	603
G.149.3.6 reset()	604
G.149.3.7 sendMessageToWorker()	604
G.149.3.8 setParameter()	604
G.149.3.9 setParameterFromDouble()	605
G.149.3.10 setParameterFromInteger()	605
G.149.3.11 setParameterFromString()	605
G.149.4 Member Data Documentation	606
G.149.4.1 _rv	606
G.149.4.2 _rvSet	606
G.149.4.3 _worker	606
G.150 BiometricEvaluation::MPI::WorkPackage Class Reference	606
G.150.1 Detailed Description	606
G.150.2 Constructor & Destructor Documentation	607
G.150.2.1 WorkPackage()	607
G.150.3 Member Function Documentation	607
G.150.3.1 getNumElements()	607

G.150.3.2 getSize()	607
G.150.3.3 setData()	607
G.150.3.4 setNumElements()	608
G.151 BiometricEvaluation::MPI::WorkPackageProcessor Class Reference	608
G.151.1 Detailed Description	608
G.151.2 Member Function Documentation	609
G.151.2.1 getLogsheet()	609
G.151.2.2 newProcessor()	609
G.151.2.3 performInitialization()	609
G.151.2.4 performShutdown()	610
G.151.2.5 processWorkPackage()	610
G.151.2.6 setLogsheet()	611
G.152 BiometricEvaluation::Image::WSQ Class Reference	611
G.152.1 Detailed Description	611
G.152.2 Member Function Documentation	611
G.152.2.1 getRawData()	612
G.152.2.2 getRawGrayscaleData()	612
G.152.2.3 isWSQ()	612
G.153 BiometricEvaluation::Feature::Sort::XY Class Reference	613
G.153.1 Detailed Description	613
G.154 BiometricEvaluation::Feature::Sort::YX Class Reference	613
G.154.1 Detailed Description	613
Index	615

Chapter 1

Introduction

This document describes the Biometric Evaluation Framework (BECCommon) and application programming interfaces (API) used to support the evaluation of biometric software within the NIST Image Group [23].

When evaluating software in a “black box” fashion many aspects of program execution must be addressed, such as non-returning function calls, I/O errors, and other resource requirements. In addition, solutions to common problems should be portable across operating systems.

An evaluation consists of the testing of vendor-supplied software that implements certain biometric algorithms, such as fingerprint matching or face recognition. The NIST Image Group defines a test process and API for each evaluation. Vendors implement the API in their software, which is delivered to NIST as a software library, where common test driver is used to call the vendor library. In order to support the common functionality used across all evaluations, such as logging, file input/output, etc., a common framework is used.

Even though the Biometric Evaluation Framework was written to support biometric software evaluations, much of the framework can be used for any general purpose program where data storage and system interaction are needed. One goal of the BECommon is to reduce the low-level error processing (particularly with input and output) done directly by applications. The Biometric Evaluation Framework provides several abstractions that are useful to applications so they can focus on the task at hand.

This document describes each package and includes example code. The long form of this document includes reference sections containing auto-generated API documentation.

The BECommon is a work-in-progress, and future development will occur in areas where the need arises for the testing programs of the NIST Image Group.

Chapter 2

Overview

The Biometric Evaluation Framework (BECCommon) is a set of C++[\[29\]](#) classes, error codes, and design patterns used to create a common environment to provide logging, data management, error handling, and other functionality that is needed for many applications used in the testing of biometric software. The goals of the framework include:

- Reduce the amount of I/O error handling implemented by applications.
- Provide standard interfaces for data management and logging;
- Remove the need for applications to handle low-level events from the operating system (signals, etc.);
- Provide services for timing the execution of code blocks;
- Allow applications to constrain the amount of processing time used by a block of code;
- Reduce memory allocation errors;
- Simplify the use of parallel processing.

The experience of the NIST Image Group when running many software evaluations has led to the need of a common code for dealing with recurring software issues. One issue is the large amounts of data consumed, and created, by the software under test. Input data sets are typically biometric images, while output sets contain derived information. Both sets of data often contain millions of items, and storing each item as a file creates a tremendous burden on the file system. The `IO` package provides a solution to managing large amounts of records in a portable, efficient manner, as well as facilities for logging and maintaining runtime settings.

BECCommon is divided into several packages, each providing a set of related functionality, such as error handling and timing operations. The packages are an informal concept, mapped to formal C++ name spaces, e.g. `IO` and `Time`. A namespace contains classes, constants, and non-class functions that relate to concepts grouped in the namespace. All classes within BECCommon belong to the top-level `BiometricEvaluation` namespace.

Biometric image data is often supplied in a compressed format (e.g. WSQ, JPEG) and must be converted to a “raw” format. The `Image` package contains classes to represent compressed image data as an object, storing the image size and other attributes, in addition to the raw image.

Memory management issues are addressed by the `Memory` package. The use of classes and templates in this package can relieve applications of the need to directly manage memory for dynamically sized arrays, or call functions that are already provided to allocate and free C library objects.

While a program is running, it is often necessary to record certain statistics about the process, such as memory and processor usage. The `Process` package provides methods to obtain this information, as well as the capability to log to a file periodically, in an asynchronous manner.

In addition to its own statistics, a program may need to query some information about the environment under which it is running. The `System` package provides a count of CPUs, memory size, other system characteristics that an application can use to tailor its behavior.

Many aspects of software performance evaluation involve the use of timers. The `Time` package provides for the calculation of a time interval in a manner that is consistent across platforms, abstracting the underlying operating system’s timing facility. Also, included is a “watchdog” facility, providing a solution to the problem of non-returning function calls. By using a watchdog timer, an application can abort a call to a function that doesn’t return in the required interval.

The `Text` package provides a set of utility functions for operating on strings. The `digest` functions are of interest to those applications that must mask any information contained in a string before passing that information to another function. For example, often the biometric image file (or record) names contain information about the image, such as the finger position.

Error propagation and handling are addressed by the `Error` package. A set of exception objects are defined within this package, allowing for communication of error conditions out of the framework to the application, along with an explanatory string. Signal handling is related to error propagation in that when a process receives a signal, often it is due to software bug. Divide by zero, for example. The `Error` package provides for simple handling of the signal by the process.

Many packages in BECommon deal with biometric data record formats, including ANSI/NIST [6] records. In order to provide a general interface to several formats, BECommon represents the biometric data as derived from a source. For example, the `Finger` package contains classes that represent all information about a finger, including the source image and derived minutiae points. The `View` package combines the notions of a source image and derived information together into a single abstraction.

Applications can use the `Messaging` package to communicate between threads and processes, or to a terminal. Messages in this context are simply an array of bytes. One such use could be providing a command line interface to an long-running process.

The `MPI` package provides wrappers around the Message Passing Interface (MPI) [21] libraries, handling all MPI communication and error events. Many parallel applications can be greatly simplified, only implementing a few methods to process data.

BECommon is designed to be used in a modular fashion, and it is possible to compile many packages independently. However, several packages do make use of other packages in the framework, and therefore, are less flexible in their reuse. However, BECommon is designed to reduce the intra-framework dependencies.

A set of test programs is included with the framework. These programs not only exercise the functions provided by the packages, but also can be used as example programs on how to use framework.

The chapters that follow this overview describe each package in detail, along with some code examples. The final set of chapters of this document contain the application programming interfaces for the types, methods, and classes that make up BECommon. However, the framework is under development, and other packages, classes, etc. will be added over time to address the needs of the NIST Image Group.

Chapter 3

Framework

The `Framework` package is used to retrieve information about the Biometric Evaluation Framework itself, as well as to provide services through general purpose utility functions to other parts of the framework.

3.1 Versioning

Version numbers, the compiler used, and other framework metadata can be queried by applications. Versioning information is recorded in the `BESCommon Makefile` and populated in the function implementation at compile-time.

Listing 3.1: Using the Framework API

```
1  /* "Framework Version: 0.4" */
2  std::cout << "Framework Version: " << BE::Framework::getMajorVersion() << "." <<
3      BE::Framework::getMinorVersion() << std::endl;
4
5  /* "Compiler Used: clang v5.1.0" */
6  std::cout << "Compiler Used: " << BE::Framework::getCompiler() << " v" <<
7      BE::Framework::getCompilerVersion() << std::endl;
8
9  /* "Date/Time Compiled: Jan 24 2014 12:16:01" */
10 std::cout << "Date/Time Compiled: " << BE::Framework::getCompileDate() << " " <<
11     BE::Framework::getCompileTime() << std::endl;
```

3.2 Enumerations

As of C++ 2011, `enum s` can be strongly-typed. The Biometric Evaluation Framework makes use of these strongly-typed `enum` classes throughout. As an added convenience, functions converting to and from `enum s`, `string s`, and `int s` are defined by using a template, eliminating many lines of boiler-plate code and creating equivalence in functionality among `enum` classes throughout `BESCommon`. The output stream operator `<<` is also defined by the template.

At the core of `Framework::Enumeration` is a `const` mapping of `enum` to `string`, defined in code and instantiated at compile-time. The procedure to create a `enum`-to-`string` map is as follows:

- Include the `be_framework_enumeration.h` file to access the template definitions;
- Define the `enum` class;
- Use the `BE_FRAMEWORK_ENUMERATION_DECLARATIONS` macro to declare the `enum`-to-`string` map;

- Define the map from the enum elements to `std::string` objects;
- Use the `BE_FRAMEWORK_ENUMERATION_DEFINITIONS` macro to define the functions based on the map (`to_string`, etc.).

This procedure is demonstrated in Listing 3.2. The functions defined by the template exist within the `BiometricEvaluation::Framework::Enumeration` namespace. In the example application, the stream operator is used both with a call to the `to_string` function as well as directly. Typically the former where a stream operation is unavailable, calling a C program for example.

Listing 3.2: `Framework::Enumeration`

```

1  /*
2   * color.hpp
3   */
4  #include <be_framework_enumeration.h>
5  enum class Color
6  {
7      Black,
8      Blue,
9      Green
10 };
11 BE_FRAMEWORK_ENUMERATION_DECLARATIONS(
12     Color, Color_EnumToStringMap);
13
14 /*
15  * color.cpp
16  */
17 #include "tfr.h"
18
19 using namespace BiometricEvaluation::Framework::Enumeration;
20
21 const std::map<Color, std::string>
22 Color_EnumToStringMap = {
23     {Color::Black, "Black"},
24     {Color::Blue, "Blue"},
25     {Color::Green, "Green"}
26 };
27
28 BE_FRAMEWORK_ENUMERATION_DEFINITIONS(
29     Color,
30     Color_EnumToStringMap);
31
32 /*
33  * Application
34  */
35 #include <iostream>
36 int main()
37 {
38     std::cout << to_string(Color::Black) << std::endl;
39     std::cout << Color::Black << std::endl;
40     std::cout << to_int_type(Color::Green) << std::endl;
41     Color color = to_enum<Color>("Blue");
42     std::cout << color << std::endl;
43 }

```


While `Framework::Enumeration` was created for `BECommon`, the `template`'s only dependency is `Exception`, and so it can easily be used in other C++ 2011 projects.

Chapter 4

Memory

To assist applications with memory management, the `Memory` package provides classes to wrap C memory allocations, and other dynamically-sized objects.

4.1 AutoBuffer

The Biometric Evaluation Framework is designed to interoperate with existing C code that has its own memory management techniques, e.g. NIST Biometric Image Software [22]. In these cases, functions exist to allocate and free blocks of memory, and these calls must be made by the applications which use those libraries. To assist BECommon clients that use these existing libraries, the `AutoBuffer` class wraps the C memory management functions, guaranteeing the release of C objects when the `AutoBuffer` goes out of scope.

The `AutoBuffer` constructor takes three function pointers as parameters: one for C object construction, one for destruction, and a third, optional, function for copying the C object. If the latter is passed a `NULL`, the `AutoBuffer` and the underlying C object cannot be copied, and an exception will be thrown.

Listing 4.1 shows the use of `AutoBuffer` to wrap the memory allocation routines that are part of the NIST Biometric Image Software ANSI/NIST library.

Listing 4.1: Using the `AutoBuffer`

```
1 #include <be_memory_autobuffer.h>
2 #include <iostream>
3 extern "C" {
4     #include <an2k.h>
5 }
6
7 int
8 main(int argc, char* argv[]) {
9
10
11     /*
12      * alloc_ANSI_NIST(), free_ANSI_NIST(), and copy_ANSI_NIST()
13      * are functions in the NBIS AN2K library.
14      */
15     Memory::AutoBuffer<ANSI_NIST> an2k =
16         Memory::AutoBuffer<ANSI_NIST>(&alloc_ANSI_NIST,
17             &free_ANSI_NIST, &copy_ANSI_NIST);
18     if (read_ANSI_NIST(fp, an2k) != 0) {
19         cerr << "Could not read AN2K file." << endl;
20         return (EXIT_FAILURE);
21     }
```

```

21 |     }
22 |
23 |     for (int i = 1; i < an2k->num_records; i++) {
24 |         // process the ANSI/NIST record ...
25 |     }
26 | }

```

4.2 AutoArray

At its simplest level, `AutoArray` is a C-style array with numerous convenience methods, such as being able to query the number of elements. C++ iterators can be used over the contents of the array. The array can be resized without the need to create a new object. C++ operator overloading allows `AutoArray` objects to be passed to C-style functions that expect pointers to `AutoArray`'s template type.

`AutoArray` is used extensively in `BECommon` to help eliminate mistakes when manually allocating memory. The `AutoArray` constructor will allocate needed memory using `new` and the destructor will delete it. This ensures that any allocated memory will be appropriately freed when the `AutoArray` goes out of scope. Copy constructors and methods as well as the assignment operator all correctly manage memory so the client does not have to. Several objects in `BECommon` return `AutoArray` objects to assist clients in proper memory management.

A common use of `AutoArray` is to deal with records sequenced from a `RecordStore`. Listing 4.2 demonstrates this. Notice the omission of memory management statements – they are completely unnecessary.

Listing 4.2: Using `AutoArray`s with `RecordStore`s

```

1 | #include <be_io_dbrecstore.h>
2 | #include <be_memory_autoarray.h>
3 |
4 | #include <iostream>
5 |
6 | using namespace BiometricEvaluation;
7 |
8 | int
9 | main(
10 |     int argc,
11 |     char *argv[])
12 | {
13 |     IO::DBRecordStore rs("db_recstore", ".", IO::READONLY);
14 |
15 |     uint64_t value_size = 0;
16 |     string key("");
17 |     Memory::AutoArray<uint8_t> value;
18 |     for (bool stop = false; stop == false; ) {
19 |         try {
20 |             // Non-destructively resize the AutoArray to hold
21 |             // the next record.
22 |             value.resize(rs.sequence(key, NULL));
23 |
24 |             // Read the record into the AutoArray (treats the
25 |             // AutoArray as a pointer).
26 |             rs.read(key, value);
27 |
28 |             // Do something with value.
29 |             std::cout << "Key " << key << " has a value of " <<
30 |                 value.size() << " bytes" << std::endl;

```

```

31         } catch (Error::ObjectDoesNotExist) {
32             stop = true;
33         }
34     }
35
36     return (0);
37 }

```

AutoArray is adapted from "c_array" [29, 496].

4.3 IndexedBuffer

Many applications have a need to read items from a data record and take action based on the value of the item read. For example, when reading a biometric data record, the number of finger minutiae points in the record is indicated by a value in the record header. Furthermore, the record format may be of a different endianness than the application's host platform.

The `IndexedBuffer` class is used to access data from a buffer in fixed-size amounts in sequence. Objects of this class maintain an index into the buffer as internal state and reads out of the buffer, when using certain methods, adjust the index. In addition, standard subscript access can be done on the buffer (reads and writes) without affecting the index. The basic element type is an unsigned eight-bit value. The `IndexedBuffer` object can be created to either manage the buffer memory directly, or to "wrap" an existing buffer.

Methods to retrieve elements from the buffer are defined in the class's interface. These functions are used to retrieve 8/16/32/64-bit values while moving the internal index. Several functions are also provided to take into account the endianness of the underlying data.

Listing 4.3 shows how an application can read a data record in big-endian format.

Listing 4.3: Using the `IndexedBuffer`

```

1 #include <be_memory_autoarray.h>
2 #include <be_memory_indexedbuffer.h>
3
4 int
5 main(int argc, char* argv[]) {
6
7     uint64_t size = IO::Utility::getFileSize("BiometricRecord");
8     FILE *fp = std::fopen("BiometricRecord", "rb");
9     Memory::IndexedBuffer iBuf(size);
10    fread(iBuf, 1, size, fp);
11    fclose(fp);
12    Memory::IndexedBuffer iBuf(recordData, recordData.size());
13
14    uint32_t lval;
15    uint16_t sval;
16
17    /*
18     * Record is big-endian:
19     * -----
20     * | NAME | LENGTH | ID | ... |
21     * -----
22     *      4       4       2
23     */
24
25    /* Read a 4-byte C string */
26    lval = iBuf.scanU32Val();          /* Format ID */
27    char *cptr = (char *)&lval;

```

```
28 |         string s(cptr);
29 |
30 |         /* Read a 4-byte length */
31 |         lval = iBuf.scanBeU32Val();
32 |
33 |         /* Read a 2-byte ID */
34 |         sval = iBuf.scanBeU16Val();
35 | }
```

Chapter 5

Error Handling

Within the Biometric Evaluation Framework, Error handling has two aspects: One for communicating error conditions out of the framework and back to applications; the other for handling error signals from the environment and operating system. Classes and other code to implement error processing are described in this chapter.

5.1 Biometric Evaluation Exceptions

The Biometric Evaluation Framework contains a set of classes used to report errors to applications. Objects of these class types are thrown and contain descriptive information as to the nature of the error. Applications must handle the errors in a manner that makes sense for the application.

Applications should catch objects of the type specified in the API for the class being called. The type of object caught indicates the nature of the error that occurred, while the string stored within that object provides more information on the error.

Listing [6.2 on page 19](#) shows an example of exception handling when using the logging classes described in Section [6.3 on page 18](#).

5.2 Signal Handling

When the application process executes in a POSIX environment, signals to the process can be generated by the operating system. In many cases, if the signal is not handled by the process, execution terminates. Because the Biometric Evaluation Framework was designed to be used with software libraries for which no source code is available, changes to the code in these libraries cannot be made, and any faults in that code cannot be fixed. A common problem is that a function in the “black box” library dereferences a bad pointer, resulting in a segmentation violation signal being sent by the operating system.

To prevent termination of the application process, signal handling must be installed. The Biometric Evaluation Framework provides a class, `SignalManager`, to simplify the installation of a signal handler in order to allow the program to continue running. For example, when extracting a fingerprint minutia template from an image, often the library call will fault on a certain image. By using the `SignalManager`, the application can log that fault, and continue on to the next image.

Signal handling in a POSIX environment covers the bare essentials, and one of two actions is usually taken. The signal can be handled and processing continues at the location the signal was generated. The second action is that, in addition to signal handling, the process continues from a different location. It is the second action that is implemented by the `SignalManager` class. The rationale for this type of signal handling is so the call to the faulting function can be aborted, but the caller can detect that the signal was handled and take action, usually by logging the fault.

By default, the `SignalManager` class installs a handler for the `SIGSEGV` and `SIGBUS` signals. However, other signals can be handled as desired.

One restriction on the use of `SignalManager` is that the POSIX calls for signal management (`signal(3)`, `sigaction(2)`, etc.) cannot be invoked inside of the signal handler block.

The example in Listing 5.1 shows application use of the `SignalManager` class.

Listing 5.1: Using the `SignalManager`

```

1 #include <be_error_signal_manager.h>
2 using namespace BiometricEvaluation;
3
4 int main(int argc, char *argv[])
5 {
6     Error::SignalManager *sigmgr = new Error::SignalManager();
7
8     BEGIN_SIGNAL_BLOCK(sigmgr, sigblock1);
9     // code that may result in signal generation
10    END_SIGNAL_BLOCK(asigmgr, sigblock1);
11    if (sigmgr->sigHandled()) {
12        // log the event, etc.
13    }
14 }
```

Within the `SignalManager` header file, two macros are defined: `BEGIN_SIGNAL_BLOCK()` and `END_SIGNAL_BLOCK()`, each taking the `SignalManager` object and label as parameters. The label must be unique for each signal block. These macros insert the jump buffer into the code, which is the location where the signal handler will jump to after handling the signal. The use of these macros greatly simplifies signal handling for the application, and it is recommended that applications use these macros instead of directly invoking the methods of the `SignalManager` class, except for changing the set of handled signals.

If a signal does occur, process control jumps to the end of the signal block, and the `sigHandled()` method of the signal manager can be called. The application may need to have the same statements inside the `sigHandled()` check as those outside of the signal handling block. For example, if a file needs to be closed before the end of the block, the same call to the close function must be made within the `sigHandled()` check. Careful application design can reduce the amount of code replication, however.

Listing 5.2 shows how an application can indicate what signals to handle. In this example, only the `SIGUSR1` signal would be handled.

Listing 5.2: Specifying Signals to the `SignalManager`

```

1 #include <be_error_signal_manager.h>
2 using namespace BiometricEvaluation;
3
4 int main(int argc, char *argv[])
5 {
6     Error::SignalManager *sigmgr = new Error::SignalManager();
7
8     sigset_t sigset;
9     sigemptyset(&sigset);
10    sigaddset(&sigset, SIGUSR1);
11    sigmgr->setSignalSet(sigset);
12
13    FILE *fp = fopen( ... );
14    BEGIN_SIGNAL_BLOCK(sigmgr, sigblock2);
15    // code that may result in signal generation
16    fclose(fp);
17    END_SIGNAL_BLOCK(asigmgr, sigblock2);
```



```
18 |     if (sigmgr->sigHandled()) {  
19 |         cout << "SIGUSR1 occurred." << endl;  
20 |         fclose(fp);  
21 |     }  
22 | }
```


Chapter 6

Input/Output

The `IO` package is used by applications for the common types of input and output: managing stores of data, log files, and individual file management. The goal of using the `IO` API is to relieve applications of the need to manage low-level I/O operations such as file opening, writing, and error handling. Furthermore, by using the classes defined in `IO`, the actual storage mechanism used for data can be managed efficiently and placed in a consistent location for all applications.

Many classes manage persistent storage within the file system, taking care of file open and close operations, as well as error handling. When errors do occur, exceptions are thrown, which then must be handled by the application.

6.1 Utility

The `IO::Utility` namespace provides functions that are used to manipulate the file system and other low-level mechanisms. These functions can be used by applications in addition to being used by other classes within the Biometric Evaluation framework. The functions in this package are used to directly manipulate objects in the POSIX file system, or to check whether a file object exists.

6.2 Record Management

The `IO::RecordStore` class provides an abstraction for performing record-oriented input and output to an underlying storage system. Each implementation of the `RecordStore` provides a self-contained entity to manage data on behalf of the application in a reliable, efficient manner.

Many biometric evaluations generate thousands of files in the form of processed images and biometric templates, in addition to consuming large numbers of files as input. In many file systems, managing large numbers of files is not efficient, and leads to longer run times as well as difficulty in backing up and processing these files outside of the actual evaluation.

The `RecordStore` abstraction de-couples the application from the underlying storage, enabling the implementation of different strategies for data management. One simple strategy is to store each record into a separate file, reproducing what has typically been done in the evaluation software itself. Archive files and small databases are other implementation strategies that have been used.

Use of the `RecordStore` abstraction allows applications to switch storage strategy by changing a few lines of code. Furthermore, error handling is consistent for all strategies by the use of common exceptions.

`RecordStore`s provide no semantic meaning to the nature of the data that passes through the store. Each record is an opaque object, given to the store as a managed memory object, or pointer and data length, and is associated with a string which is the key. Keys must be unique and are associated with a single data item. Attempts to insert multiple records with the same key result in an exception being thrown.

Listing 6.1 illustrates the use of a database RecordStore within an application.

Listing 6.1: Using a RecordStore

```

1 #include <be_io_dbrecstore.h>
2 #include <be_io_utility.h>
3 using namespace BiometricEvaluation;
4 int
5 main(int argc, char* argv[]) {
6
7     std::shared_ptr<IO::RecordStore> srs;
8     try {
9         srs = IO::RecordStore::createRecordStore(
10             "myRecords", "My Record Store",
11             IO::RecordStore::Kind::BerkeleyDB);
12     } catch (Error::Exception& e) {
13         cout << "Caught " << e.whatString() << endl;
14         return (EXIT_FAILURE);
15     }
16
17     try {
18         Memory::uint8Array theData;
19         theData = getSomeData();
20         srs->insert("key1", theData);
21
22         theData = getSomeData();
23         srs->insert("key2", theData);
24
25     } catch (Error::Exception& e) {
26         cout << "Caught " << e.whatString() << endl;
27         return (EXIT_FAILURE);
28     }
29
30     // Some more processing where new data for a key comes in ...
31     theData = getSomeData();
32     srs->replace("key1", theData);
33
34     // Obtain the data for all keys and write data to a file
35     while (true) {
36         IO::RecordStore::Record record = srs->sequence();
37         cout << "Read data for key " << record.key << " of length "
38             << record.data.size() << endl;
39         IO::Utility::writeFile(record.data, record.key);
40     }
41     // The data for the key is no longer needed ...
42     srs->remove("key1");
43     return (EXIT_SUCCESS);
44 }
```

6.3 Logging

Many applications are required to log information during their processing. In particular, the evaluation test drivers often create a log record for each call to the software under test. There is a need for the log entries to be consistent, yet any logging facility must be flexible in accepting the type of data that is to be written to the log file.

The logging classes in the `IO` package provide a straight-forward method for applications to record their progress without the need to manage the low-level storage details. Management of the log messages to the backing store is done within the `Logsheet` implementations. `Logsheet` specifies the common interface to all implementations. In addition, objects of this class can be created to provide a “Null” `Logsheet` where messages are not saved.

A `Logsheet` is an output stream (subclass of `std::ostream`), and therefore can handle built-in types and any class that supports streaming. Each entry is numbered by the `Logsheet` class when written to the log. A call to the `newEntry()` method commits the current entry to the log, and resets the write position to the beginning of the entry buffer.

In addition to streaming by using the `Logsheet::<<` operator, applications can directly commit an entry to the log file by calling the `write()` method, thereby not disrupting the entry that is being formed. After an entry is committed, the entry number is automatically incremented. `Logsheet` also supports the writing of “debug” and comment entries. Each entry is prefixed with a letter code indicating the type.

6.3.1 FileLogsheet

`IO::FileLogsheet` uses a file to store the log messages. Access to this file is not controlled, and therefore, if two instances of this class are made with the same file name, the results are undefined. The description of the sheet is placed at the top of the file during construction of the object. Objects of this class can be constructed with a string containing a `file://` Uniform Resource Locator (URL) or a simple file name.

`IO::FileLogCabinet` is a container of `FileLogsheet` where each log file is contained within the same directory owned by this container class.

The example code in Listing 6.2 shows how an application can use a `FileLogsheet`, contained within a `FileLogCabinet`, to record operational information.

Listing 6.2: Using a `FileLogsheet` within a `FileLogCabinet`

```

1 #include <be_io_filelogcabinet.h>
2 using namespace BiometricEvaluation;
3 using namespace BiometricEvaluation::IO;
4
5 FileLogCabinet *lc;
6 try {
7     lc = new FileLogCabinet(lcname, "A Log Cabinet", "");
8 } catch (Error::ObjectExists &e) {
9     cout << "The Log Cabinet already exists." << endl;
10    return (-1);
11 } catch (Error::StrategyError& e) {
12    cout << "Caught " << e.whatString() << endl;
13    return (-1);
14 }
15 std::unique_ptr<FileLogCabinet> ulc(lc);
16 try {
17     ls = alc->newLogsheet("log01", "Log Sheet in Cabinet");
18 } catch (Error::ObjectExists &e) {
19     cout << "The log sheet already exists." << endl;
20     return (-1);
21 } catch (Error::StrategyError& e) {
22     cout << "Caught " << e.whatString() << endl;
23     return (-1);
24 }
25 ls->setAutoSync(true); // Force write of every entry when finished
26 int i = ...
27 *ls << "Adding an integer value " << i << " to the log." << endl;

```

```

28 ls->newEntry();           // Forces the write of the current entry
29 .....
30 delete ls;
31 return;                  // The LogCabinet is destructed by the unique_ptr

```

6.3.2 SysLogsheet

The `SysLogsheet` is an implementation of `Logsheet` which writes log entries to a system logger service. Objects of this class are created with a URL starting with `syslog://`. When using a system logger, the URL must give the hostname of the logger as well as the network port: `syslog://node00:4315` for example. The system logger must understand the Syslog protocol as specified in RFC5424 [30].

Multiple instances of a `SysLogsheet` can be created with the same URL with the assumption that the logging server can manage multiple incoming message streams.

6.4 Properties

The `Properties` class is used to store simple key-value string pairs, with the option to save to a file. Applications can use a `Properties` object to manage runtime settings that are persistent across invocations, or to simply store some settings in memory only.

Listing 6.3: Using a `Properties` Object

```

1 IO::Properties *props;
2 string fname = "test.prop";
3 try {
4     props = new IO::Properties(fname);
5 } catch (Error::StrategyError &e) {
6     cerr << "Caught " << e.whatString() << endl;
7     return;
8 } catch (Error::FileError& e) {
9     cerr << "A file error occurred: " << e.whatString() << endl;
10    return;
11 }
12 props->setProperty("foo", "bar");
13 props->setProperty("theAnswer", "42");
14 :
15 :
16 :
17 try {
18     int64_t theAnswer = props->getProperty("theAnswer");
19     cout << "The answer is " << theAnswer << endl;
20 } catch (Error::ObjectDoesNotExist &e) {
21     cerr << "The answer is elusive." << endl;
22     return;
23 }
24 string fooProp = props->getProperty("foo");
25 cout << "Foo is set to " << fooProp << endl;
26 :
27 :
28 :
29 try {
30     props->removeProperty("foo");
31 } catch (Error::ObjectDoesNotExist &e) {
32     cerr << "Failed to remove property." << endl;

```

33 | }

6.5 Compressor

Support for data compression and decompression can be found in the Biometric Evaluation Framework through the Compressor class hierarchy. Compressor is an abstract base class defining several pure-virtual methods for compression and decompression of buffers and files. Derived classes implement these methods and can be instantiated through the factory method in the base class. As such, children should also be enumerated within Compressor::Kind. The Biometric Evaluation Framework comes with an example, GZIP, which compresses and decompresses the gzip format through interaction with zlib [8].

Listing 6.4: Using a Compressor Object

```

1 shared_ptr<IO::Compressor> compressor;
2 Memory::uint8Array compressedBuffer, largeBuffer = /* ... */;
3 try {
4     compressor = IO::Compressor::createCompressor(Compressor::Kind::GZIP);
5     /* Overloaded for all combination of buffer and file */
6     compressor->compress("largeInputFile", "compressedOutputFile");
7     compressor->compress(largeBuffer, compressedBuffer);
8 } catch (Error::Exception &e) {
9     cerr << "Could not compress (" << e.whatString() << ')' << endl;
10 }
```

Different Compressors may be able to respond to options that tune their operations. These options (and approved values) should be well-documented in the child class, however, a no-argument constructor of a child Compressor should automatically set any required options to default values. Setting and retrieving these options is very similar to interacting with a Properties object (see Section 6.4 on the facing page).

Listing 6.5: Setting Compressor Options

```

1 shared_ptr<IO::Compressor> compressor =
2     IO::Compressor::createCompressor(Compressor::Kind::GZIP);
3
4 /* A large GZIP chunk size can speed operations on systems with copious RAM */
5 compressor->setOption(IO::GZIP::CHUNK_SIZE, 32768);
```


Chapter 7

Text

The `Text` package consists of functions to perform common operations on `strings` and `char` arrays. Many of the operations may be considered “trivial,” but are used often enough within the Biometric Evaluation Framework and other applications that a common implementation in `BECommon` is more than warranted. A complete listing of functions is available in the documentation appendix for `BiometricEvaluation::Text2`.

Listing 7.1 shows how to use the `split()` function from the `Text` package. `split()` can separate a `string` into tokens delimited by a character, useful for processing comma- or space-separated text files (such files could be produced by a `LogSheet` (Section 6.3 on page 18), for instance). Here, a text file containing metadata for an image is being parsed, perhaps to be passed to the `RawImage` constructor (Section 11.3 on page 38).

Listing 7.1: Tokenizing a string

```
1  /* Definition of input strings */
2  static const vector<string>::size_type filenameToken = 0;
3  static const vector<string>::size_type widthToken = 1;
4  static const vector<string>::size_type heightToken = 2;
5  static const vector<string>::size_type depthToken = 3;
6
7  /* Split the string, presumably input from a file */
8  string input = "/mnt/raw\\ images/1.raw 500 500 8";
9  vector<string> tokens = Text::split(input, ' ', true);
10
11 /* Assign the retrieved tokens */
12 string filename;
13 uint32_t width, height, depth;
14 try {
15     filename = tokens.at(filenameToken);    /* "/mnt/raw images/1.raw" */
16     width = atoi(tokens.at(widthToken).c_str());    /* "500" */
17     height = atoi(tokens.at(heightToken).c_str()); /* "500" */
18     depth = atoi(tokens.at(depthToken).c_str());    /* "8" */
19 } catch (out_of_range) {
20     throw Error::FileError("Malformed input");
21 }
```

Notice the `true` parameter to `split()` in Listing 7.1. This instructs `split()` to not tokenize based on an escaped delimiter. If `false`, the first token would be split into two at the presence of the delimiter.

`Text` also contains functions to perform hashing via `OpenSSL`. A two-line program that emulates the command-line `md5sum` program is shown in Listing 7.2. Changing the digest parameter to `"sha1"` would make the program emulate `'openssl sha1'`.

Listing 7.2: md5sum via BECommon

```
1 #include <cstdlib>
2 #include <iostream>
3
4 #include <be_io_utility.h>
5 #include <be_text.h>
6 #include <be_memory_autoarray.h>
7
8 using namespace std;
9 using namespace BiometricEvaluation;
10
11 int
12 main(
13     int argc,
14     char *argv[])
15 {
16     if (argc == 0)
17         return (EXIT_FAILURE);
18
19     try {
20         Memory::uint8Array file = IO::Utility::readFile(argv[1]);
21         cout << Text::digest(file, file.size(), "md5") << " " <<
22             argv[1] << endl;
23     } catch (Error::Exception) {
24         return (EXIT_FAILURE);
25     }
26
27     return (EXIT_SUCCESS);
28 }
```

Chapter 8

Time and Timing

The `Time` package within the Biometric Evaluation Framework provides a set of classes for performing timing-related operations, such as elapsed time and limiting execution time.

8.1 Elapsed Time

The `Timer` class provides applications a method to determine how long a block of code takes to execute. On many systems (e.g. Linux) the timer resolution is in microseconds.

Listing 8.1 shows how an application can use a `Timer` object to limit obtain the amount of time used for the execution of a block of code.

Listing 8.1: Using the `Timer`

```
1 #include <be_time_timer.h>
2
3 int main(int argc, char *argv[])
4 {
5     Time::Timer timer = new Time::Timer();
6
7     try {
8         atimer->start();
9         // do something useful, or not
10        atimer->stop();
11        cout << "Elapsed time: " << atimer->elapsed() << endl;
12    } catch (Error::StrategyError &e) {
13        cout << "Failed to create timer." << endl;
14    }
15 }
```

8.2 Limiting Execution Time

The `Watchdog` class allows applications to control the amount of time that a block of code has to execute. The time can be *real* (i.e. “wall”) time, or *process* time (not available on Windows). One typical usage for a `Watchdog` timer is when a call is made to a function that may never return, due to problems processing an input biometric image.

`Watchdog` timers can be used in conjunction with `SignalManager` in order to both limit the processing time of a call, and handle all signals generated as a result of that call. See 5.2 for information on the `SignalManager` class.

One restriction on the use of Watchdog is that the POSIX calls for signal management (`signal(3)`, `sigaction(2)`, etc.) cannot be invoked inside of the WATCHDOG block. This restriction includes calls to `sleep(3)` because it is based on signal handling as well.

Listing 8.2 shows how an application can use a Watchdog object to limit the amount of process time for a block of code.

Listing 8.2: Using the Watchdog

```

1 #include <be_time_watchdog.h>
2 int main(int argc, char *argv[])
3
4     Time::Watchdog theDog = new Time::Watchdog(Time::Watchdog::PROCESSTIME);
5     theDog->setInterval(300);    // 300 microseconds
6
7     Time::Timer timer;
8
9     BEGIN_WATCHDOG_BLOCK(theDog, watchdogblock1);
10        timer.start();
11        // Do something that may take more than 300 usecs
12        timer.stop();
13        cout << "Total time was " << timer.elapsed() << endl;
14    END_WATCHDOG_BLOCK(theDog, watchdogblock1);
15    if (theDog->expired()) {
16        timer.stop();
17        cerr << "That took too long." << endl;
18    }
19 {
20 }
```

Within the Watchdog header file, two macros are defined: `BEGIN_WATCHDOG_BLOCK()` and `END_WATCHDOG_BLOCK()`, each taking the Watchdog object and label as parameters. The label must be unique for each WATCHDOG block. The use of these macros greatly simplifies Watchdog timers for the application, and it is recommended that applications use these macros instead of directly invoking the methods of the Watchdog class, except for setting the timeout value.

Any processing that is normally done at the end of the WATCHDOG block must also be done within the `expired()` check due to the fact that process control jumps to the end of the WATCHDOG block in the event of a timeout. A typical example is the use of the Timer object inside a WATCHDOG block, as the example in Listing 8.2 shows. In most cases, however, careful application design can remove the need for duplicate code. In the example, placing the Timer `start()/stop()` calls outside of the WATCHDOG block simplifies the coding, although the small amount of time for the WATCHDOG setup and tear down would be included in the time.

Chapter 9

Process Information and Control

The `Process` package is a set of APIs used to gather information on a process, limit the capabilities of a process, and to manage the life cycle of processes.

9.1 Process Statistics

When an application is running, there may be a need to obtain information of the process executing that application. The `Process` can be used by the application itself to gather statistics related to the current amount of memory being used, the number of threads, and other items. Biometric evaluation test drivers are linked against a third party library, and therefore, the application writer does not control the thread count or memory usage for much of the processing. Listing 9.1 shows how an application can use the `Statistics` API.

Listing 9.1: Gathering Process Statistics

```
1 #include <be_error_exception.h>
2 #include <be_process_statistics.h>
3 using namespace BiometricEvaluation;
4
5 int main(int argc, char *argv[])
6 {
7     Process::Statistics stats;
8     uint64_t userstart, userend;
9     uint64_t systemstart, systemend;
10    uint64_t diff;
11    try {
12        stats.getCPUTimes(&userstart, &systemstart);
13
14        // Do some long processing....
15
16        stats.getCPUTimes(&userend, &systemend);
17        diff = userend - userstart;
18        cout << "User time elapsed is " << diff << endl;
19        diff = systemend - systemstart;
20        cout << "System time elapsed is " << diff << endl;
21    } catch (Error::Exception) {
22        cout << "Caught " << e.getInfo() << endl;
23    }
24
25 }
```

In addition to using the `Process` API to gather statistics to be returned from the function call, the API provides a means to have a “standard” set of statistics logged either synchronously or asynchronously to a `LogSheet` (See Section 6.3 on page 18) contained within a `LogCabinet`. Applications can start and stop logging at will to this `LogSheet`. Post-mortem analysis can then be done on the entries in the log. Listing 9.2 shows the use of logging.

The `LogSheet` will have a file name constructed from the process name (i.e. the application executable) and the process ID. An example `LogSheet` contains this information at the start:

```
Description: Statistics for test_be_process_statistics (PID 28370)
# Entry Ustime Systeime RSS VMSize VMPeak VMData VMStack Threads
E0000000001 728889 6998 1788 57472 62612 31020 84 1
E0000000002 1300802 6998 1792 57472 62612 31020 84 1
```

The `Statistics` object creates the `LogSheet` with an appropriate description and comment entry with column headers. Each gathering of the statistics results in a single log entry.

Listing 9.2: Logging Process Statistics

```
1 #include <be_error_exception.h>
2 #include <be_io_logcabinet.h>
3 #include <be_process_statistics.h>
4 using namespace BiometricEvaluation;
5
6 int main(int argc, char *argv[])
7 {
8     IO::LogCabinet lc("statLogCabinet", "Cabinet for Statistics", "");
9
10    Process::Statistics *logstats;
11    try {
12        logstats = new Process::Statistics(&lc);
13    } catch (Error::Exception &e) {
14        cout << "Caught " << e.getInfo() << endl;
15        return (EXIT_FAILURE);
16    }
17    try {
18        while (some_processing_to_do) {
19            // Do the work
20            // Synchronously log after the work is done.
21            logstats->logStats();
22        }
23    } catch (Error::Exception &e) {
24        cout << "Caught " << e.getInfo() << endl;
25        delete logstats;
26        return (EXIT_FAILURE);
27    }
28
29    // Set up asynchronous logging, every second
30    try {
31        logstats->startAutoLogging(1);
32    } catch (Error::ObjectExists &e) {
33        cout << "Caught " << e.getInfo() << endl;
34        delete logstats;
35        return (EXIT_FAILURE);
36    }
37
38    // Do some other work
```

```

39 |
40 |     // Stop logging
41 |     logstats->stopAutoLogging();
42 |     delete logstats;
43 | }

```

9.2 Process Management

During a biometric evaluation or other long-running CPU-bound task, it's beneficial to make efficient use of all the hardware available on the system. Applications can take advantage of a multi-core machine, for example. BECommon aims to simplify this by abstracting the usage of process and thread creation to run multiple instances of the same function in parallel.

9.2.1 Manager

There are three class hierarchies involved in the abstraction. The `BiometricEvaluation::Process::Manager` classes control the technique of process manipulation that will be used. BECommon provides two example abstractions: `ForkManager` and `POSIXThreadManager`. When using `ForkManager`, new processes will be created with `fork(2)`, with mediated access to these new processes through the `Manager`. Likewise, `POSIXThreadManager` creates new POSIX threads. Because both of these classes inherit from `Manager`, it is as trivial as changing the `Manager` object type to change how the workload is parallelized.

9.2.2 Worker

In the application using a `Manager`, a `Worker` subclass must be implemented. An example `Worker` is shown in Listing 9.3. The entry-point for a `Worker` is the `workerMain()` method, which must be implemented by the client application. Although `workerMain()` takes no arguments, data may be transmitted into the object through `WorkerController's` (9.2.3) `setParameter()` method. Within the `Worker` instance, the parameters are then retrieved with `getParameter()` when provided with the unique parameter name.

A responsible worker performs its operations as fast as it can. However, at any given time, the manager may ask the worker to stop. It then becomes the *responsibility of the worker* to stop as soon as possible. The `Worker` is notified of the stop request through its `stopRequested()` method. Note that the manager does **not** force the worker to stop, though prolonged work or cleanup in the worker would likely produce undesired results in the client application. As such, a responsible worker checkpoints itself to prepare for premature stops requested by the manager. While it is important for a worker to stop as soon as possible after the request is received, it is also important not to leave work in an unsynchronized state. In Listing 9.3, notice how the `Employee` must continue the interaction with the `Customer` before a stop request is handled, even if the `Employee's` shift has ended. Leaving the method before the `Customer's` order has been delivered would leave the `Customer` object in an unsafe state (hungry).

Listing 9.3: A Responsible Worker Implementation

```

1 | #include <cstdlib>
2 | #include <tr1/memory>
3 | #include <queue>
4 |
5 | #include <restaurant.h>
6 |
7 | #include <be_process_forkmanager.h>
8 |
9 | using namespace std;
10 | using namespace BiometricEvaluation;

```

```

11 using namespace Restaurant;
12
13 class ResponsibleEmployeeTask : public Process::Worker
14 {
15 public:
16     int32_t
17     workerMain()
18     {
19         int32_t status = EXIT_FAILURE;
20
21         /* Retrieve objects assigned to this Task */
22         tr1::shared_ptr<Employee> employee =
23             tr1::static_pointer_cast<Employee>(
24                 this->getParameter("employee"));
25         tr1::shared_ptr< queue<Customer*> > customers =
26             tr1::static_pointer_cast< queue<Customer*> >(
27                 this->getParameter("customers"))
28
29         employee->clockIn();
30
31         Customer *customer;
32         /* Checkpoint after each customer */
33         while (this->stopRequested() == false ||
34             employee->isShiftOver() == false) {
35             customer = customers->front();
36
37             if (customer != NULL) {
38                 employee->takeOrder(customer);
39                 employee->cookFood(customer);
40                 employee->deliverOrder(customer);
41
42                 customers->pop();
43             }
44         }
45
46         employee->settleCashDrawer();
47         employee->clockOut();
48
49         status = EXIT_SUCCESS;
50         return (status);
51     }
52     ~ResponsibleEmployeeTask() {}
53 };

```

After a manager starts its workers, the manager has the option of waiting until all `Worker`s exit `workerMain()` before continuing code execution. If not waiting, there are several methods the manager can perform to keep track of the status of the workers. Even if not waiting for workers to return, a responsible manager will wait a reasonable amount of time for workers to return before application termination. An example of this reasonable waiting period can be seen in [Listing 9.4 on the facing page](#).

9.2.3 WorkerController

The final piece of the process management puzzle is the `WorkerController` hierarchy. This class decorates and mediates communication between the `Manager` and the `Worker`. `WorkerController` objects may only be instantiated by a `Manager` object. All communications to the `Worker` (e.g. `isWorking()`) should be delegated through the `WorkerController`. If defining a new `Manager`, note that the `Worker`

Controller may seem unnecessary for the parallelization technique being employed. It's true that some parallelization techniques may not require this "middle-man" approach, but others do. Do not be concerned if a `WorkerController` implementation ends up being nothing more than a "pass-thru" to the `Worker`.

Listing 9.4 is a continuation of Listing 9.3 on page 29 demonstrating the use of `Manager`s and `WorkerController`s.

Listing 9.4: Using `Manager`s and `WorkerController`s

```

1 int
2 main(
3     int argc,
4     char *argv[])
5 {
6     static const uint32_t numEmployees = 3;
7     int status = EXIT_FAILURE;
8
9     tr1::shared_ptr<Process::Manager> shiftLeader(new Process::ForkManager);
10    queue<Customer*> *customers = new queue<Customer*>();
11
12    /* Create Employees (Workers/WorkerControllers) */
13    tr1::shared_ptr<Process::WorkerController> employees[numEmployees];
14    for (uint32_t i = 0; i < numEmployees; i++) {
15        employees[i] = shiftLeader->addWorker(
16            tr1::shared_ptr<ResponsibleEmployeeTask>(
17                new ResponsibleEmployeeTask()));
18
19        /* Assign employees to each Task */
20        employees[i]->setParameter("employee",
21            tr1::shared_ptr<Employee>(new Employee()));
22        employees[i]->setParameter("customers",
23            tr1::shared_ptr<queue<Customer*>>(customers));
24    }
25
26    /* Employees start serving customers while shift leader manages */
27    shiftLeader->startWorkers(false);
28
29    /* Customers enter the queue... */
30    queue<Restaurant::AdministrativeTasks> adminTasks;
31    adminTasks.push("Inventory");
32    adminTasks.push("Customer Complaints");
33    adminTasks.push("Clean Dining Room");
34
35    while (shiftLeader->getNumActiveWorkers() != 0) {
36        shiftLeader->doTask(adminTasks.front());
37        adminTasks.pop();
38    }
39
40    /* ...end of the day */
41    for (uint32_t i = 0; i < numEmployees; i++)
42        if (employees[i]->isWorking())
43            shiftLeader->stopWorker(employees[i]);
44
45    /*
46     * Wait a reasonable amount of time before locking up for the night
47     * (in this case, indefinitely).
48     */

```

```

49     while (shiftLeader->getNumActiveWorkers() > 0)
50         sleep(1);
51
52     shiftLeader->armAlarmAndExit();
53
54     status = EXIT_SUCCESS;
55     return (status);
56 }

```

9.2.4 Communications

Managers and workers may have a good reason to send and receive messages directly. A communications mechanism is built-in to the [Process Management](#) model to facilitate such communications. The type and content of the message is completely up to the client implementation, since messages are sent as `AutoArray s`. A manager does not directly send messages to a worker. This service is provided by the `WorkerController` (via `sendMessageToWorker()`).

Managers can keep an eye on incoming messages by calling the (optionally blocking) `waitForMessage()` method. This method will return a handle to the worker that sent a message. Alternatively, the manager can invoke `getNextMessage()` (again, blocking optional) to immediately receive the next message.

Listing 9.5 and Listing 9.6 are continuations of Listing 9.3 on page 29 and Listing 9.4 on the preceding page respectively, showing an example of communication, using `std::string` messages.

Listing 9.5: Worker Communication

```

1     Memory::uint8Array msg;
2
3     /* Deal with next customer unless Manager interrupts in next second */
4     if (this->waitForMessage(1)) {
5         if (this->receiveMessageFromManager(msg)) {
6             Action action = Restaurant::messageToAction(msg);
7             switch (action) {
8                 case TAKE_BREAK:
9                     employee->goOnBreak();
10                    break;
11                    /* ... */
12                }
13            }
14        }
15
16        /* ... */
17
18        if (customer->isComplaining()) {
19            sprintf((char *)&(*msg), "Customer Complaint");
20            this->sendMessageToManager(msg);
21        }

```

Listing 9.6: Manager Communication

```

1     tr1::shared_ptr<Process::WorkerController> sender;
2     Memory::uint8Array msg;
3
4     /* Do routine tasks unless employee has concern in the next 2 seconds */
5     while (this->getNextMessage(sender, msg, 2)) {
6         Action action = Restaurant::messageToAction(msg);
7         switch (action) {

```

```
8         case CUSTOMER_COMPLAINT:
9             sprintf((char *)&(*msg), "I'll take care of it.");
10            this->sendMessageToWorker(msg);
11            break;
12        /* ... */
13    }
14
15
16    /* ... */
17
18    /* Closing Time */
19    sprintf((char *)&(*msg), "Clock out and go home.");
20    this->broadcastMessage(msg);
```


Chapter 10

System

The `System` package provides a set of functions in the that return information about the hardware and operating system. This information can be used by applications to determine the amount of real memory, number of central processing units, or current load average. This information can be used to dynamically tailor the application behavior, or simply to provide additional information in a runtime log.

Listing 10.1 shows how an application can spawn several child processes based on the number of CPUs and memory available. Note that this information may not be available on all platforms, and therefore, the application must be prepared to handle that situation.

Listing 10.1: Using the `System` CPU Count Information

```
1 #include <iostream>
2 #include <be_system.h>
3
4 using namespace BiometricEvaluation;
5
6 int
7 main(int argc, char* argv[]) {
8
9     // perform some application setup ...
10
11     uint32_t cpuCount;
12     uint64_t memSize, vmSize;
13     try {
14         cpuCount = System::getCPUCount();
15         cpuCount--; // subtract one CPU for the parent process
16         memSize = System::getRealMemorySize();
17         Process::Statistics::getMemorySizes(NULL, &vmSize, NULL, NULL, NULL);
18         memSize -= vmSize; // subtract off memory used by parent
19
20         // Give each child a fraction of the memory
21         spawnChildren(cpuCount, memSize / cpuCount);
22     } catch (Error::NotImplemented) {
23         cout << "Running a single process only." << endl;
24     }
25
26     // processing done by parent ...
27 }
```


Chapter 11

Image

The `Image` package maintains the classes and other information related to images and image processing. Within the Biometric Evaluation Framework, many classes refer to images, such as when dealing with fingerprint data. Many biometric data standards supply the actual image encoded in one of several standard formats. Applications can retrieve the image as stored in the record, or decompressed by the `Image` class into a “raw” format. Therefore, within the `BECommon`, several of the common compression formats are supported, removing the need for applications to decompress the image directly, while maintaining access to the as-recorded image format.

11.1 The Image Namespace

The `Image` namespace contains several data types used to represent aspects of an image. The types defined are chiefly used to retrieve common information from images stored in an `Image` class (section 11.2). Data types in the `Image` namespace do not perform any translation of scale units or sizing, as each set of attributes is copied directly from the image data itself when possible.

The same applies to images encapsulated in biometric records. Although some biometric records have fields for image attributes like dimensions and resolution, the corresponding fields of an `Image` class are **not** populated with their contents. The `Image` namespace data types *are* used outside of the namespace, such as in finger views, to retrieve image attributes stored as part of the biometric record. Applications can compare those values against the values within the `Image` object, as in most cases those values are taken directly from the underlying image data. See Chapter 15 on page 51 for more information on image-based biometric records.

The `Image` namespace contains all of the `Image` classes that are used to represent an image. These classes are described in the following sections.

11.2 The Image Class

The `Image` class is an abstract base class that defines a set of minimum functionality for all supported image formats. Once an `Image` has been constructed, it may not be modified. For any supported image format, the following information is required to be accessible:

- Original binary data
- Compression algorithm
- Decompressed (“raw”) format binary data (grayscale, full color)
- Depth

- Dimensions (width, height)
- Resolution (horizontal, vertical)

A rudimentary implementation of generating a grayscale image is provided by the `Image` class in `getRawGrayscaleData()`. This implementation calculates the luminance value Y (of YCbCr) for each pixel of a color image. The resulting image always uses 8-bits to represent a pixel, but can return a raw image using 2 gray levels (1-bit) or 256 gray levels (8-bit). The 1-bit algorithm quantizes to black when the 8-bit color value is ≤ 127 . `Image` subclasses may override and implement their own grayscale conversion methods.

Also of interest in the `Image` class is `valueInColorspace()`, a static function to convert color values between bit depths.

11.3 Raw Image

The `RawImage` class represents a decompressed image, or an image where `getRawData()` would return the exact same data as `getData()`. `RawImage` has no special implementation or additional methods.

11.4 JPEG

The `JPEG` class represents an image encoded according to the JPEG image standard [16]. Decompression and grayscale conversion are accomplished via `libjpeg` [14].

As of version 8.0, `libjpeg` provided a way to handle JPEG images existing within in-memory buffers, as opposed to on-disk files. Because the `Image` class requires in-memory buffers, `JPEG` includes a JPEG memory source manager implementation, but it is built only if a version of `libjpeg` older than 8.0 is detected at compile-time.

`JPEG` provides a static function to determine whether or not a data buffer appears to be encoded in the JPEG image standard format. Errors within `libjpeg` will be caught and rethrown as `Exceptions`.

11.5 JPEGL

Similar to `JPEG`, the `JPEGL` class performs `Image` class services for lossless JPEG encoded images. `JPEGL` decompression is performed by NIST Biometric Image Software's `libjpegl` [22].

11.6 JPEG2000

The `JPEG2000` class provides `Image` class functionality to JPEG 2000-encoded images [15]. The class makes an attempt to support the following JPEG 2000 codecs:

- JPEG 2000 codestream (.j2k)
- JPEG 2000 compressed image data (.jp2)
- JPEG 2000 interactive protocol (.jpt)

Decompression is provided by the OpenJPEG library (`libopenjpeg`) [19]. `JPEG2000` also provides a static function to test whether or not an image appears to be JPEG 2000-encoded.

Not all information required by the `Image` class is present in a JPEG 2000-encoded image. In particular, some codecs and encoders omit the “Display Resolution Box.” It is generally accepted that the resolution will be 72 pixels-per-inch when the “Display Resolution Box” is not present.

Errors within `libopenjpeg` will be caught and rethrown as `Exceptions`.

11.7 NetPBM

The `NetPBM` class provides `Image` class functionality to all types of NetPBM formatted images, up to 48-bit depth. This includes the following formats:

- ASCII Portable Bitmap (P1, .pbm)
- ASCII Portable Graymap (P2, .pgm)
- ASCII Portable Pixmap (P3, .ppm)
- Binary Portable Bitmap (P4, .pbm)
- Binary Portable Graymap (P5, .pgm)
- Binary Portable Pixmap (P6, .ppm)

`NetPBM` provides some of its more general use parsing algorithms as static functions for use outside of the class. This includes ASCII to binary pixel conversion. A function to test for NetPBM formats is also provided.

11.8 PNG

The `PNG` class represents an image encoded according to the PNG image standard [11]. Decompression is provided by `libpng` [26].

PNG provides a static function to test whether or not an image appears to be encoded in the PNG image standard format. Errors within `libpng` are caught and rethrown as `Exceptions`.

11.9 TIFF

The `TIFF` provides the ability to decompress many TIFF-encoded images. Decompression routines are provided by `libtiff` [27]. Like most other `Image` classes, only basic grayscale and RGB-based images are parsable. The `TIFF` class will throw a `NotImplemented` exception in the event that unsupported TIFF data is provided.

11.10 WSQ

Images encoded in the WSQ-image standard [31] are represented by the `WSQ` class. The WSQ decompressor found in NIST Biometric Image Software [22], `libwsq`, is used by this class. The class provides a static function to determine whether or not an image appears to be encoded in the WSQ format.

Errors from the `libwsq` will be displayed through `stderr` and will **not** be thrown as exceptions.

11.11 BMP

The bitmap image file format [20] is decoded by the `BMP` class. Only images with the 40-byte `BITMAPINFOHEADER`, uncompressed or RLE8 compression are supported. The bits-per-pixel value can be 8, 24, or 32.

Chapter 12

Video

The `Video` package is used to access video (and, in the future, audio) streams from containers in several formats, such as MPEG4. The classes in this package rely on the FFmpeg [12] libraries to de-multiplex video streams from a container, and to decode the streams and retrieve the frames from the video.

12.1 Container

`Container` objects can be instantiated in three ways:

1. With a filename: Memory usage will equal to the size of the container stream;
2. With a `AutoArray::uint8Array`: Memory usage will be twice that of the size of the container stream;
3. With a `std::shared_ptr` wrapping a `AutoArray::uint8Array`: Memory usage equal to the size of the container stream. Applications must not modify the container data.

By careful coding, the application can prevent duplicate copies of the container buffer when using method three. By taking advantage of C++ 2011 move semantics, `BECommon` and the application avoid duplicate copies. See Listing 12.1 for examples of using all three methods.

12.2 Stream

`Stream` objects represent a single video stream within the container and provide access to individual frames from the video stream. In addition, these frames can be retrieved at their native size, or can be scaled to a different size. Frames can be returned as 24-bit red/green/blue images, grayscale, or two-color monochrome.

`Stream` objects can be obtained only from a `Container` object. The reason for this is that video frames must be pulled from a stream that is de-multiplexed from the container stream shared with the `Container` object. Future versions of `BECommon` may allow for `Streams` to be directly instantiated with coded video streams.

Listing 12.1 shows the use of `Container` and `Stream`.

Listing 12.1: Using the Video Framework

```
1 #include <iostream>
2 #include <be_memory_autoarray.h>
3 #include <be_io_utility.h>
4 #include <be_video_container.h>
5
```

```

6 using namespace BiometricEvaluation;
7 using namespace std;
8
9 int
10 main(int argc, char* argv[])
11 {
12     std::unique_ptr<Video::Container> pvc;
13
14     std::string filename = "./test_data/2video1audio.mp4";
15     if ((argc != 1) && (argc != 2)) {
16         cerr << "usage: " << argv[0] << " [filename]" << endl
17             << "If <filename> is not given, " << filename
18             << " is used instead." << endl;
19         return (EXIT_FAILURE);
20     }
21     if (argc == 2)
22         filename = argv[1];
23
24     cout << "Construct an program stream from file "
25          << filename << endl;
26     /*
27      * Three ways to open the container:
28      * 1) Have the framework open the file directly;
29      * 2) Read the file into a local buffer and give that to the framework;
30      * 3) Read the file into a buffer wrapped in a shared pointer and pass
31      *    that to the framework.
32      */
33     try {
34         //         pvc.reset(new
35         //             Video::Container(filename));
36
37         //         Memory::uint8Array buf =
38         //             IO::Utility::readFile(filename);
39         //         pvc.reset(new Video::Container(buf));
40
41         std::shared_ptr<Memory::uint8Array> buf;
42         buf.reset(new Memory::uint8Array(
43             IO::Utility::readFile(filename)));
44         pvc.reset(new Video::Container(buf));
45     } catch (Error::Exception &e) {
46         cout << "Caught: " << e.whatString() << endl;
47         return (EXIT_FAILURE);
48     }
49
50     cout << "Video Count: " << pvc->getVideoCount() << endl;
51
52     std::unique_ptr<Video::Stream> stream;
53     /*
54      * Open the first video stream.
55      */
56     try {
57         stream = pvc->getVideoStream(1);
58     } catch (Error::Exception &e) {
59         cerr << "Could not retrieve video stream: " << e.whatString()
60             << endl;
61         return (EXIT_FAILURE);

```

```

62     }
63     /*
64     * Read all the frames, one at a time, scaled down and converted
65     * to 8-bit grayscale.
66     */
67     float scaleFactor = 0.5;
68     Image::PixelFormat pixelFormat = Image::PixelFormat::Gray8;
69     stream->setFrameScale(scaleFactor, scaleFactor);
70     stream->setFramePixelFormat(pixelFormat);
71     uint64_t expectedCount = stream->getFrameCount();
72
73     cout << "First video stream: " << stream->getFPS() << " FPS, "
74          << expectedCount << " frames." << endl;
75     /*
76     * The frame count can be zero, meaning unknown. If that is the case,
77     * loop until a parameter error is indicated.
78     */
79     if (expectedCount == 0)
80         expectedCount = 99999999;
81     uint64_t count = 0;
82     for (uint64_t f = 1; f <= expectedCount; f++) {
83         try {
84             auto frame = stream->getFrame(f);
85             count++;
86             /* Do something with frame.data */
87             std::cout << "frame size is "
88                      << frame.size.xSize << "x" << frame.size.ySize
89                      << std::endl;
90         } catch (Error::ParameterError &e) {
91             cout << "No more frames.";
92             break;
93         } catch (Error::Exception &e) {
94             std::cout << "Caught " << e.whatString() << endl;
95             return (EXIT_FAILURE);
96         }
97     }
98     cout << "Retrieved " << count << " frames." << endl;
99     return (EXIT_SUCCESS);
100 }

```


Chapter 13

Device

The `Device` package consists of classes, constants, and other structures used to communicate with hardware devices. These include smartcards that conforms to the ISO Smartcard standard [5].

13.1 TLV

The `TLV` class represents a single tag-length-value object as described in [5]. The data for a `TLV` can be represented in two manners:

- As a “raw” set of octets; this is the format used by smartcards;
- As an object giving accessed to the parsed fields, data, and children.

Both “constructed” and “primitive” basic-encoding-rule (BER) `TLV` objects are supported by the `TLV` class. Methods are provided to obtain the children of a constructed BER-`TLV` and to obtain the data of a primitive BER-`TLV`.

13.2 Smartcard

13.2.1 APDU

The `APDU` represents an [Application Protocol Data Unit \(APDU\)](#) that is sent to a card. An `APDU` object directly represents the data according to [5] as all fields of the the class are public. Applications can send an `APDU` to the card, but the more effective approach is to subclass `Smartcard` and wrap `APDU` communication with methods that are specific to the type of card.

13.2.2 Smartcard Communication

The `Smartcard` class provides generic access to a any card that is inserted in the system. An application on the card can be activated during construction. Card data objects can be retrieved based on the object ID, and any `APDU` can be sent to the card.

Because communicating with a card depends on a command/response protocol, `Smartcard` provides methods to retrieve the response returned by the card. This retrieval is useful when the status words must be examined as many commands can result in several values for each status word.

Listing 13.1: Accessing a PIV smartcard

```

1 #include <iostream>
2 #include <be_device_smartcard.h>
3 #include <be_device_tlv.h>
4 #include <be_error_exception.h>
5
6 int main(int argc, char *argv[])
7 {
8     std::cout << "Attempt to activate PIV: " << std::endl;
9     for (int i = 0; i < 4; i++) {
10         try {
11             std::cout << "\tReader " << i << ": ";
12             BE::Device::Smartcard smc(i,
13                                     {0xA0, 0x00, 0x00, 0x03, 0x08, 0x00, 0x00,
14                                      0x10, 0x00, 0x01, 0x00});
15             std::cout << "Found." << std::endl;
16
17             std::cout << "Get Card Capability Container: "
18                         << std::endl;
19             BE::Memory::uint8Array
20                 objID{0x5C, 0x03, 0x5F, 0xC1, 0x07};
21             auto obj = smc.getDedicatedFileObject(objID);
22
23             /* The CCC is contained within a TLV */
24             std::cout << BE::Device::TLV::stringFromTLV(obj, 1);
25
26             /* Do something with the TLV data, which is the CCC */
27             BE::Device::TLV tlv(obj);
28             processCCC(tlv.getPrimitive());
29
30             // The card responded with something other than normal
31             // processing complete, catch the exception from the
32             // Framework so the status words can be examined.
33         } catch (BE::Device::Smartcard::APDUException &e) {
34             std::cout << "Bad response: ";
35             printf("0x%02hhX%02hhX\n",
36                  e.response.sw1, e.response.sw2);
37             std::cout << "Sent APDU: " << std::endl;
38             // Dump the octets from the sent APDU
39             dumpUInt8Array(e.apdu);
40         } catch (BE::Error::ParameterError &e) {
41             std::cout << "Caught " << e.whatString();
42         } catch (BE::Error::StrategyError &e) {
43             std::cout << "Other error: " << e.whatString();
44         }
45         std::cout << std::endl;
46     }
47     return (EXIT_SUCCESS);
48 }

```

The example code in Listing 13.1 shows how to activate the PIV smartcard and retrieve one of its data objects.

Chapter 14

Feature

The `Feature` package contains those items that relate to the representation of biometric features, such as fingerprint minutiae, facial features (eyes, etc.), and related information. Objects of these class types are typically associated with `View` (Chapter 15 on page 51) or `DataInterchange` (Chapter 19 on page 63) objects. For example, a minutiae object is usually obtained from a finger view, which may have been obtained from a data interchange object representing an entire biometric record for an individual.

The data contained within a `Feature` object is represented as the “native” format as it was extracted from the underlying data record. There is no translation to a common format and it is the application’s responsibility to interpret or translate the data as necessary.

Currently, fingerprint and palm print minutiae are the features supported within the `BCommon`. As development continues, additional features contained within biometric data records will be supported.

14.1 ANSI/NIST Features

The ANSI/NIST [6] standard defines several features represented as data elements within a record. Fingerprint and palm minutiae is contained within Type-9 record. The `AN2K7Minutiae` class, contained in the `Feature` package, represents a single Type-9 record. An object of this class can be constructed directly from a complete ANSI/NIST record. However, it is more common for an application to retrieve these objects from the `AN2KView` object defined in the `Finger` package (Chapter 16 on page 53).

See Listing 16.1 on page 54 for a complete example of how to obtain the fingerprint minutiae data from an ANSI/NIST record. If only extended feature set data is required from the file, a `Feature::AN2K11EFS::ExtendedFeatures` object can be created directly from the file or memory buffer.

14.1.1 ANSI/NIST 2011 Extended Feature Sets

The 2011 edition of the ANSI/NIST standard [7] adds a new form of feature data representation to the Type-9 fingerprint minutiae record. The extended feature set information is represented by an object that can be retrieved from the `AN2KMinutiaeDataRecord` object created from the data file.

Listing 14.1 shows how to read the extended feature set data from an ANSI/NIST file, both as a data interchange object (see Section 19 on page 63) or an extended feature set object constructed directly from a file.

Listing 14.1: Using AN2K Extended Feature Sets

```
1 #include <iostream>
2 #include <be_data_interchange_an2k.h>
3 #include <be_feature_an2k11efs.h>
4
```

```

5  /*
6   * This test program exercises the Evaluation framework to process AN2K
7   * records stored in a RecordStore. The intent is to model what a real
8   * program would do by retrieving AN2K records, doing some processing
9   * on the image, and displaying the results.
10  */
11  using namespace BiometricEvaluation;
12
13  static void
14  printAN2K11EFS (Feature::AN2K11EFS::ExtendedFeatureSet &efs)
15  {
16      Image::ROI roi = efs.getImageInfo().roi;
17      std::cout << "ROI:\n"
18          << "\tSize: ("
19          << roi.size.xSize << "," << roi.size.ySize << ")\n"
20          << "\tOffset: ("
21          << roi.horzOffset << "," << roi.vertOffset << ")\n"
22          << "\tPath: ";
23      for (auto const& point: roi.path) {
24          std::cout << point << " ";
25      }
26      std::cout << "\n";
27
28      std::cout << "Image Info:\n" << efs.getImageInfo() << "\n\n";
29
30      Feature::AN2K11EFS::CorePointSet cps = efs.getCPS();
31      std::cout << "CPS: Have " << cps.size() << " EFS core point(s):\n";
32      for (auto const& cp: cps) {
33          std::cout << "\t" << cp << "\n";
34      }
35
36      Feature::AN2K11EFS::DeltaPointSet dps = efs.getDPS();
37      std::cout << "DPS: Have " << dps.size() << " EFS delta point(s):\n";
38      for (auto const& dp: dps) {
39          std::cout << "\t" << dp << "\n";
40      }
41
42      Feature::AN2K11EFS::MinutiaPointSet mps = efs.getMPS();
43      std::cout << "MPS: Have " << mps.size() << " EFS minutia point(s):\n";
44      for (auto const& mp: mps) {
45          std::cout << mp << "\n";
46      }
47
48      std::cout << "No Features Present:\n";
49      std::cout << efs.getNFP();
50
51      std::cout << "\nMinutiae Ridge Count Information:\n";
52      auto mrci = efs.getMRCI();
53      std::cout << mrci << "\n";
54  }
55
56  int
57  main(int argc, char* argv[]) {
58
59      std::string fname = "test_data/type9-efs.an2k";
60      /*

```

```

61      * Read the EFS data from the DataInterchange::AN2KRecord object
62      */
63      std::cout << "Extended Feature Set data in " << fname << ": ";
64      try {
65          DataInterchange::AN2KRecord an2k(fname);
66          std::vector<Finger::AN2KMinutiaeDataRecord> minutiae =
67              an2k.getMinutiaeDataRecordSet();
68          printAN2K11EFS(*minutiae[0].getAN2K11EFS());
69      } catch (Error::Exception &e) {
70          std::cout << "Failed; caught " << e.whatString() << "\n";
71      }
72
73      /*
74      * Read the EFS data by constructing directly from the filename
75      */
76      try {
77          Feature::AN2K11EFS::ExtendedFeatureSet efs(fname, 1);
78          printAN2K11EFS(efs);
79      } catch (Error::Exception &e) {
80          std::cout << "Failed; caught " << e.whatString() << "\n";
81      }
82  }

```

14.2 ISO/INCITS Features

The ISO [4] and INCITS [1] fingerprint minutiae standards are represented within BECommon with the same class, `INCITSMinutiae`, as the minutiae format is identical in both standards.

Listing 16.2 on page 55 shows how to create a view object for the fingerprint minutiae record contained in a file.

Chapter 15

View

Within the Biometric Evaluation Framework a view represents all the information that was derived from an image of a biometric sample. For example, with a fingerprint image, any minutiae that were extracted from that image, as well as the image itself, are contained within a single `View` object. In many cases the image may not be present, however the image size and other information is contained within a biometric record, along with the derived information. A view is used to represent these records as well.

In the case where a raw image is part of the biometric record, the `View` object's related `Image` (Chapter 11 on page 37) object will have identical size, resolution, etc. values because the `View` class sets the `Image` attributes directly. For other image types (e.g. JPEG) the `Image` object will return attribute values taken from the image data.

Views are high-level abstractions of the biometric sample, and concrete implementations of a `View` include finger, face, iris, etc. views based on a specific type of biometric. Therefore, `View` objects are not created directly. Subclasses, such as finger views (see Chapter 16 on page 53), represent the specific type of biometric sample.

`View` objects are created with information taken from a biometric data record, an ANSI/NIST 2007 file, for example. Most record formats contain information about the image itself, such as the resolution and size. The object can be used to retrieve this information. However, the data may differ from that contained in the image itself, and applications can compare the corresponding values taken from the `Image` object (when available) to those taken from the `View` object.

Listing 15.1 shows a function that will print the information obtained from any `View` object.

Listing 15.1: `View::View` Class

```
1 void
2 printViewInfo(BiometricEvaluation::View::View &view)
3 {
4     cout << "Image size is " << view.getImageSize() << endl;
5     cout << "Image resolution is " << view.getImageResolution() << endl;
6     cout << "Scan resolution is " << view.getScanResolution() << endl;
7     cout << "Image color depth is " << view.getImageColorDepth() << endl;
8     cout << "Compression is " << view.getCompressionAlgorithm() << endl;
9     try {
10         auto theImage = view.getImage();
11         cout << "Information from the Image data item:" << endl;
12         cout << "\tResolution: " << theImage->getResolution() << endl;
13         cout << "\tDimensions: " << theImage->getDimensions() << endl;
14         cout << "\tDepth: " << theImage->getColorDepth() << endl;
15     } catch (Error::Exception &e) {
16         cout << "Caught " << e.what() << endl;
17     }
```

18 | }

15.1 ANSI/NIST Views

The ANSI/NIST standard [6] describes fixed and variable resolution finger, latent, and palm image records. These are represented within BECommon by `View::AN2KView` (subclass of `View::View` and `View::AN2KViewVariableResolution`, subclass of `AN2KView`). As these classes only define the common interface for the ANSI/NIST records, objects of these class types cannot be created. These classes are further extended by classes in the `Finger`, `Latent`, and `Palm` name spaces. See [16.1](#) and [17.1](#).

Chapter 16

Finger

One of the most commonly used biometric source is the fingerprint. Multiple types of information can be derived from a fingerprint, including minutiae and the pattern, such as whorl, etc. The `Finger` package contains the types, classes, and other items that are related to fingers and fingerprints. Objects of the `Finger` classes are typically not used in a stand-alone fashion, but are usually obtained from an object in the `DataInterchange` (Chapter 19 on page 63) package.

Several enumerated types are defined in the `Finger` package. The types are used to represent those elements related to fingers and fingerprints that are common across all data formats. Types that represent finger position, impression type, and others are included in the package. Stream operators are defined for these types so they can be printed in human-readable format.

Most of the classes in the `Finger` package represent data taken directly from a record in a standard format (e.g. ANSI/NIST [6]). In addition to general information, such as finger position, other information may be represented: The source of the finger image; the quality of the image, etc. In addition to this descriptive information, the finger object will provide the set of derived minutiae or other data sets.

When representing the information about a finger (and fingerprint), the class in the `Finger` package implements the interface defined in the `View` package. A finger is a specific type of view in that it represents all the available information about the finger, including the source image, minutiae (often in several formats), as well as the capture data (date, location, etc.)

16.1 ANSI/NIST Minutiae Data Record

Finger views are objects that represent all the available information for a specific finger as contained in one or more biometric records. For example, an ANSI/NIST file may contain a Type-3 record (finger image) and an associated Type-9 record (finger minutiae). A finger view object based on the ANSI/NIST record can be instantiated and used by an application to retrieve all the desired information, including the source finger image. The internals of record processing and error handling are encapsulated within the class.

The `BECommon` provides several classes that are derived from a base `View` class, contained within the `Finger` package. See Chapter 16 for more information on the types associated with fingers and fingerprints. This section discusses finger views, the classes which are derived from the general `View` class. These subclasses represent specific biometric file types, such as ANSI/NIST or INCITS/M1. In the latter case, two files must be provided when constructing the object because INCITS finger image and finger minutiae records are defined in two separate standards.

16.1.1 ANSI/NIST Finger Views

An ANSI/NIST record may contain one or more finger views, each based on a type of finger image. These Type-3, Type-4, etc. records contain the image and Type-9 minutiae data, among other information. These

record types are grouped into either the fixed- or variable-resolution categories, and are represented as specific classes within BECommon, AN2KViewFixedResolution and AN2KViewVariableResolution.

The AN2KMinutiaeDataRecord class represents all of the information taken from a ANSI/NIST Type-9 record. A Type-9 record may include minutiae data items in several formats (standard and proprietary) and the impression type code.

Listing 16.1 shows how an application can use the AN2KViewFixedResolution to retrieve image information, image data, and derived minutiae information from a file containing an ANSI/NIST record with Type-3 (fixed resolution image) and Type-9 (fingerprint minutiae) records.

Listing 16.1: Using an AN2K Finger View

```

1 #include <iostream>
2
3 #include <be_finger_an2kview_fixedres.h>
4 #include <be_error_exception.h>
5 #include <be_io_utility.h>
6
7 using namespace BiometricEvaluation;
8 using namespace BiometricEvaluation::Framework::Enumeration;
9
10 int
11 main(int argc, char* argv[]) {
12
13     /*
14      * Call the constructor that will open an existing AN2K file.
15      */
16     std::unique_ptr<Finger::AN2KViewFixedResolution> an2kv;
17     try {
18         an2kv.reset(new Finger::AN2KViewFixedResolution(
19             "test_data/type3.an2k",
20             View::AN2KView::RecordType::Type_3, 1));
21     } catch (Error::DataError &e) {
22         std::cout << "Caught " << e.what() << std::endl;
23         return (EXIT_FAILURE);
24     } catch (Error::FileError& e) {
25         std::cout << "A file error occurred: " << e.what() << std::endl;
26         return (EXIT_FAILURE);
27     }
28     std::cout << "Image resolution is "
29         << an2kv->getImageResolution() << std::endl;
30     std::cout << "Image size is " << an2kv->getImageSize() << std::endl;
31     std::cout << "Image color depth is "
32         << an2kv->getImageColorDepth() << std::endl;
33     std::cout << "Compression is " <<
34         to_string(an2kv->getCompressionAlgorithm()) << std::endl;
35     std::cout << "Scan resolution is "
36         << an2kv->getScanResolution() << std::endl;
37     std::cout << "Impression Type: " <<
38         to_string(an2kv->getImpressionType()) << std::endl;
39
40     /*
41      * Get the compressed image data and process
42      */
43     std::shared_ptr<Image::Image> img = an2kv->getImage();
44     if (img.get() == nullptr) {
45         std::cout << "Image was nullptr" << std::endl;

```



```

46     } else {
47         // Process the image data
48     }
49     /*
50     * Get the raw image data and save to a file
51     */
52     std::ofstream img_out("imgdata.raw", std::ofstream::binary);
53     Memory::uint8Array imgData{img->getRawData()};
54     img_out.write((char *)&(imgData[0]), imgData.size());
55     if (img_out.good()) {
56         img_out.close();
57     } else {
58         std::cout << "Error occurred when writing." << std::endl;
59     }
60     /*
61     * Get all the positions from the data record.
62     */
63     Finger::PositionSet positions = an2kv->getPositions();
64     std::cout << "There are " << positions.size() << " positions:"
65         << std::endl;
66     for (auto p: positions) {
67         std::cout << "\t" << to_string(p) << std::endl;
68     }
69     /*
70     * Get the minutiae data records and print the minutiae points in
71     * each data record
72     */
73     auto mdrs = an2kv->getMinutiaeDataRecordSet(); // The set of records
74     std::cout << "There are " << mdrs.size() << " minutiae data records."
75         << std::endl;
76     for (auto mdr: mdrs) {
77         for (auto mp: mdr.getAN2K7Minutiae()->getMinutiaPoints()) {
78             std::cout << mp << std::endl;
79         }
80     }
81
82     return(EXIT_SUCCESS);
83 }

```

16.1.2 ISO/INCITS Finger Views

The ISO [18] and INCITS [17] standards typically use separate files for the source biometric data and the derived data. For example, the ISO 19794-2 standard is for fingerprint minutiae data, while 19794-4 is for finger image data. The corresponding BECommon view objects are constructed with both files, although a view can be constructed with only one file. In the latter case, the view object will represent only that information contained in the single file.

(NOTE: Reading data from finger image records is not currently supported)

Listing 16.2 shows how an application can create a view from an ANSI/INCTIS 378 finger minutiae format record [1].

Listing 16.2: Using an INCITS Finger View

```

1 #include <iostream>
2 #include <be_finger_ansi2004view.h>
3 #include <be_feature_incitsminutiae.h>

```

```

4 using namespace std;
5 using namespace BiometricEvaluation;
6 using namespace BiometricEvaluation::Framework::Enumeration;
7
8 int
9 main(int argc, char* argv[])
10 {
11     Finger::ANSI2004View fngv;
12     try {
13         fngv = Finger::ANSI2004View("test_data/fmr.ansi2004", "", 3);
14     } catch (Error::Exception &e) {
15         cerr << "Caught " << e.whatString() << endl;
16         return (EXIT_FAILURE);
17     }
18     cout << "Image resolution is " << fngv.getImageResolution() << endl;
19     cout << "Image size is " << fngv.getImageSize() << endl;
20     cout << "Image color depth is " << fngv.getImageColorDepth() << endl;
21     cout << "Compression is " << fngv.getCompressionAlgorithm() << endl;
22     cout << "Scan resolution is " << fngv.getScanResolution() << endl;
23
24     Feature::INCITSMinutiae fmd = fngv.getMinutiaeData();
25     cout << "Minutiae format is " << fmd.getFormat() << endl;
26     Feature::MinutiaPointSet mps = fmd.getMinutiaPoints();
27     cout << "There are " << mps.size() << " minutiae points:" << endl;
28     for (auto mp: mps)
29         cout << mp;
30
31     Feature::RidgeCountItemSet rcis = fmd.getRidgeCountItems();
32     cout << "There are " << rcis.size() << " ridge count items:" << endl;
33     for (auto rci: rcis)
34         cout << "\t" << rci;
35
36     Feature::CorePointSet cores = fmd.getCores();
37     cout << "There are " << cores.size() << " cores:" << endl;
38     for (auto core: cores)
39         cout << "\t" << core;
40
41     Feature::DeltaPointSet deltas = fmd.getDeltas();
42     cout << "There are " << deltas.size() << " deltas:" << endl;
43     for (auto delta: deltas)
44         cout << "\t" << delta;
45
46     exit (EXIT_SUCCESS);
47 }

```

Chapter 17

Palm

The `Palm` package provides access to palm print information stored in standard record formats. Within this package are defined the common elements relevant to palm images, such as position and minutiae data.

17.1 ANSI/NIST Palm Views

The `Palm::AN2KView` class, extends `View::AN2KViewVariableResolution` (See 15) by adding methods to retrieve palm information from an ANSI/NIST ([7]) Type-15 record.

Listing 17.1 shows how an application can query the information from an ANSI/NIST data file.

Listing 17.1: Using the `Palm::AN2KView` Class

```
1 #include <iostream>
2 #include <be_io_utility.h>
3 #include <be_palm_an2kview.h>
4
5 using namespace std;
6 using namespace BiometricEvaluation;
7 using namespace BiometricEvaluation::Framework::Enumeration;
8
9 static void
10 printViewInfo(const Palm::AN2KView &an2kv) {
11     cout << "Source Agency: " << an2kv.getSourceAgency() << endl;
12     cout << "Capture Date: " << an2kv.getCaptureDate() << endl;
13     cout << "Comment: [" << an2kv.getComment() << "]" << endl;
14
15     cout << "Image resolution: " << an2kv.getImageResolution() << endl;
16     cout << "Image size: " << an2kv.getImageSize() << endl;
17     cout << "Image color depth: " << an2kv.getImageColorDepth() << endl;
18     cout << "Compression: " << an2kv.getCompressionAlgorithm() << endl;
19     cout << "Scan resolution: " << an2kv.getScanResolution() << endl;
20     cout << "Impression Type: " << an2kv.getImpressionType() << endl;
21     cout << "Position: " << an2kv.getPosition() << endl;
22     auto qms = an2kv.getPalmQualityMetric();
23     cout << "Palm Quality has " << qms.size() << " entries:" << endl;
24     for (auto &qm: qms) {
25         cout << "\t" << qm << endl;
26     }
27     shared_ptr<Image::Image> img = an2kv.getImage();
28     if (img != nullptr) {
```

```

29         cout << "Image info:" << endl;
30         cout << "\tCompression: " << img->getCompressionAlgorithm()
31             << endl;
32         cout << "\tDimensions: " << img->getDimensions() << endl;
33         cout << "\tResolution: " << img->getResolution() << endl;
34         cout << "\tDepth: " << img->getColorDepth() << endl;
35     } else {
36         cout << "No Image available." << endl;
37     }
38 }
39 }
40
41 int
42 main(int argc, char* argv[]) {
43
44     /*
45      * Call the constructor that will open an existing AN2K file.
46      */
47     std::shared_ptr<Palm::AN2KView> an2kv;
48     try {
49         an2kv.reset(new Palm::AN2KView(
50             "test_data/type9-15.an2k", 1));
51     } catch (Error::Exception &e) {
52         cout << "Caught " << e.what() << endl;
53         return (EXIT_FAILURE);
54     }
55     printViewInfo(*an2kv);
56
57     cout << "Get the set of minutiae data records: ";
58     auto minutiae = an2kv->getMinutiaeDataRecordSet();
59     cout << "There are " << minutiae.size()
60         << " minutiae data record sets." << endl;
61     if (minutiae.size() != 0) {
62         cout << "Minutiae Points:\n";
63         for (auto m:
64             minutiae[0].getAN2K7Minutiae()->getMinutiaPoints()) {
65             cout << m << endl;
66         }
67         cout << "Cores:\n";
68         for (auto c:
69             minutiae[0].getAN2K7Minutiae()->getCores()) {
70             cout << c << endl;
71         }
72         cout << "Deltas:\n";
73         for (auto d:
74             minutiae[0].getAN2K7Minutiae()->getDeltas()) {
75             cout << d << endl;
76         }
77     }
78     return (EXIT_SUCCESS);
79 }

```

Chapter 18

Face

The `Face` package provides access to facial information stored in standard record formats. Within this package are defined the common elements relevant to facial images, such as hair color, expression, pose angle, and others.

18.0.1 ISO/INCITS Face Views

The `Face::INCITSView` class, extends `View::View` (See [15](#)) by adding methods to retrieve facial information. A `Face::INCITSView` object cannot be constructed by applications but rather this class is subclassed to represent each standard format. For example, the `ISO2005View` class represents the ISO/IEC 19794-5 [\[3\]](#) standard.

Listing 18.1 shows how an application can query the information from a standard ISO/INCITS-385 facial information record.

Listing 18.1: Using the `Face::ISO2005View` Class

```
1 #include <iostream>
2 #include <iomanip>
3 #include <be_face_iso2005view.h>
4
5 using namespace std;
6 using namespace BiometricEvaluation;
7 using namespace BiometricEvaluation::Framework::Enumeration;
8
9 void
10 printViewInfo(View::View &view)
11 {
12     /*
13      * Provided by the View::View interface.
14      */
15     cout << "Image resolution is " << view.getImageResolution() << endl;
16     cout << "Scan resolution is " << view.getScanResolution() << endl;
17     cout << "Image size is " << view.getImageSize() << endl;
18     cout << "Image depth is " << view.getImageColorDepth() << endl;
19     cout << "Compression is " <<
20         view.getCompressionAlgorithm() << endl;
21
22     try {
23         std::shared_ptr<Image::Image> theImage = view.getImage();
24         cout << "Information from the Image data item:" << endl;
25         cout << "\tResolution: " << theImage->getResolution() << endl;
```

```

26         cout << "\tDimensions: " << theImage->getDimensions() << endl;
27         cout << "\tDepth: " << theImage->getColorDepth() << endl;
28     } catch (Error::Exception &e) {
29         cout << "Caught " << e.what() << endl;
30     }
31     cout << "-----" << endl;
32 }
33
34 void
35 printFaceInfo(Face::ISO2005View &facev)
36 {
37     /*
38      * Provided by the Face::INCITSView interface.
39      */
40     cout << "Gender: " << facev.getGender() << endl;
41     cout << "Eye Color: " << facev.getEyeColor() << endl;
42     cout << "Hair Color: " << facev.getHairColor() << endl;
43     cout << "Expression: " << facev.getExpression() << endl;
44
45     Face::PoseAngle pa = facev.getPoseAngle();
46     cout << "Pose angle info: ";
47     cout << "Yaw/Uncer: " << (int)pa.yaw << "/" << (int)pa.yawUncertainty;
48     cout << "; Pitch/Uncer: "
49         << (int)pa.pitch << "/" << (int)pa.pitchUncertainty;
50     cout << "; Roll/Uncer: "
51         << (int)pa.roll << "/" << (int)pa.rollUncertainty << endl;
52
53     cout << "Image type is " << facev.getImageType() << endl;
54     cout << "Image data type is " << facev.getImageDataType()
55         << endl;
56     cout << "Color space is " << facev.getColorSpace() << endl;
57     cout << "Source type is " << facev.getSourceType() << endl;
58     cout << "Device type is " << "0x" << hex << setw(4) << setfill('0')
59         << facev.getDeviceType() << dec << endl;
60
61     Face::PropertySet properties;
62     bool haveProps = facev.propertiesConsidered();
63     if (haveProps) {
64         facev.getPropertySet(properties);
65         cout << "There are " << properties.size() << " properties: ";
66         for (size_t i = 0; i < properties.size(); i++) {
67             if (i != properties.size() - 1)
68                 cout << properties[i] << ", ";
69             else
70                 cout << properties[i];
71         }
72         cout << endl;
73     } else {
74         cout << "There are no properties." << endl;
75     }
76
77     Feature::MPEGFacePointSet fps;
78     facev.getFeaturePointSet(fps);
79     cout << "There are " << fps.size() << " feature points." << endl;
80     if (fps.size() != 0) {
81         cout << "\tType\tCode\tPosition" << endl;

```

```
82     }
83     for (size_t i = 0; i < fps.size(); i++) {
84         cout << "\t" << (int)fps[i].type
85             << "\t" << (int)fps[i].major << "." << (int)fps[i].minor
86             << "\t" << fps[i].coordinate
87             << endl;
88     }
89     cout << "-----" << endl;
90 }
91
92 int
93 main(int argc, char* argv[])
94 {
95     Face::ISO2005View facev;
96     try {
97         facev = Face::ISO2005View("test_data/face01.iso2005", 1);
98     } catch (Error::Exception &e) {
99         cout << "Caught " << e.what() << endl;
100         return (EXIT_FAILURE);
101     }
102     printViewInfo(facev);
103     printFaceInfo(facev);
104     return (EXIT_SUCCESS);
105 }
```


Chapter 19

Data Interchange

The `DataInterchange` package consists of classes and other elements used to process an entire biometric data record, or set of records. For example, a single ANSI/NIST record, consisting of many smaller records (fingerprint images, latent data, etc.) can be accessed by instantiating a single object. Classes in this package typically use has-a relationships to classes in the `Finger` and other packages that process individual biometric samples.

The design of classes in the `DataInterchange` package allows applications to create a single object from a biometric record, such as an ANSI/NIST file. After creating this object, the application can retrieve the needed information (such as finger views [Chapter 16 on page 53](#)) from this object. A typical example would be to retrieve all images from the record and pass them into a function that extracts a biometric template or some other image processing.

19.1 ANSI/NIST Data Records

The ANSI/NIST Data Interchange package contains the classes used to represent ANSI/NIST [\[6\]](#) records. One class, `AN2KRecord`, is used to represent the entire ANSI/NIST record. An object of this class will contain objects of the `Finger` classes, as well as other packages. By instantiating the `AN2KRecord` object, the application can retrieve all the information and images contained in the ANSI/NIST record.

The `AN2KMinutiaeDataRecord` class represents an entire Type-9 record from an ANSI/NIST file. However, some components of this class are represented by classes in other packages. For example, the `AN2K7Minutiae` class in the `Feature` package represents the “standard” format minutiae in the Type-9 record.

[Listing 19.1](#) shows how an application can retrieve all finger latents (Type-13) and captures (Type-14) from an ANSI/NIST record. Also shown is the general record information such as the capture date, etc. Once the views are retrieved, the application obtains the set of minutiae records associated with that view. In addition, the example shows how the entire set of minutiae records can be read independent of a view.

[Listing 14.1 on page 47](#) shows how to retrieve the extended feature set data by constructing a data interchange object.

Listing 19.1: ANSI/NIST Data Interchange

```
1 #include <iostream>
2 #include <be_data_interchange_an2k.h>
3
4 /*
5  * This test program exercises the Evaluation framework to process an AN2K
6  * records stored in a file. The intent is to model what a real program
7  * would do by retrieving AN2K records, doing some processing on the image,
```

```

8  * and displaying the results.
9  */
10 using namespace std;
11 using namespace BiometricEvaluation;
12 using namespace BiometricEvaluation::Framework::Enumeration;
13
14 static void
15 printRecordInfo(const DataInterchange::AN2KRecord &an2k)
16 {
17     cout << "\tVersion: " << an2k.getVersionNumber() << endl;
18     cout << "\tDate: " << an2k.getDate() << endl;
19     cout << "\tDestination Agency: " <<
20         an2k.getDestinationAgency() << endl;
21     cout << "\tOriginating Agency: " <<
22         an2k.getOriginatingAgency() << endl;
23     cout << "\tTransaction Control Number: " <<
24         an2k.getTransactionControlNumber() << endl;
25     cout << "\tNative Scanning Resolution: " <<
26         an2k.getNativeScanningResolution() << endl;
27     cout << "\tNominal Transmitting Resolution: " <<
28         an2k.getNominalTransmittingResolution() << endl;
29     cout << "\tCapture Count: " << an2k.getFingerCaptureCount() << endl;
30     cout << "\tLatent Count: " << an2k.getFingerLatentCount() << endl;
31 }
32
33 static void
34 printViewInfo(const View::AN2KViewVariableResolution &an2kv)
35 {
36     cout << "\tRecord Type: " <<
37         static_cast<std::underlying_type<
38             View::AN2KView::RecordType>::type>(an2kv.getRecordType()) << endl;
39     cout << "\tImage resolution: " << an2kv.getImageResolution() << endl;
40     cout << "\tImage size: " << an2kv.getImageSize() << endl;
41     cout << "\tImage color depth: " << an2kv.getImageColorDepth() << endl;
42     cout << "\tCompression: " <<
43         to_string(an2kv.getCompressionAlgorithm()) << endl;
44     cout << "\tScan resolution: " << an2kv.getScanResolution() << endl;
45     cout << "\tImpression Type: " << to_string(an2kv.getImpressionType()) <<
46         endl;
47     cout << "\tSource Agency: " << an2kv.getSourceAgency() << endl;
48     cout << "\tCapture Date: " << an2kv.getCaptureDate() << endl;
49     cout << "\tComment: [" << an2kv.getComment() << "]" << endl;
50
51     /*
52      * Get the image data.
53      */
54     auto img = an2kv.getImage();
55     if (img != nullptr) {
56         /* Do something with the image info and data */
57         ;
58     } else {
59         cout << "No Image available.\n";
60     }
61
62     /*
63      * Print info for the minutiae associated with this view.

```

```

64     */
65     auto minutiae = an2kv.getMinutiaeDataRecordSet();
66     cout << "\tThere are " << minutiae.size() <<
67         " minutiae data records.\n";
68 }
69
70 int
71 main(int argc, char* argv[]) {
72     try {
73         DataInterchange::AN2KRecord an2k("test_data/a002.an2");
74         printRecordInfo(an2k);
75         /*
76          * Obtain the finger capture and latent views from the
77          * AN2k file.
78          */
79         int i = 0;
80         for (auto c: an2k.getFingerCaptures()) {
81             cout << "[Capture View " << i++ <<"\n";
82             printViewInfo(c);
83             cout << "\tPosition: " << c.getPosition()
84                 << endl;
85             cout << "[End of Capture View]\n";
86         }
87         i = 0;
88         for (auto l: an2k.getFingerLatents()) {
89             cout << "[Latent View " << i++ <<"\n";
90             printViewInfo(l);
91             cout << "\tPositions: ";
92             for (auto p: l.getPositions()) {
93                 cout << p << " ";
94             }
95             cout << endl << "[End of Latent View]\n";
96         }
97         /*
98          * Obtain the entire set of minutiae records from the
99          * AN2k file, independently of the view.
100         */
101         auto minutiae = an2k.getMinutiaeDataRecordSet();
102         cout << "There is a total of " << minutiae.size()
103             << " minutiae data records in the AN2K file.\n";
104         cout << ">>>>>>>>>>>>>>>>>>>>>>>>>\n";
105     } catch (Error::Exception &e) {
106         cout << "Failed sequence: " << e.what() << endl;
107         return (EXIT_FAILURE);
108     }
109 }

```

19.2 INCITS Data Records

The INCITS class of data records covers all those record formats that are derived from the standards defined by the InterNational Committee for Information Technology Standards [17]. These formats include the ANSI-2004 Finger Minutiae Record Format [1], the ISO equivalent [4], and other data formats, including finger images.

The `DataInterchange::ANSI2004Record` represents all the finger views contained in a pair of

ANSI 2004 fingerprint([1]) and finger image ([2]) records. This class supports the insert/update/remove of finger views from the data interchange record, enabling the runtime updating of the object. In addition, the encoded format of the minutia record can be obtained, enabling the read/modification/write of the record.

(NOTE: Reading data from finger image records is not currently supported)

Listing 19.2: ANSI 2004 Data Interchange

```

1 #include <iostream>
2 #include <be_data_interchange_ansi2004.h>
3
4 using namespace std;
5 using namespace BiometricEvaluation;
6 using namespace BiometricEvaluation::Framework::Enumeration;
7
8 void
9 printViewInfo(Finger::INCITSView &fngv)
10 {
11     cout << "Begin -----" << endl;
12     cout << "Image resolution is " << fngv.getImageResolution() << endl;
13     cout << "Image size is " << fngv.getImageSize() << endl;
14     cout << "Image depth is " << fngv.getImageColorDepth() << endl;
15     cout << "Compression is " << fngv.getCompressionAlgorithm() << endl;
16     cout << "Scan resolution is " << fngv.getScanResolution() << endl;
17
18     cout << "Finger position is " << fngv.getPosition() << endl;
19     cout << "Impression type is " << fngv.getImpressionType() << endl;
20     cout << "Quality is " << fngv.getQuality() << endl;
21     cout << "Eqpt ID is " << hex << showbase << fngv.getCaptureEquipmentID() << endl;
22     cout << dec;
23
24     Feature::INCITSMinutiae fmd = fngv.getMinutiaeData();
25     cout << "Minutiae format is " << to_string(fmd.getFormat()) << endl;
26     cout << "There are " << fmd.getMinutiaPoints().size()
27         << " minutiae points." << endl;
28     cout << "End -----" << endl;
29 }
30
31 bool
32 showAllViews(const DataInterchange::ANSI2004Record &record)
33 {
34     if (record.getNumFingerViews() == 0) {
35         cout << "No finger views present.\n";
36         return (true);
37     }
38     for (int i = 1; i <= record.getNumFingerViews(); i++) {
39         cout << "+++++++\n";
40         cout << "View number " << i << ":\n";
41         auto fngv = record.getView(i);
42         printViewInfo(fngv);
43         cout << "Test getMinutia(): View " << i << " has "
44             << record.getMinutia(i).getMinutiaPoints().size()
45             << " minutiae points.\n";
46     }
47     return (true);
48 }
49

```

```

50 int
51 main(int argc, char* argv[])
52 {
53     std::unique_ptr<DataInterchange::ANSI2004Record> record;
54
55     /* Construct with a file, minutia record only. */
56     try {
57         record.reset(new DataInterchange::ANSI2004Record(
58             "test_data/fmr.ansi2004", ""));
59     } catch (Error::Exception& e) {
60         cout << "A file error occurred: " << e.what() << endl;
61         return (EXIT_FAILURE);
62     }
63
64     /* Remove all views but the first */
65     record->isolateView(1);
66     showAllViews(*record);
67
68     /* Modify the minutia in a finger view */
69     auto minutiaRecord = record->getMinutia(1);
70     auto minutiaPoints = minutiaRecord.getMinutiaPoints();
71     for (auto& fm: minutiaPoints) {
72         fm.coordinate.x += 10;
73         fm.coordinate.y += 10;
74     }
75     /* Replace minutiae in the remaining view */
76     minutiaRecord.setMinutiaPoints(minutiaPoints);
77     record->setMinutia(1, minutiaRecord);
78     showAllViews(*record);
79
80     /* Obtain the ANSI-378 record and instantiate an object from it */
81     auto fmr = record->getFMR();
82     BE::Finger::ANSI2004View fmrView(fmr, Memory::uint8Array{}, 1);
83     /* The fmr object can also be written to a file */
84
85     return (EXIT_SUCCESS);
86 }

```


Chapter 20

Messaging

Biometric Evaluation Framework contains a collection of classes to facilitate receiving messages asynchronously over a network. What is done with these messages and how (or if) to respond is ultimately up to the application. BECommon uses this messaging in a concrete way to receive text-based commands from a `telnet` session over the Internet.

20.1 Message Center

`Process::MessageCenter` is the public-facing class an application uses to receive messages over a network. A *message* is a user-defined blob of data stored in an array of bytes. Instantiate a `MessageCenter`, and it will diligently await connections on the specified port in a separate process. During its run-loop, the application may poll or wait to determine if a message is waiting. The application has the choice of dealing with the message, sending a response, or ignoring the message entirely. Because the `MessageCenterListener` is in a separate process, the main run-loop of the application does not have to be interrupted. The `MessageCenter` classes utilize existing framework inter-process communication techniques to propagate messages (see Subsection 9.2.4 on page 32).

Listing 20.1: Basic MessageCenter Usage

```
1 namespace BE = BiometricEvaluation;
2
3 uint32_t clientID;
4 BE::Memory::uint8Array message;
5 BE::Process::MessageCenter mc;
6 for (;;) {
7     /* ... do work ... */
8
9     if (mc.hasUnseenMessages()) {
10         mc.getNextMessage(clientID, message);
11         std::cout << clientID << " sent a " << message.size() <<
12             " byte message." << std::endl;
13
14         Memory::AutoArrayUtility::setString(message, "ACK\n");
15         mc.sendResponse(clientID, message);
16     }
17 }
```

Messages can be sent to the `MessageCenter` in a number of ways, like `telnet` connections or `write()` ing to a socket. Messages are terminated with a newline (`\n`) character.

20.2 Command Center

It's easy to see how `MessageCenter` might be used for passing *commands* to a running application. One might want to query the *status* of an operation or ask a process to *stop*. The aim of `CommandCenter` was to take this common command-passing pattern and make it easier.

With `CommandCenter`, an application defines one or more `enum class` es using `Framework::Enumerations` (see Section 3.2 on page 5). For convenience, the application should subclass the `CommandParser` template, with the enumeration as the templated type. The base class instantiates a `MessageCenter` and listens for connections. Just like `MessageCenter`, commands do not have to be dealt with or responded to, and the application will only know if a command is awaiting a response if the application asks.

Because `CommandParser` operates off of strongly-typed enumerations, a pure virtual method, `parse(Command)`, must be implemented in the child class. It is expected that this method will simply be a `switch` statement of all possible enumerations (*commands*). The body of the `switch` will likely call other methods, each dealing with a single command.

`CommandParser` performs some additional convenience functions to help application developers quickly respond to commands. A *usage* string may be automatically sent when an invalid command is received. The application's main run-loop will never see the failed command attempt. If a valid command is received, `CommandParser` will tokenize any extra text in the sent command and store it in an easily retrieved `vector`. The method called from `parse()` can then sanity-check the arguments and send an error message back to the client if the arguments are invalid.

Listing 20.2: Basic `CommandCenter` Usage

```

1 namespace BE = BiometricEvaluation;
2
3 enum class TestCommand
4 {
5     Stop,
6     Help
7 };
8
9 template<>
10 const std::map<TestCommand, std::string>
11 BE::Framework::EnumerationFunctions<TestCommand>::enumToStringMap {
12     {TestCommand::Stop, "STOP"},
13     {TestCommand::Help, "HELP"}
14 };
15
16 class TestCommandParser : public BE::Process::CommandParser<TestCommand>
17 {
18 public:
19     void
20     parse(
21         const BE::Process::CommandParser<TestCommand>::Command &command)
22     {
23         switch (command.command) {
24             case TestCommand::Stop:
25                 this->stop(command);
26                 break;
27             case TestCommand::Help:
28                 this->help(command);
29                 break;
30         }
31     }
32

```



```

33 private:
34     void
35     stop(
36         const BE::Process::CommandParser<TestCommand>::Command &command)
37     {
38         /* Ensure proper arguments */
39         if (command.arguments.size() != 1) {
40             this->sendResponse(command.clientID, "Usage: " +
41                 to_string(command.command) + " <process>");
42             return;
43         }
44
45         /* ... perform stop operation ... */
46     }
47
48     void
49     help(
50         const BE::Process::CommandParser<TestCommand>::Command &command)
51     {
52         this->sendResponse(command.clientID, "Available Commands:\n"
53             "\tSTOP <process>\n\tHELP");
54     }
55 };
56
57 int
58 main()
59 {
60     TestCommandParser commandCenter;
61     TestCommandParser::Command command;
62     for (;;) {
63         /* ... do work ... */
64
65         if (commandCenter.hasPendingCommands()) {
66             commandCenter.getNextCommand(command);
67             commandCenter.parse();
68         }
69     }
70
71     return (EXIT_SUCCESS);
72 }

```

It's perfectly acceptable for an application to make use of more than one `CommandParser` for different enum s, assuming they are listening on different ports.

Chapter 21

Parallel Processing

21.1 MPI Parallel Processing Package

The `MPI` package is a set of APIs used implement parallel processing using the `MPI` [21] network-based messaging system. The core concept implemented in the framework is that of a distributor, one or more receivers, work packages, and a processing element to be implemented by the application.

The classes that make up the `MPI` package encapsulate all the necessary function calls and error handling in order to create an `MPI` job. Furthermore, the distribution and reception of packages containing data to be used for processing are also encapsulated within the `MPI` Framework. Lastly, logging, both for the tracing of Framework activity as well as application needs, is managed by these classes.

Figure 21.1 on the next page shows the processes and data flow for a typical parallel job using components of the Framework. The distributor process (Task-0) executes code from the `Distributor` class, and the receiver processes (Task-N) execute `Receiver` class code. Within each process is shown the Framework packages that could be used for the job. The *Lib* element refers to the “black-box” component of software being tested, a fingerprint matching library, for example. In this example, a record store is used as the data source, and record keys are sent in the work packages. On the receiving side, the keys are used to read record data (values) from the same store.

Receiver processing is separated into two areas of responsibility. Each Task-N is responsible for managing the workers (Task-N:1 ... Task-N:c) by starting them, accepting work requests, and sending a command to have them shut down when the job finishes. Each worker is responsible for consuming the contents of the work packages; that implementation is done in the application.

The partitioning of responsibility enables two features of the Framework. First, a worker process can handle signals or other errors and decide to shutdown without affecting the rest of the job. This capability is important when testing “black-box” software where function calls cannot be trusted.

Second, each Task-N can perform some work before creating the workers. One example is the loading of a large data set into memory; again, this is done within the application. Because Task-N calls the POSIX function `fork()` to create the workers, each worker inherits the work done by Task-N. In the case of a memory load, each worker now has that memory mapped into it’s address space. See Section 21.7 on page 77 for more details.

21.2 Work Package

A `WorkPackage` object wraps a simple container of data with some access methods. There is no information in this class pertaining to the nature or format of the data; it is simply treated as an array of unsigned integer values. However, clients of the class can store a value, the “number of elements”, that is transmitted along with the package. This value only has meaning to the client, and is usually equivalent to the number of larger-sized components making up the package. For example, this value may be the number of records contained in the package. It is up to the client of `WorkPackage` to understand how to separate the array into components.

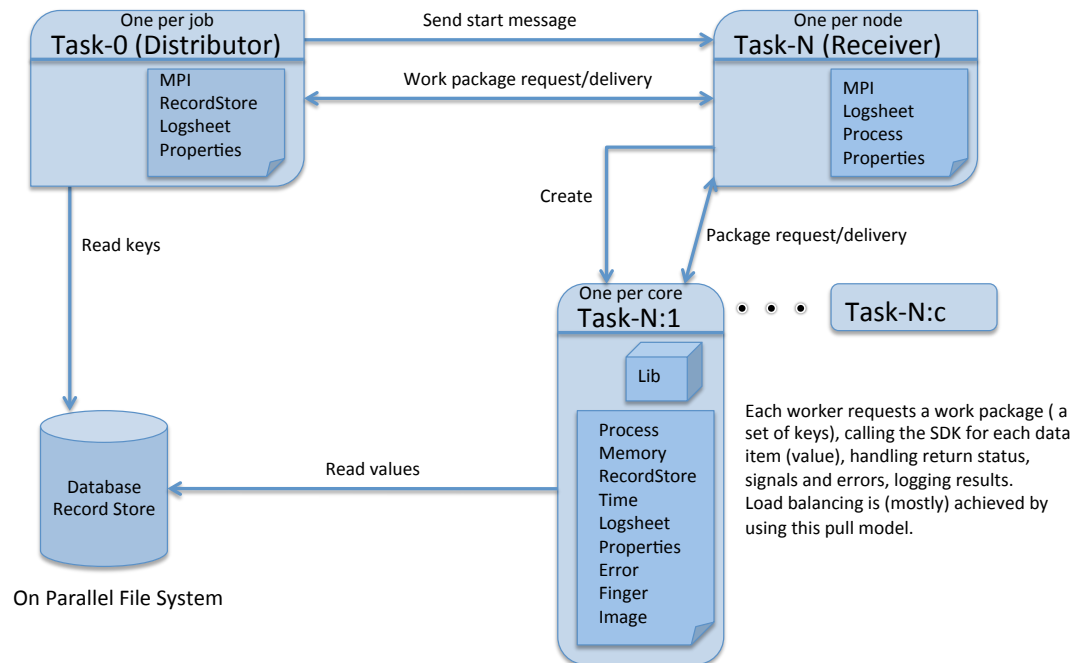


Figure 21.1: MPI Parallel Job Processes and Data Flow

The classes `RecordStoreDistributor` (Section 21.5.1 on the next page) and `RecordProcessor` (Section 21.7.1 on page 78) are examples of `WorkPackage` clients that insert and remove data from a work package.

21.3 MPI Resources

Every MPI job depends on a set of properties contained within a text file. These properties are read into a `Properties` object contained within the `Resources` object.

The core MPI classes (`Distributor` and `Receiver`) use these properties:

Workers Per Node Used by the receiver process to start the required number of workers. This value is either an integer string, or one of the special values:

NUMCPUS The number of logical CPUs, also known as hyperthreads;

NUMCORES The number of processing cores;

NUMSOCKETS The number of physical processor sockets.

Choosing the number of workers based on one of the special values depends on whether the processing is to take advantage of instruction pipelines, cache hierarchies, or other features of the processor hardware implementation.

Logsheet URL Used by distributor and receiver processes (and children) to open the log.

The `Logsheet URL` property is optional, and if present all MPI Framework trace messages will be written to the specified logging target. Two types of Uniform Resource Locator schemes are allowed: `file://` and `syslog://`, corresponding to the types of `Logsheet` classes (Section 6.3 on page 18) in the Framework.

Checkpoint Path Used by the distributor process to place checkpoint files. This property is required when checkpointing is enabled, otherwise ignored.

Subclasses and other components of the MPI Framework may add properties as needed, usually to the same file as the above properties.

Applications can add one or more properties to the file as needed. One example would be a URL for a `Logsheet` used only by the application.

21.4 Checkpoint Save and Restore

The MPI package supports checkpointing, where the state of an MPI job can be saved and restored. In the current implementation, checkpoints simply save information about the last work package that was distributed. Hence, the `Distributor` classes (See Section 21.5 on the following page) are responsible for saving this state. As a consequence, a distributor cannot record whether the work package was actually processed. When checkpointing is enabled, the resources (see Section 21.3) file for the job must contain the `Checkpoint Path` property.

A checkpoint is saved when the application enables the checkpoint capability via the `Runtime` (21.8) object, and a clean shutdown is performed by sending a signal to the distributor process, `Task-0`.

A checkpoint is restored when the application enables checkpointing, and the checkpoint file is opened. If restore is requested, but no checkpoint file is present, the job starts in the same manner as if checkpointing was not enabled.

While the MPI job is running, the checkpoint file will minimally contain the process ID of the `Task-0` distributor. A job script or other program can use this ID to shutdown the job with a complete checkpoint. An example command is:

```
kill -QUIT `cat /tmp/Distributor.chk | grep PID | cut -d= -f2`
```

21.5 Distributor

The `Distributor` is an abstract class that encapsulates the MPI functionality and is responsible for distributing work packages to other elements within the MPI job (the receivers). However, this class is also responsible for coordinating the startup and shutdown of the receiver tasks. MPI messages are used for this coordination. An MPI job may fail to start if the distributor fails to initialize, or if none of the receivers initialize.

One method of the `Distributor` class, `createWorkPackage()`, is implemented by child classes. This method creates a single work package with the knowledge of how the elements of the package are to be stored in the package's data buffer. `RecordStoreDistributor` is an implementation of `Distributor`.

For distributors, a basic checkpoint file is created. This text file is a set of key-value pairs. The `Distributor` class writes this information to the checkpoint file:

PID The process ID of the Task-0 distributor. This value is written on job startup and can be used to send the clean shutdown signal to the process.

21.5.1 Record Store Distributor

`RecordStoreDistributor` reads records from a `RecordStore`, packs record keys, and optionally, values into a `WorkPackage`. This class inherits all of the MPI communication, intra-job coordination, logging, and other aspects of the `Distributor` parent class. `RecordStoreDistributor` supports checkpoint save and restore.

An application can create an instance of a `RecordStoreDistributor` with the name of a record store in order to distribute records for processing across the MPI job. Listing 21.3 on page 83 shows an example section of code to create a record store distributor. In this type of application there is no need for the application code to refine any of the Framework classes.

Class `RecordStoreDistributor` has these additional MPI resources:

Input Record Store The input record store,

Chunk Size How many record keys or key-value pairs to place into a work package.

For a record store job, an example properties file might be:

```
Input Record Store = test.rs
Chunk Size = 7
Workers Per Node = 3
Logsheets URL = file://mpi.log
Checkpoint Path = /tmp
```

The `RecordStoreDistributor` class writes this information to the checkpoint file:

Reason A string describing the reason the checkpoint was taken.

Last Key The last record store key that was distributed.

Num Keys The number of keys that were distributed.

21.5.2 CSV Distributor

`CSVDistributor` reads text lines from an input file with no other semantic reasoning. The lines of the file are distributed in the work package containing an application-defined line count. Additional features of the `CSVDistributor` class include randomizing the input, reading the entire file into a buffer before distribution begins, and checkpoint support.

Class `CSVDistributor` has these additional MPI resources:

Input CSV The input CSV file.

Chunk Size How many lines of the file to distribute in a work package.

Read Entire File Read the entire file into buffer; “YES” or “NO”.

CSV Delimiter Character delimiter used to tokenize lines of the file.

Randomize Lines Whether to randomize distribution of the data; “YES” or “NO”

Random Seed Integer value used to seed the random function.

Trim CSV Whitespace Whether to trim white space from the input lines; “YES” or “NO”.

The `CSVDistributor` class writes this information to the checkpoint file:

Reason A string describing the reason the checkpoint was taken.

Line Count The number of lines from the CSV file that were distributed.

Random Seed The seed that was used to randomize the CSV file. Required when the “Random Seed” property is used in the distributor resources file.

21.6 Receiver

The `Receiver` class encapsulates all the MPI messaging needed to participate in the MPI job as the receiver of data to be processed. In addition, this class is responsible for starting other processes that perform work on the actual data from the work package.

It is expected, as part of the MPI job, that a single receiver process will be started on each node in the job. More than one can be started, however. Each receiver starts one or more child processes to consume data. The receiver monitors each worker process and will instruct them to shut down when the job is finished (no more data), early termination signals are received, or in the case of errors encountered by the receiver.

By keeping the data consumers as separate processes, the receiving half of the MPI job can be more robust as a premature termination of a worker process (due to memory corruption, for example) will not affect other workers.

21.7 Work Package Processor

The `WorkPackageProcessor` class is pure-virtual and provides the interface for any class that uses a `WorkPackage` to receive data from the MPI Framework. `WorkPackageProcessor` also maintains a `Logsheet` object which can be used by subclasses to store log messages.

Implementations of this class can be considered to have dual responsibilities. First is the management of common state used by all workers (Task-N:c in Figure 21.1 on page 74); creating state data shared by all workers, for example. Second, as a factory to create a package consumer for the worker process.

The `performInitialization()` method is called before the `Receiver` object forks and creates the worker processes. The application can use this function to load a large data set into memory (taking advantage of copy-on-write memory semantics present in most modern operating systems), or perform any node-local setup that should only be done once the MPI job has begun.

`newProcessor()` returns a new instance of the package processor. This method is called by the Framework when a new process is started by the receiver to consume work packages sent by the distributor. This method is a factory, creating new instances of the `WorkPackageProcessor` implementation. Therefore, it must create a “fully-formed” object that may have different state than that created by the class constructor. An example would be creating an output log file with record information. This output file would not be created in the constructor because the object returned from that will not process a work package; it is the factory object.

It is the responsibility of the `newProcessor()` method to ensure there is no resource contention between instances of this class, as the methods of this object will be executed within a separate process. The `MPI::generateUniqueID()` function can be used to create a name string that to identify the process.

The `performShutdown()` method is optionally implemented by the application to take action after all the work packages have been distributed, and is called by the framework after all the workers have terminated. The default implementation of this method does nothing.

21.7.1 Record Processor

`RecordProcessor` is a partial implementation of `WorkPackageProcessor` and defines the `processWorkPackage()` of the `WorkPackageProcessor` interface; other methods are declared as pure-virtual and must be implemented by a child class. In addition, `RecordProcessor` declares a new pure-virtual method, `processRecord()` to be implemented by a subclass to process a single record from the record store. In summary, `RecordProcessor` removes records from the work package to be processed within the subclass, which is defined by the application. See Listing 21.1 on the facing page and Listing 21.2 on page 80 for an example of such an implementation.

21.8 MPI Runtime

The `Runtime` class is the interface between the application and the MPI runtime environment. The `argv` and `argc` parameters to the `main()` function as passed through to the `Runtime` object, then onto the core Open-MPI functions. The `Runtime` object also sets up a signal handler for the job, and starts the `Distributor` and `Receiver` processes. A method is also provided for the application to abort the MPI job, providing for a somewhat clean shutdown.

One optional parameter to the `Runtime` constructor control the checkpoint capability (see 21.4 and Listing 21.2 on page 80):

checkpointEnable Write a checkpoint file when a clean shutdown is requested and restore from a checkpoint if the file is present. This parameter defaults to `false`.

One of the key features of an MPI job under the Framework is premature shutdown with minimal loss of work. Three types of exit condition can be set by sending a signal to the distributor, receiver or worker processes.

SIGQUIT Exit when the current work package is exhausted (“clean exit”);

SIGINT Exit when the current work item is finished (“quick exit”);

SIGTERM Exit immediately (“termination exit”).

For the normal exit and quick exit cases, a clean shutdown is performed for the distributor, receivers, and all worker processes. For term exit, each worker process is terminated immediately and therefore cannot finish processing the current work item. However, distributors and receivers will shutdown in a clean manner.

Any of the signals can be sent to the distributor process, which then sends messages to the receivers. In addition, if a signal is sent to a receiver or worker process, only that process (receiver or worker) is affected, but the termination condition is communicated “up” the chain. By selectively sending signals to certain processes, a user can shutdown the entire job (send to the distributor), an entire node (send to the receiver on that node), or a single worker. A worker receiving a signal sends a message back to the receiver. Likewise, a receiver will communicate the shutdown state back to the distributor.

In addition to sending signals from outside the process, a worker can shutdown itself or the entire job through exceptions. Any type of exception thrown from within a worker will cause that individual worker to shutdown, and its status will be communicated up the chain. A special type of exception, `TerminateJob`, will shutdown the individual worker, and additionally communicate up the chain to the distributor that all other workers should immediately exit. Throwing `TerminateJob` from a worker is similar in result to sending `SIGTERM` to a distributor.

21.9 Logging

In order to aid tracing and debugging of a parallel job, the MPI Framework can be configured to write trace messages to the log storage. These trace messages are logged as debug messages instead of normal entries. The type and location of the log is given to the Framework by using a URL as a property when starting the MPI job (see Section 21.3 on page 75).

When the URL for a log is the `file://` type, the MPI Framework will create several log files on the node where it runs. The reason for this is that during `Receiver` processing, one or more worker processes are created in addition to the main receiver process. Each of these processes requires exclusive access to the file-based log sheet in order to avoid conflicts with the log entry commitment. The log files will be named with the property value as a prefix, and the hostname/MPI task number/process ID added as a suffix. For example, if the property is `file://mpijob.log`, a log file might have a name of `mpijob.log-node01-1-12345`.

To aid logging within the application, access to the `Logsheet` opened by the Framework is available via the class whose interface is implemented within the application, `WorkPackageProcessor`, for example.

Two wrapper functions, `MPI::logMessage()` and `MPI::logEntry()`, are provided in order to “safely” log. These functions handle all errors from the `Logsheet` object, and will turn off log message commitment once an error occurs. The Framework and application can continue processing.

21.10 MPI Framework Applications

An application of the MPI Framework is responsible for implementing several functions declared in the Framework, requiring subclassing of the MPI classes. In this section an example application that processes records from a store will be described.

Listing 21.1 shows the header file that declares a subclass of `RecordProcessor`. The `newProcessor()`, `performInitialization()`, and `processRecord()` methods are those required to complete an implementation of `RecordProcessor`. A memory buffer pointer is managed with a smart pointer object.

Listing 21.1: MPI Framework Application Classes

```

1 class TestRecordProcessor : public BiometricEvaluation::MPI::RecordProcessor {
2 public:
3     /**
4      * @brief
5      * The property string 'Logsheet URL'.
6      */
7     static const std::string RECORDLOGSHEETURLPROPERTY;
8
9     static const uint32_t SHAREDMEMORYSIZE = 2048;
10
11     TestRecordProcessor(
12         const std::string &propertiesFileName);
13     ~TestRecordProcessor();
14
15     std::shared_ptr<BE::MPI::WorkPackageProcessor>
16     newProcessor(std::shared_ptr<BE::IO::Logsheet> &logsheet);
17
18     void
19     performInitialization(std::shared_ptr<BE::IO::Logsheet> &logsheet);
20
21     void processRecord(const std::string &key);
22
23     void processRecord(
24         const std::string &key,
25         const BE::Memory::uint8Array &value);

```

```

26
27     void
28     performShutdown();
29 protected:
30 private:
31     std::shared_ptr<BE::IO::Logsheet> _recordLogsheet;
32     std::shared_ptr<char> _sharedMemory;
33     uint32_t _sharedMemorySize;
34 };

```

Next, Listing 21.2 shows the implementation of the class methods. In this simple example, each record is acknowledged with a log entry.

Also shown in several of the methods is the use of the `Logsheet` object provided to the application by the Framework, along with wrapper functions, `logMessage()` and `logEntry()`.

The application also creates its own `Logsheet` object in order to separate Framework log messages from the application messages when processing the actual record. In error cases, the Framework log is used in order to keep the set of calls from the Framework to the application in sequence and package processing together.

A common memory buffer is allocated in `performInitialization()` method, and this buffer's pointer is copied to each processing instance in the `newProcessor()` method. Access to this common memory is shown in each `processRecord()` method. The actual memory buffer is not copied because the Framework will invoke the system call `fork()` which results in all memory of the parent process being copied into the child.

Listing 21.2: MPI Framework Application Implementation

```

1 #include <be_mpi_receiver.h>
2 #include <be_mpi_recordstoredistributor.h>
3 #include <be_mpi_runtime.h>
4
5 #include "test_be_mpi.h"
6
7 using namespace BiometricEvaluation;
8
9 static const std::string DefaultPropertiesFileName("test_be_mpi.props");
10
11 /*
12  * Implementations of the MPI RecordProcessor class interface.
13  * Calls the parent constructor to manage the properties file name.
14  */
15 TestRecordProcessor::TestRecordProcessor(
16     const std::string &propertiesFileName) :
17     RecordProcessor(propertiesFileName)
18 {
19 }
20
21 TestRecordProcessor::~TestRecordProcessor()
22 {
23 }
24
25 /*
26  * Factory object: Log our call and set up the shared memory buffer.
27  */
28 void
29 TestRecordProcessor::performInitialization(
30     std::shared_ptr<IO::Logsheet> &logsheet)
31 {

```

```

32     this->setLogsheet(logsheet);
33
34     /*
35      * Set up the memory that will be shared across all instances.
36      */
37     char *buf = (char *)malloc(SHAREDMEMORYSIZE);
38     strcpy(buf, "SHARED MEMORY");
39     this->_sharedMemorySize = SHAREDMEMORYSIZE;
40     this->_sharedMemory = std::unique_ptr<char>(buf);
41
42     *logsheet.get() << std::string(__FUNCTION__) << " called: ";
43     *logsheet.get()
44         << "Shared memory size is " << this->_sharedMemorySize
45         << " and contents is [" << buf << "];
46     BE::MPI::logEntry(*logsheet.get());
47 }
48
49 /*
50  * Factory object: Create a new instance of the TestRecordProcess
51  * that will work on work package records. Each instance gets
52  * its own instance of the log sheet.
53  */
54 std::shared_ptr<BiometricEvaluation::MPI::WorkPackageProcessor>
55 TestRecordProcessor::newProcessor(
56     std::shared_ptr<IO::Logsheet> &logsheet)
57 {
58     std::string propertiesFileName =
59         this->getResources()->getPropertiesFileName();
60     TestRecordProcessor *processor =
61         new TestRecordProcessor(propertiesFileName);
62     processor->setLogsheet(logsheet);
63
64     /*
65      * If we have our own Logsheet property, and we can open
66      * that Logsheet, use it for record logging; otherwise,
67      * create a Null Logsheet for these events. We use the
68      * framework's Logsheet for tracing of processing, not
69      * record handling logs.
70      */
71     std::string url;
72     std::unique_ptr<BE::IO::PropertiesFile> props;
73     try {
74         /* It is crucial that the Properties file be
75          * opened read-only, else it will be rewritten
76          * when the unique ptr is destroyed, causing
77          * a race condition with other processes that
78          * are reading the file.
79          */
80         props.reset(new BE::IO::PropertiesFile(
81             propertiesFileName, IO::READONLY));
82         url = props->getProperty(
83             TestRecordProcessor::RECORDLOGSHEETURLPROPERTY);
84     } catch (BE::Error::Exception &e) {
85         url = "";
86     }
87     processor->_recordLogsheet = BE::MPI::openLogsheet(

```

```

88         url, "Test Record Processing");
89     processor->_sharedMemory = this->_sharedMemory;
90     processor->_sharedMemorySize = this->_sharedMemorySize;
91
92     std::shared_ptr<BiometricEvaluation::MPI::WorkPackageProcessor> sptr;
93     sptr.reset(processor);
94     return (sptr);
95 }
96
97 /*
98  * Helper function to log some information about a record.
99  */
100 static void
101 dumpRecord(
102     BE::IO::Logsheet &log,
103     const std::string key,
104     const Memory::uint8Array &val)
105 {
106     log << "Key [" << key << "]: ";
107     /* Dump some bytes from the record */
108     for (uint64_t i = 0; i < 8; i++) {
109         log << std::hex << (int)val[i] << " ";
110     }
111     log << " |";
112     for (uint64_t i = 0; i < 8; i++) {
113         log << (char)val[i];
114     }
115     log << " |";
116     BE::MPI::logEntry(log);
117 }
118
119 /*
120  * The worker object: Log to the Framework Logsheet, obtain the data for
121  * the record, and log some information to the record Logsheet.
122  */
123 void
124 TestRecordProcessor::processRecord(const std::string &key)
125 {
126     BE::IO::Logsheet *log = this->getLogsheet().get();
127
128     if (this->getResources()->haveRecordStore() == false) {
129         BE::MPI::logMessage(*log, "processRecord(" + key + ") "
130             + " called but have no record store; returning.");
131         return;
132     }
133     *log << "processRecord(" << key << ") called: ";
134     char *buf = this->_sharedMemory.get();
135     *log << "Shared memory size is " << this->_sharedMemorySize
136         << " and contents is [" << buf << "];";
137     BE::MPI::logEntry(*log);
138
139     Memory::uint8Array value(0);
140     std::shared_ptr<IO::RecordStore> inputRS =
141         this->getResources()->getRecordStore();
142     try {
143         inputRS->read(key, value);

```

```

144         } catch (Error::Exception &e) {
145             *log << string(__FUNCTION__) <<
146                 " could not read record: " <<
147                 e.whatString();
148             return;
149         }
150     /*
151     * Log record info to our own Logsheet instead of
152     * the one provided by the framework.
153     */
154     BE::IO::Logsheet *rlog = this->_recordLogsheet.get();
155     dumpRecord(*rlog, key, value);
156 }
157
158 /*
159 * The worker object: Log to the Framework Logsheet, and log some record
160 * information to the record Logsheet.
161 */
162 void
163 TestRecordProcessor::processRecord(
164     const std::string &key,
165     const BiometricEvaluation::Memory::uint8Array &value)
166 {
167     BE::IO::Logsheet *log = this->getLogsheet().get();
168     *log << "processRecord(" << key << ", [" << value << "]) called: ";
169     char *buf = this->_sharedMemory.get();
170     *log << "Shared memory size is " << this->_sharedMemorySize
171         << " and contents is [" << buf << "]\n";
172     BE::MPI::logEntry(*log);
173
174     /*
175     * Log record info to our own Logsheet instead of
176     * the one provided by the framework.
177     */
178     BE::IO::Logsheet *rlog = this->_recordLogsheet.get();
179     dumpRecord(*rlog, key, value);
180 }
181
182 /*
183 * Factory object: Log our call.
184 */
185 void
186 TestRecordProcessor::performShutdown()
187 {
188     std::shared_ptr<BE::IO::Logsheet> logsheet = this->getLogsheet();
189     *logsheet.get() << std::string(__FUNCTION__)
190         << " called in PID " << getpid() << ": ";
191     BE::MPI::logEntry(*logsheet.get());
192 }

```

Listing 21.3: MPI Framework Application Main

```

1 int
2 main(int argc, char* argv[])
3 {
4     /*

```

```

5      * Process optional checkpoint and include-values flags.
6      */
7      bool cpEnable{false}, includeValues{false};
8      char ch;
9      while ((ch = getopt(argc, argv, "cv")) != -1) {
10         switch (ch) {
11             case 'r': cpEnable = true; break;
12             case 'v': includeValues = true; break;
13         }
14     }
15     MPI::Runtime runtime(argc, argv, cpEnable);
16     std::unique_ptr<MPI::RecordStoreDistributor> distributor;
17     std::unique_ptr<MPI::Receiver> receiver;
18     std::shared_ptr<TestRecordProcessor> processor;
19
20     if (includeValues) {
21         MPI::printStatus("Test Distributor and Receiver, keys and values");
22     } else {
23         MPI::printStatus("Test Distributor and Receiver, keys only");
24     }
25     try {
26         distributor.reset(
27             new MPI::RecordStoreDistributor(propFile, includeValues));
28         processor.reset(new TestRecordProcessor(propFile));
29         receiver.reset(new MPI::Receiver(propFile, processor));
30         runtime.start(*distributor, *receiver);
31         runtime.shutdown();
32     } catch (Error::Exception &e) {
33         MPI::printStatus("Caught: " + e.whatString());
34         runtime.abort(EXIT_FAILURE);
35     } catch (...) {
36         MPI::printStatus("Caught some other exception");
37         runtime.abort(EXIT_FAILURE);
38     }
39
40     return (EXIT_SUCCESS);
41 }

```

References

- [1] *ANSI INCITS 378-2004: Finger Minutiae Format for Data Interchange*. ANSI/INCITS, 2004.
- [2] *ANSI INCITS 381-2004: Finger Image-Based Data Interchange Format*. ANSI/INCITS, 2004.
- [3] *ANSI INCITS 385-2004: Face Recognition Format for Data Interchange*. ANSI/INCITS, 2004.
- [4] *ISO/IEC 19794-2: Information technology - Biometric data interchange formats - Part 2: Finger minutiae data*. ISO/IEC, first edition, 2005.
- [5] *ISO/IEC 7816-4: Identification cards - Integrated circuit cards - Part 4: Organization, security and commands for interchange*. ISO/IEC, second edition, 2005.
- [6] *American National Standard for Information Systems - Data Format for the Interchange of Fingerprint, Facial, & Other Biometric Information*. ANSI/NIST-ITL, 1-2007 edition, 2007. NIST Special Publication 500-271.
- [7] *American National Standard for Information Systems - Data Format for the Interchange of Fingerprint, Facial, & Other Biometric Information*. ANSI/NIST-ITL, 1-2011 edition, 2011. NIST Special Publication 500-290.
- [8] Mark Adler. zlib, 2012. <http://www.zlib.net>.
- [9] Berkeley DB. <https://sqlite.org>.
- [10] CMake. <https://cmake.org>.
- [11] World Wide Web Consortium. Portable Network Graphics Standard, 2003. <http://www.w3.org/TR/PNG/>.
- [12] FFmpeg Multimedia Framework. <http://www.ffmpeg.org>.
- [13] GNU Make Project. <http://www.gnu.org/software/make>.
- [14] Independent JPEG Group. libjpeg, 2011. <http://www.ijg.org/>.
- [15] Joint Photographic Experts Group. JPEG2000 Image Standard, 1992. <http://www.jpeg.org/jpeg2000/index.html>.
- [16] Joint Photographic Experts Group. JPEG Image Standard, 2011. <http://www.jpeg.org/jpeg/index.html>.
- [17] International Committee for Information Technology Standards. <http://www.incits.org>.
- [18] ISO/IEC Joint Technical Committee 1/SC 37 Biometrics.
- [19] Communications and Remote Sensing Lab, Université catholique de Louvain. OpenJPEG Library, 2011. <http://www.openjpeg.org/>.

- [20] Microsoft. Bitmap Image File Format. https://en.wikipedia.org/wiki/BMP_file_format.
- [21] Message Passing Interface Forum. <http://www.mpi-forum.org>.
- [22] NIST Biometric Image Software, 2011. <https://www.nist.gov/services-resources/software/nist-biometric-image-software-nbis>.
- [23] NIST Image Group. <http://www.nist.gov/itl/iad/ig>.
- [24] The Open MPI Project. <http://www.openmpi.org>.
- [25] The OpenSSL Project. <http://www.openssl.org>.
- [26] Greg Roelofs. libpng, 2011. <http://www.libpng.org/pub/png/libpng.html>.
- [27] Sam Leffler. LibTIFF – TIFF Library and Utilities, May 2017. <http://www.simplesystems.org/libtiff/>.
- [28] The SQLite Project. <https://sqlite.org>.
- [29] Bjarne Stroustrup. *The C++ Programming Language*. Addison Wesley, special edition, 2000.
- [30] The Syslog Protocol. <http://tools.ietf.org/html/rfc5424>.
- [31] Wavelet Scalar Quantization Gray-Scale Fingerprint Image Compression Standard, 2010. https://www.fbi/specs.cjis.gov/Document/Get?fileName=WSQ_Gray-scale_Specification_Version_3_1_Final.pdf.

Appendix A

Building the Framework

A.1 Language Features

The Biometric Evaluation Framework was developed using the 2011 version of the C++ language standard. It is not possible to subset BECommon to use an earlier version of C++.

Two implementations of C++11 known to compile BECommon are:

- GNU Compiler Collection version 4.8.2 on Linux.
- Apple LLVM version 6.0 (clang-600.0.56) on OS-X.

A.2 The Framework Build System

The distribution of BECommon includes a set of `make` files used to build the BECommon library, as well as install the library and header files. These `make` file use some features of the GNU `make` [13] system, and therefore the GNU software must be installed on the user's system. Future versions of BECommon may use a different build system.

In order to tailor the build of the BECommon library (file `libbiomeval`), the `common/src/libbiomeval/Makefile` file needs editing. At the top of this file are `make` variables for locating the header files and libraries for NBIS, and other libraries.

The `make` file also sets variables that create subsets of the BECommon. `CORE` and `IO` are required as they form the basis of the BECommon. The `SOURCES` variable contains a list of variables pertaining to the desired build of BECommon.

A.3 The CMake Build System

Building the BECommon using CMake [10] is possible, and provides a simpler cross-platform build system. In the `common/src/libbiomeval` directory is a `CMakeLists.txt` file that controls the build.

To build static and shared library versions of `libbiomeval`, including the subset of NBIS included with the Framework, the steps are:

1. Create a build directory; in this example, it will be under `libbiomeval`:

```
mkdir build; cd build
```

2. Run CMake using the `CMakeLists.txt`:

```
cmake ..
```

3. Build the Framework:

```
make
```

4. Install static and shared libraries plus headers:

```
make install
```

5. Create an RPM on CentOS or RedHat Linux systems:

```
make package
```

To build the debug version of the library, substitute for step 2:

```
cmake -DCMAKE_BUILD_TYPE=Debug ..
```

To use a different compiler for the MPI component (Intel, OpenMPI are among the supported compilers), substitute for step 2:

```
cmake -DMPI_CXX_COMPILER=mpiicpc ..
```

A.4 External Software Dependencies

The Biometric Evaluation Framework is built upon several other software packages. The packages are used for image processing, biometric data record formats, the message passing interface [21], as well as operating system and compiler tool chains.

Other common software development libraries used by BECommon are documented in the sections that follow. Specific instructions for installing these packages are not given here. However, in general, many systems that provide a packaging system split the library support into two packages: One for runtime (containing the binary library file only), and one for use when developing applications. This second package installs the header files needed to build the BECommon.

A.4.1 NIST Biometric Image Software

The NIST Biometric Image Software (NBIS) [22] is a set of packages used for ANSI-NIST [6], WSQ [31] formats, and other support. The BECommon uses NBIS to process these biometric record formats, and contains a subset of the NBIS packages. Therefore there is no need to install NBIS. However, the BECommon build system supports using an installed NBIS package as an alternative.

A.4.2 Video and Image Processing

For the Image classes, the JPEG [14], NBIS [22], OpenJPEG [19], PNG [26], and TIFF [27] development libraries are required.

For Video classes, the FFmpeg [12] libraries are used. When building from source, configure to build and install shared libraries. By default, only static libraries are built.

A.4.3 Cryptography

Cryptography support is provided by the OpenSSL [25] library. An example is the `openssl-devel` package on Linux systems which provides the `libcrypto` file and associated header files for development.

A.4.4 Sqlite

SQLite is an embedded Structured Query Language (SQL) database engine and is used by the `IO::SQLiteRecordStore` class to provide an `IO::RecordStore` that is backed by a SQLite database. Information on SQLite can be found at [28].

A.4.5 Berkeley Database

The Berkeley Database BDB [9] is available as both open source and closed source commercial variants. The BECommon class `IO::DBRecordStore` uses the BDB software to store key/value pairs. There are two versions of the BDB API; BECommon uses version 1.85 as defined in the original open source distribution.

A.4.6 Message Passing Interface

An implementation of the MPI specification must be installed on the user's system before the full BECommon can be built. However, the MPI package can be optionally left out of the BECommon build system, if desired.

One common implementation of MPI is OpenMPI [24], available as source code, or binary packages. Often the MPI runtime is a separate binary package from the MPI development software. As an example, for many Linux distributions, an example of the runtime package is `openmpi-1.6.4-3`, while the related development package would be `openmpi-devel-1.6.4-3`.

The location of the OpenMPI libraries may be installed in a specific location. For example, on the CnetOS-7 Linux distribution, the MPI libraries are installed on `/usr/lib64/openmpi/lib/`, but the dynamic linker configuration will not locate those libraries, and linking of an application against the BECommon library will fail. To fix this problem create `/etc/ld.so.conf.d/openmpi.conf` with the line `/usr/lib64/openmpi/lib/`, then run the `ldconfig` command (as root) to update the dynamic linker configuration.

To build the BECommon, both packages are installed. In order to run an MPI job, only the runtime package needs to be installed on all nodes that participate in the MPI job. Chapter B has more information on running an MPI job.

Appendix B

Running an MPI Job

B.1 OpenMPI

This chapter describes how to use the OpenMPI [24] runtime system to execute an MPI job. Some parameters passed to the `mpirun` command are related to the notions captured in the Biometric Evaluation Framework MPI support.

B.2 Example Shell Script

Listing B.1: Example Script to run MPI

```
1 #
2 #
3 # Record store for the input.
4 #
5 INPUTRS=./SD29.rs
6
7 #
8 # Create the properties file for this run
9 #
10 # Logsheet URL is used by the framework for logging and is optional.
11 # Record Logsheet URL is defined and used by the application and is
12 # optional in the test_mpi program.
13 #
14 # An example config file for rsyslogd, listening on a non-default port:
15 #
16 #     $ModLoad imtcp
17 #     # Provides TCP syslog reception
18 #     $InputTCPServerRun 2514
19 #     local0.info /home/wsalamon/sandbox/rsyslog/record.log
20 #     local1.debug /home/wsalamon/sandbox/rsyslog/debug.log
21 #
22 PROPS=test_mpi.props
23 cat > $PROPS << EOF
24 Input Record Store = $INPUTRS
25 Chunk Size = 64
26 Workers Per Node = 8
27 Logsheet URL = syslog://loghost:2514
28 Record Logsheet URL = syslog://loghost:2514
```

```

29 EOF
30
31 #
32 # Two forms of the nodes string, one for the script to copy all
33 # files out, one for the mpirun command.
34 #
35 NODES="node01b node02b node03b node04b"
36 MPINODES="node01b,node02b,node03b,node04b"
37
38 #
39 # MPIPROCS must be >= 2, is the Task-N count plus one for Task-0.
40 #
41 MPIPROCS=5
42
43 #
44 # Set any options to the OpenMPI mpirun command. The example below will
45 # turn on some tracing and how processes are mapped to nodes.
46 #
47 #MPIOPTS=" --show-progress --debug-daemons --display-devel-map"
48
49 # Where the program is run. The directory must exist on all the
50 # nodes, and this script must be started here.
51 DIR=$PWD
52
53 #
54 # LIBS is any libraries th must coexist with the program to be run.
55 #
56 LIBS=
57 PROGRAM=test_mpi
58 CPFILES="$PROGRAM $PROPS $LIBS"
59
60 #
61 # The test program and dependencies must exist on all nodes, so copy
62 # everything to the runtime directory on all nodes. It helps to run
63 # an SSH agent or something similar.
64 #
65 for n in $NODES; do
66     echo $n;
67     scp -p $CPFILES $n:$DIR;
68 done
69
70 #
71 # Run the program as an MPI job. mpirun must be in the users path.
72 # The properties file name is the only parameter to the program.
73 #
74 EXECSTR="$PROGRAM $PROPS"
75 mpirun $MPIOPTS -H $MPINODES -np $MPIPROCS --path $DIR $EXECSTR

```

Appendix C

Namespace Index

C.1 Namespace List

Here is a list of all documented namespaces with brief descriptions:

BiometricEvaluation	107
BiometricEvaluation::Error	
Exceptions, and other error handling	108
BiometricEvaluation::Face	
Biometric information relating to face images and derived information	109
BiometricEvaluation::Feature	
Biometric information relating to biometric features not specific to any type of biometric record	111
BiometricEvaluation::Feature::Sort	113
BiometricEvaluation::Finger	
Biometric information relating to finger images and derived information	115
BiometricEvaluation::Framework	
Information about the framework	117
BiometricEvaluation::Image	
Basic information relating to images	120
BiometricEvaluation::IO	
Input/Output functionality	126
BiometricEvaluation::IO::Utility	127
BiometricEvaluation::Iris	
Biometric information relating to iris images and derived information	138
BiometricEvaluation::Memory	
Support for memory-related operations	139
BiometricEvaluation::Memory::AutoArrayUtility	143
BiometricEvaluation::MPI	
Common declarations and functions for the MPI-based functionality	145
BiometricEvaluation::Palm	
Biometric information relating to palm images and derived information	149
BiometricEvaluation::Plantar	
Biometric information relating to plantar images and derived information	150
BiometricEvaluation::Process	
Process (p. 150) information and controls	150
BiometricEvaluation::System	
Operating system, hardware, etc	152

BiometricEvaluation::Text	
Text (p. 154) processing for string objects	154
BiometricEvaluation::Time	
Support for time and timers	161
BiometricEvaluation::Video	
Basic information relating to video and streams	163
BiometricEvaluation::View	
View (p. 587) information	164

Appendix D

Hierarchical Index

D.1 Class Hierarchy

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

BiometricEvaluation::Finger::AN2KMinutiaeDataRecord	171
BiometricEvaluation::View::AN2KViewVariableResolution::AN2KQualityMetric	174
BiometricEvaluation::DataInterchange::AN2KRecord	175
BiometricEvaluation::Feature::Sort::Angle	205
BiometricEvaluation::DataInterchange::ANSI2004Record	205
BiometricEvaluation::Device::Smartcard::APDU	216
BiometricEvaluation::Device::Smartcard::APDUException	219
BiometricEvaluation::Device::Smartcard::APDUResponse	220
BiometricEvaluation::Framework::API< T >	221
BiometricEvaluation::Memory::AutoArray< T >	233
BiometricEvaluation::Memory::AutoArray< uint8_t >	233
BiometricEvaluation::Memory::AutoArrayIterator< CONST, T >	243
BiometricEvaluation::Memory::AutoBuffer< T >	251
BiometricEvaluation::Memory::AutoBuffer< ANSL_NIST >	251
BiometricEvaluation::DataInterchange::AN2KRecord::CharacterSet	254
BiometricEvaluation::Image::BMP::ColorTableEntry	255
BiometricEvaluation::Process::CommandCenter< T, typename >::Command	256
BiometricEvaluation::Process::CommandCenter< T, typename >	256
BiometricEvaluation::Process::CommandCenter< T >	256
BiometricEvaluation::Process::CommandParser< T >	259
BiometricEvaluation::IO::Compressor	270
BiometricEvaluation::IO::GZip	347
BiometricEvaluation::Video::Container	281
BiometricEvaluation::Image::Coordinate	284
BiometricEvaluation::Feature::AN2K11EFS::CorePoint	285
BiometricEvaluation::Feature::CorePoint	285
BiometricEvaluation::Feature::AN2K11EFS::DeltaPoint	304
BiometricEvaluation::Feature::DeltaPoint	304
BiometricEvaluation::MPI::Distributor	305
BiometricEvaluation::MPI::CSVDDistributor	286
BiometricEvaluation::MPI::RecordStoreDistributor	517
BiometricEvaluation::DataInterchange::AN2KRecord::DomainName	308
BiometricEvaluation::Feature::AN2K7Minutiae::PatternClassification::Entry	309

exception	
BiometricEvaluation::Error::Exception	310
BiometricEvaluation::Error::ConversionError	283
BiometricEvaluation::Error::DataError	295
BiometricEvaluation::Error::FileError	316
BiometricEvaluation::Error::MemoryError	436
BiometricEvaluation::Error::NotImplemented	456
BiometricEvaluation::Error::ObjectDoesNotExist	457
BiometricEvaluation::Error::ObjectExists	458
BiometricEvaluation::Error::ObjectIsClosed	459
BiometricEvaluation::Error::ObjectIsOpen	460
BiometricEvaluation::Error::ParameterError	474
BiometricEvaluation::Error::StrategyError	567
BiometricEvaluation::MPI::Exception	312
BiometricEvaluation::MPI::TerminateJob	575
BiometricEvaluation::Feature::AN2K11EFS::ExtendedFeatureSet	313
BiometricEvaluation::IO::FileLogCabinet	317
BiometricEvaluation::Feature::AN2K7Minutiae::FingerprintReadingSystem	333
BiometricEvaluation::Finger::AN2KViewCapture::FingerSegmentPosition	334
BiometricEvaluation::Feature::AN2K11EFS::FPPPosition	345
BiometricEvaluation::Video::Frame	346
BiometricEvaluation::Feature::FrictionRidgeGeneralizedPosition	346
BiometricEvaluation::Image::Image	355
BiometricEvaluation::Image::BMP	252
BiometricEvaluation::Image::JPEG	406
BiometricEvaluation::Image::JPEG2000	408
BiometricEvaluation::Image::JPEGL	411
BiometricEvaluation::Image::NetPBM	450
BiometricEvaluation::Image::PNG	476
BiometricEvaluation::Image::Raw	497
BiometricEvaluation::Image::TIFF	576
BiometricEvaluation::Image::WSQ	611
BiometricEvaluation::Feature::AN2K11EFS::ImageInfo	367
BiometricEvaluation::Memory::IndexedBuffer	395
BiometricEvaluation::Memory::MutableIndexedBuffer	446
BiometricEvaluation::Process::Manager	430
BiometricEvaluation::Process::ForkManager	335
BiometricEvaluation::Process::POSIXThreadManager	480
BiometricEvaluation::Process::MessageCenter	436
BiometricEvaluation::Feature::Minutiae	442
BiometricEvaluation::Feature::AN2K7Minutiae	167
BiometricEvaluation::Feature::INCITSMinutiae	368
BiometricEvaluation::Feature::AN2K11EFS::MinutiaeRidgeCount	442
BiometricEvaluation::Feature::AN2K11EFS::MinutiaeRidgeCountConfidence	443
BiometricEvaluation::Feature::AN2K11EFS::MinutiaeRidgeCountInfo	444
BiometricEvaluation::Feature::MinutiaPoint	444
BiometricEvaluation::Feature::AN2K11EFS::MinutiaPoint	445
BiometricEvaluation::Feature::MPEGFacePoint	445
BiometricEvaluation::Feature::AN2K11EFS::NoFeaturesPresent	456
BiometricEvaluation::Memory::OrderedMap< Key, T >	460

BiometricEvaluation::Memory::OrderedMapConstIterator< Key, T >	465
BiometricEvaluation::Memory::OrderedMapIterator< Key, T >	469
BiometricEvaluation::Feature::AN2K11EFS::Orientation	473
ostringstream	
BiometricEvaluation::IO::Logsheet	420
BiometricEvaluation::IO::FileLogsheet	319
BiometricEvaluation::IO::SysLogsheet	569
BiometricEvaluation::Feature::AN2K7Minutiae::PatternClassification	474
BiometricEvaluation::Feature::Sort::Polar	478
BiometricEvaluation::Face::PoseAngle	480
BiometricEvaluation::View::AN2KViewVariableResolution::PrintPositionCoordinate	486
BiometricEvaluation::IO::Properties	486
BiometricEvaluation::IO::PropertiesFile	493
BiometricEvaluation::Feature::Sort::Quality	496
BiometricEvaluation::Iris::INCITSView::QualitySubBlock	496
BiometricEvaluation::MPI::Receiver	498
BiometricEvaluation::IO::RecordStore::Record	500
BiometricEvaluation::IO::RecordStore	504
BiometricEvaluation::IO::ArchiveRecordStore	223
BiometricEvaluation::IO::CompressedRecordStore	261
BiometricEvaluation::IO::DBRecordStore	295
BiometricEvaluation::IO::FileRecordStore	325
BiometricEvaluation::IO::ListRecordStore	413
BiometricEvaluation::IO::SQLiteRecordStore	554
BiometricEvaluation::IO::RecordStoreIterator	520
BiometricEvaluation::IO::RecordStoreUnion	526
BiometricEvaluation::IO::PersistentRecordStoreUnion	475
BiometricEvaluation::Image::Resolution	531
BiometricEvaluation::MPI::Resources	533
BiometricEvaluation::MPI::CSVResources	291
BiometricEvaluation::MPI::RecordStoreResources	524
BiometricEvaluation::Framework::API< T >::Result	537
BiometricEvaluation::Feature::RidgeCountItem	538
BiometricEvaluation::Image::ROI	539
BiometricEvaluation::MPI::Runtime	540
BiometricEvaluation::Process::Semaphore	541
BiometricEvaluation::Error::SignalManager	545
BiometricEvaluation::Image::Size	549
BiometricEvaluation::Device::Smartcard	550
BiometricEvaluation::Process::Statistics	560
BiometricEvaluation::Framework::Status	565
BiometricEvaluation::Video::Stream	567
BiometricEvaluation::Time::Timer	578
BiometricEvaluation::Device::TLV	581
BiometricEvaluation::Memory::unique_if< T >	586
BiometricEvaluation::Memory::unique_if< T[]>	586
BiometricEvaluation::Memory::unique_if< T[S]>	587
BiometricEvaluation::View::View	587
BiometricEvaluation::Face::INCITSView	371
BiometricEvaluation::Face::ISO2005View	400

BiometricEvaluation::Finger::INCITSView	383
BiometricEvaluation::Finger::ANSI2004View	212
BiometricEvaluation::Finger::ANSI2007View	214
BiometricEvaluation::Finger::ISO2005View	402
BiometricEvaluation::Iris::INCITSView	377
BiometricEvaluation::Iris::ISO2011View	404
BiometricEvaluation::View::AN2KView	183
BiometricEvaluation::Finger::AN2KView	188
BiometricEvaluation::Finger::AN2KViewFixedResolution	198
BiometricEvaluation::View::AN2KViewVariableResolution	200
BiometricEvaluation::Finger::AN2KViewCapture	193
BiometricEvaluation::Latent::AN2KView	187
BiometricEvaluation::Palm::AN2KView	181
BiometricEvaluation::Time::Watchdog	591
BiometricEvaluation::Process::Worker	595
BiometricEvaluation::Process::MessageCenterListener	439
BiometricEvaluation::Process::MessageCenterReceiver	440
BiometricEvaluation::Process::WorkerController	601
BiometricEvaluation::Process::ForkWorkerController	341
BiometricEvaluation::Process::POSIXThreadWorkerController	483
BiometricEvaluation::MPI::WorkPackage	606
BiometricEvaluation::MPI::WorkPackageProcessor	608
BiometricEvaluation::MPI::CSVProcessor	288
BiometricEvaluation::MPI::RecordProcessor	500
BiometricEvaluation::Feature::Sort::XY	613
BiometricEvaluation::Feature::Sort::YX	613

Appendix E

Class Index

E.1 Class List

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

BiometricEvaluation::Feature::AN2K7Minutiae	
A class to represent a set of minutiae in an ANSI/NIST record	167
BiometricEvaluation::Finger::AN2KMinutiaeDataRecord	
Representation of a Type-9 Record from an AN2K file	171
BiometricEvaluation::View::AN2KViewVariableResolution::AN2KQualityMetric	
A structure to represent an AN2K quality metric	174
BiometricEvaluation::DataInterchange::AN2KRecord	
A class to represent an entire ANSI/NIST record	175
BiometricEvaluation::Palm::AN2KView	
A class to represent a single Palm (p. 149) view and derived information	181
BiometricEvaluation::View::AN2KView	
A class to represent single biometric view and derived information	183
BiometricEvaluation::Latent::AN2KView	187
BiometricEvaluation::Finger::AN2KView	
A class to represent single finger view and derived information	188
BiometricEvaluation::Finger::AN2KViewCapture	
Represents an ANSI/NIST variable-resolution finger image	193
BiometricEvaluation::Finger::AN2KViewFixedResolution	
A class to represent single finger view and derived information	198
BiometricEvaluation::View::AN2KViewVariableResolution	
A class to represent single view based on an ANSI/NIST record	200
BiometricEvaluation::Feature::Sort::Angle	205
BiometricEvaluation::DataInterchange::ANSI2004Record	205
BiometricEvaluation::Finger::ANSI2004View	
A class to represent single finger view and derived information	212
BiometricEvaluation::Finger::ANSI2007View	
A class to represent single finger view and derived information	214
BiometricEvaluation::Device::Smartcard::APDU	216
BiometricEvaluation::Device::Smartcard::APDUException	
Exception thrown when a command fails	219
BiometricEvaluation::Device::Smartcard::APDUResponse	
The data and status words returned by the card in response to a command	220

BiometricEvaluation::Framework::API< T >	
A convenient way to execute biometric technology evaluation API (p. 221) methods safely	221
BiometricEvaluation::IO::ArchiveRecordStore	
This class implements the IO::RecordStore (p. 504) interface by storing data items in single file, with an associated manifest file	223
BiometricEvaluation::Memory::AutoArray< T >	
A C-style array wrapped in the facade of a C++ STL container	233
BiometricEvaluation::Memory::AutoArrayIterator< CONST, T >	
RandomAccessIterator for any AutoArray (p. 233)	243
BiometricEvaluation::Memory::AutoBuffer< T >	251
BiometricEvaluation::Image::BMP	
A BMP-encoded image	252
BiometricEvaluation::DataInterchange::AN2KRecord::CharacterSet	254
BiometricEvaluation::Image::BMP::ColorTableEntry	255
BiometricEvaluation::Process::CommandCenter< T, typename >::Command	256
BiometricEvaluation::Process::CommandCenter< T, typename >	256
BiometricEvaluation::Process::CommandParser< T >	259
BiometricEvaluation::IO::CompressedRecordStore	
Sibling-implemented RecordStore (p. 504) with Compression	261
BiometricEvaluation::IO::Compressor	270
BiometricEvaluation::Video::Container	
Representation of a video container	281
BiometricEvaluation::Error::ConversionError	
Error (p. 108) when converting one object into another, a property value from string to int, for example	283
BiometricEvaluation::Image::Coordinate	
A structure to contain a two-dimensional coordinate without a specified origin	284
BiometricEvaluation::Feature::AN2K11EFS::CorePoint	285
BiometricEvaluation::Feature::CorePoint	
Representation of the core	285
BiometricEvaluation::MPI::CSV Distributor	
An implementation of the MPI::Distributor abstraction that distribute lines of a text file via work packages	286
BiometricEvaluation::MPI::CSV Processor	
An implementation of a work package processor that will extract lines (and optionally tokenize) a line from a CSV text file	288
BiometricEvaluation::MPI::CSVResources	291
BiometricEvaluation::Error::DataError	
Error (p. 108) when reading data from an external source	295
BiometricEvaluation::IO::DBRecordStore	
A class that implements IO::RecordStore (p. 504) using a Berkeley DB database as the underlying record storage system	295
BiometricEvaluation::Feature::AN2K11EFS::DeltaPoint	
Representation of an extended feature set delta	304
BiometricEvaluation::Feature::DeltaPoint	
Representation of the delta	304
BiometricEvaluation::MPI::Distributor	
A class to represent an MPI (p. 145) task that distributes work to other tasks	305
BiometricEvaluation::DataInterchange::AN2KRecord::DomainName	
Representation of a domain name for the user-defined Type-2 logical record implementation	308
BiometricEvaluation::Feature::AN2K7Minutiae::PatternClassification::Entry	309

BiometricEvaluation::Error::Exception	
The parent class of all BiometricEvaluation (p. 107) exceptions	310
BiometricEvaluation::MPI::Exception	312
BiometricEvaluation::Feature::AN2K11EFS::ExtendedFeatureSet	
A class to represent the Extended Feature (p. 111) Set optionally present in an ANSI/NIST	
Type-9 record	313
BiometricEvaluation::Error::FileError	
File error when opening, reading, writing, etc	316
BiometricEvaluation::IO::FileLogCabinet	317
BiometricEvaluation::IO::FileLogsheet	
A class to represent a single logging mechanism with a file as the backing store	319
BiometricEvaluation::IO::FileRecordStore	325
BiometricEvaluation::Feature::AN2K7Minutiae::FingerprintReadingSystem	
Representation of information about a fingerprint reader system	333
BiometricEvaluation::Finger::AN2KViewCapture::FingerSegmentPosition	
Locations of an individual finger segment in a slap	334
BiometricEvaluation::Process::ForkManager	
Manager (p. 430) implementation that starts Workers by calling fork(2)	335
BiometricEvaluation::Process::ForkWorkerController	
Wrapper of a Worker (p. 595) returned from a Process::ForkManager (p. 335)	341
BiometricEvaluation::Feature::AN2K11EFS::FPPPosition	
Representation of finger-palm-plantar position	345
BiometricEvaluation::Video::Frame	346
BiometricEvaluation::Feature::FrictionRidgeGeneralizedPosition	
Representation of the position (Finger/Palm/Plantar) used in this class and child classes	346
BiometricEvaluation::IO::GZip	
Compressor (p. 270) for gzip compression from zlib	347
BiometricEvaluation::Image::Image	
Represent attributes common to all images	355
BiometricEvaluation::Feature::AN2K11EFS::ImageInfo	
A structure representing information about the image and extended feature set region	367
BiometricEvaluation::Feature::INCITSMinutiae	
A class to represent a set of minutiae in an ANSI/INCITS record	368
BiometricEvaluation::Face::INCITSView	
A class to represent single facial image view and derived information	371
BiometricEvaluation::Iris::INCITSView	
A class to represent single iris view and derived information	377
BiometricEvaluation::Finger::INCITSView	
A class to represent single finger view and derived information	383
BiometricEvaluation::Memory::IndexedBuffer	
Wrap a memory buffer with an index	395
BiometricEvaluation::Face::ISO2005View	
A class to represent single face view and derived information	400
BiometricEvaluation::Finger::ISO2005View	
A class to represent single finger view and derived information	402
BiometricEvaluation::Iris::ISO2011View	
A class to represent single iris view and derived information	404
BiometricEvaluation::Image::JPEG	
A JPEG-encoded image	406
BiometricEvaluation::Image::JPEG2000	
A JPEG-2000-encoded image	408

BiometricEvaluation::Image::JPEG	
A Lossless JPEG-encoded image	411
BiometricEvaluation::IO::ListRecordStore	
RecordStore (p. 504) that reads a list of keys from a text file, and retrieves the data from another RecordStore (p. 504)	413
BiometricEvaluation::IO::Logsheet	
A class to represent a logging mechanism	420
BiometricEvaluation::Process::Manager	
An interface for intranode process management classes	430
BiometricEvaluation::Error::MemoryError	
An error occurred when allocating an object	436
BiometricEvaluation::Process::MessageCenter	436
BiometricEvaluation::Process::MessageCenterListener	439
BiometricEvaluation::Process::MessageCenterReceiver	
Receives message from a client, forwarding to the central MessageCenter (p. 436)	440
BiometricEvaluation::Feature::Minutiae	
A class to represent a set of minutiae data points	442
BiometricEvaluation::Feature::AN2K11EFS::MinutiaeRidgeCount	
Representation of an extended feature set ridge count info	442
BiometricEvaluation::Feature::AN2K11EFS::MinutiaeRidgeCountConfidence	
Representation of an extended feature set minutiae ridge count confidence item	443
BiometricEvaluation::Feature::AN2K11EFS::MinutiaeRidgeCountInfo	
All the ridge count information in one place	444
BiometricEvaluation::Feature::MinutiaPoint	
Representation of a finger minutiae data point	444
BiometricEvaluation::Feature::AN2K11EFS::MinutiaPoint	
Representation of an extended feature set minutia data point	445
BiometricEvaluation::Feature::MPEGFacePoint	
Representation of a feature point and a set of points	445
BiometricEvaluation::Memory::MutableIndexedBuffer	446
BiometricEvaluation::Image::NetPBM	
A NetPBM-encoded image	450
BiometricEvaluation::Feature::AN2K11EFS::NoFeaturesPresent	
A set of flags indicating "No features present" indicators contained within the extended feature set	456
BiometricEvaluation::Error::NotImplemented	
A NotImplemented (p. 456) object is thrown when the underlying implementation of this interface has not or could not be created	456
BiometricEvaluation::Error::ObjectDoesNotExist	
The named object does not exist	457
BiometricEvaluation::Error::ObjectExists	
The named object exists and will not be replaced	458
BiometricEvaluation::Error::ObjectIsClosed	
The object is closed	459
BiometricEvaluation::Error::ObjectIsOpen	
The object is already opened	460
BiometricEvaluation::Memory::OrderedMap< Key, T >	460
BiometricEvaluation::Memory::OrderedMapConstIterator< Key, T >	465
BiometricEvaluation::Memory::OrderedMapIterator< Key, T >	469
BiometricEvaluation::Feature::AN2K11EFS::Orientation	
Representation of orientation (deviation from upright) and its uncertainty	473

BiometricEvaluation::Error::ParameterError	
An invalid parameter was passed to a constructor or method	474
BiometricEvaluation::Feature::AN2K7Minutiae::PatternClassification	
Pattern classification codes	474
BiometricEvaluation::IO::PersistentRecordStoreUnion	475
BiometricEvaluation::Image::PNG	
A PNG-encoded image	476
BiometricEvaluation::Feature::Sort::Polar	
Sort (p. 113) by increasing distance from center and angle (theta)	478
BiometricEvaluation::Face::PoseAngle	
Representation of pose angle and uncertainty	480
BiometricEvaluation::Process::POSIXThreadManager	
Manager (p. 430) implementation that starts Workers in POSIX threads	480
BiometricEvaluation::Process::POSIXThreadWorkerController	
Decorated Worker (p. 595) returned from a Process::POSIXThreadManager (p. 480)	483
BiometricEvaluation::View::AN2KViewVariableResolution::PrintPositionCoordinate	
Offsets to the bounding boxes for the EJI, full finger views, or EJI segments	486
BiometricEvaluation::IO::Properties	
Maintain key/value pairs of strings, with each property matched to one value	486
BiometricEvaluation::IO::PropertiesFile	
A Properties (p. 486) object persisted in a file on disk	493
BiometricEvaluation::Feature::Sort::Quality	496
BiometricEvaluation::Iris::INCITSView::QualitySubBlock	
Representation of an iris quality block	496
BiometricEvaluation::Image::Raw	
An image with no encoding or compression	497
BiometricEvaluation::MPI::Receiver	
A class to represent an MPI (p. 145) task that receives WorkPackages containers from the Distributor (p. 305)	498
BiometricEvaluation::IO::RecordStore::Record	500
BiometricEvaluation::MPI::RecordProcessor	
An implementation of a work package processor that will extract record store keys, and optionally, values, from a WorkPackage (p. 606)	500
BiometricEvaluation::IO::RecordStore	
A class to represent a data storage mechanism	504
BiometricEvaluation::MPI::RecordStoreDistributor	
An implementation of the MPI::Distributor abstraction that uses a record store for input to create the work packages	517
BiometricEvaluation::IO::RecordStoreIterator	
Generic ForwardIterator for all RecordStores	520
BiometricEvaluation::MPI::RecordStoreResources	
A class to represent a set of resources needed by an MPI (p. 145) program using a RecordStore for input	524
BiometricEvaluation::IO::RecordStoreUnion	
A collection of N related read-only RecordStores, operated on simultaneously	526
BiometricEvaluation::Image::Resolution	
A structure to represent the resolution of an image	531
BiometricEvaluation::MPI::Resources	533
BiometricEvaluation::Framework::API< T >::Result	537

BiometricEvaluation::Feature::RidgeCountItem	
Representation of ridge count data, which is the number of ridges between any two minutia data points, each represented by its index number	538
BiometricEvaluation::Image::ROI	
A structure to represent a region of interest (ROI (p. 539)), which is a bounding box and a set of coordinates	539
BiometricEvaluation::MPI::Runtime	
Runtime (p. 540) support for the startup/shutdown of MPI (p. 145) jobs	540
BiometricEvaluation::Process::Semaphore	
Represent a semaphore that can be used for interprocess communication	541
BiometricEvaluation::Error::SignalManager	
A SignalManager (p. 545) object is used to handle signals that come from the operating system	545
BiometricEvaluation::Image::Size	
A structure to represent the size of an image, in pixels	549
BiometricEvaluation::Device::Smartcard	550
BiometricEvaluation::IO::SQLiteRecordStore	
A RecordStore (p. 504) implementation using a SQLite database as the underlying record storage system	554
BiometricEvaluation::Process::Statistics	
Interface for gathering process statistics, such as memory usage, system time, etc	560
BiometricEvaluation::Framework::Status	565
BiometricEvaluation::Error::StrategyError	
A StrategyError (p. 567) object is thrown when the underlying implementation of this interface encounters an error	567
BiometricEvaluation::Video::Stream	567
BiometricEvaluation::IO::SysLogsheet	
A class to represent a single logging mechanism to a logging service on the network	569
BiometricEvaluation::MPI::TerminateJob	
An exception that when thrown from a Task should result in the entire job (all tasks) being shut down by the Distributor (p. 305)	575
BiometricEvaluation::Image::TIFF	576
BiometricEvaluation::Time::Timer	
This class can be used by applications to report the amount of time a block of code takes to execute	578
BiometricEvaluation::Device::TLV	
A class to represent a Tag-Length-Value (TLV (p. 581)) data structure as described in the ISO 7816-4 integrated circuit card standard	581
BiometricEvaluation::Memory::unique_if< T >	
Define a type that is visible when T is not an array	586
BiometricEvaluation::Memory::unique_if< T[] >	
Define a type that is visible when T is an unknown-bound array	586
BiometricEvaluation::Memory::unique_if< T[S] >	
Define a type that is visible when T is a known-bound array	587
BiometricEvaluation::View::View	
A class to represent single biometric element view	587
BiometricEvaluation::Time::Watchdog	
A Watchdog (p. 591) object can be used by applications to limit the amount of processing time taken by a block of code	591
BiometricEvaluation::Process::Worker	
An abstraction of an instance that performs work on given data	595

BiometricEvaluation::Process::WorkerController	
Wrapper of a Worker (p. 595) returned from a Process::Manager (p. 430)	601
BiometricEvaluation::MPI::WorkPackage	
A class to represent a piece of work to be acted upon by a processor	606
BiometricEvaluation::MPI::WorkPackageProcessor	
Represents an object that processes the contents of a work package	608
BiometricEvaluation::Image::WSQ	
A WSQ-encoded image	611
BiometricEvaluation::Feature::Sort::XY	613
BiometricEvaluation::Feature::Sort::YX	613

Appendix F

Namespace Documentation

F.1 BiometricEvaluation Namespace Reference

Namespaces

- **Error**
Exceptions, and other error handling.
- **Face**
Biometric information relating to face images and derived information.
- **Feature**
Biometric information relating to biometric features not specific to any type of biometric record.
- **Finger**
Biometric information relating to finger images and derived information.
- **Framework**
Information about the framework.
- **Image**
Basic information relating to images.
- **IO**
Input/Output functionality.
- **Iris**
Biometric information relating to iris images and derived information.
- **Memory**
Support for memory-related operations.
- **MPI**
Common declarations and functions for the MPI-based functionality.
- **Palm**
Biometric information relating to palm images and derived information.
- **Plantar**
Biometric information relating to plantar images and derived information.
- **Process**
Process (p. [150](#)) information and controls.
- **System**
Operating system, hardware, etc.

- **Text**
Text (p. 154) processing for string objects.
- **Time**
Support for time and timers.
- **Video**
Basic information relating to video and streams.
- **View**
View (p. 587) information.

F.1.1 Detailed Description

This software was developed at the National Institute of Standards and Technology (NIST) by employees of the Federal Government in the course of their official duties. Pursuant to title 17 Section 105 of the United States Code, this software is not subject to copyright protection and is in the public domain. NIST assumes no responsibility whatsoever for its use by other parties, and makes no guarantees, expressed or implied, about its quality, reliability, or any other characteristic.

This software was developed at the National Institute of Standards and Technology (NIST) by employees of the Federal Government in the course of their official duties. Pursuant to title 17 Section 105 of the United States Code, this software is not subject to copyright protection and is in the public domain. NIST assumes no responsibility whatsoever for its use by other parties, and makes no guarantees, expressed or implied, about its quality, reliability, or any other characteristic. An interface to the object that processes a package of work from the **MPI** (p. 145) Receiver.

F.2 BiometricEvaluation::Error Namespace Reference

Exceptions, and other error handling.

Classes

- class **ConversionError**
Error (p. 108) when converting one object into another, a property value from string to int, for example.
- class **DataError**
Error (p. 108) when reading data from an external source.
- class **Exception**
*The parent class of all **BiometricEvaluation** (p. 107) exceptions.*
- class **FileError**
File error when opening, reading, writing, etc.
- class **MemoryError**
An error occurred when allocating an object.
- class **NotImplemented**
*A **NotImplemented** (p. 456) object is thrown when the underlying implementation of this interface has not or could not be created.*
- class **ObjectDoesNotExist**
The named object does not exist.
- class **ObjectExists**
The named object exists and will not be replaced.
- class **ObjectIsClosed**

- The object is closed.*

 - class **ObjectIsOpen**
- The object is already opened.*

 - class **ParameterError**
- An invalid parameter was passed to a constructor or method.*

 - class **SignalManager**
- A **SignalManager** (p. 545) object is used to handle signals that come from the operating system.*

 - class **StrategyError**
- A **StrategyError** (p. 567) object is thrown when the underlying implementation of this interface encounters an error.*

Functions

- std::string **errorStr** (bool includeErrno=false)
Convert the value of errno to a human-readable error message.
- void **SignalManagerSigHandler** (int signo, siginfo_t *info, void *uap)

F.2.1 Detailed Description

Exceptions, and other error handling.

The **Error** (p. 108) package contains classes for exceptions, and functions used for error handling, including signals generated by a process.

F.2.2 Function Documentation

F.2.2.1 errorStr()

```
std::string BiometricEvaluation::Error::errorStr (
    bool includeErrno = false )
```

Convert the value of errno to a human-readable error message.

Parameters

<i>includeErrno</i>	Whether or not to include the value of errno in the returned string.
---------------------	--

Returns

The current error message specified by errno.

F.3 BiometricEvaluation::Face Namespace Reference

Biometric information relating to face images and derived information.

Classes

- class **INCITSView**
A class to represent single facial image view and derived information.

- class **ISO2005View**
A class to represent single face view and derived information.
- struct **PoseAngle**
Representation of pose angle and uncertainty.

Typedefs

- typedef std::vector< **BiometricEvaluation::Face::Property** > **PropertySet**

Enumerations

- enum **Gender** { **Unspecified** = 0x00, **Male** = 0x01, **Female** = 0x02, **Unknown** = 0xFF }
Gender identifiers.
- enum **EyeColor** {
Unspecified = 0x00, **Black** = 0x01, **Blue** = 0x02, **Brown** = 0x03,
Gray = 0x04, **Green** = 0x05, **MultiColored** = 0x06, **Pink** = 0x07,
Unknown = 0xFF }
Eye color.
- enum **HairColor** {
Unspecified = 0x00, **Bald** = 0x01, **Black** = 0x02, **Blonde** = 0x03,
Brown = 0x04, **Gray** = 0x05, **White** = 0x06, **Red** = 0x07,
Unknown = 0xFF }
Hair color.
- enum **Property** {
Glasses = 1, **Moustache** = 2, **Beard** = 3, **Teeth** = 4,
Blink = 5, **MouthOpen** = 6, **LeftEyePatch** = 7, **RightEyePatch** = 8,
DarkGlasses = 9, **MedicalCondition** = 10 }
Face property codes.
- enum **Expression** {
Unspecified = 0x0000, **Neutral** = 0x0001, **SmileClosedJaw** = 0x0002, **SmileOpenJaw** = 0x0003,
RaisedEyebrows = 0x0004, **EyesLookingAway** = 0x0005, **Squinting** = 0x0006, **Frowning** = 0x0007 }
Face expression codes.
- enum **ImageType** { **Basic** = 0x00, **FullFrontal** = 0x01, **TokenFrontal** = 0x02 }
Face image type classification codes.
- enum **ImageDataType** { **JPEG** = 0x00, **JPEG2000** = 0x01 }
Face image data type classification codes.
- enum **ColorSpace** {
Unspecified = 0x00, **RGB24** = 0x01, **YUV422** = 0x02, **Grayscale8** = 0x03,
Other = 0x04 }
Color space codes.
- enum **SourceType** {
Unspecified = 0x00, **StaticPhotoUnknown** = 0x01, **StaticPhotoDigitalStill** = 0x02, **StaticPhotoScan** = 0x03,
VideoFrameUnknown = 0x04, **VideoFrameAnalog** = 0x05, **VideoFrameDigital** = 0x06, **Unknown** = 0x07 }
Source type codes.

F.3.1 Detailed Description

Biometric information relating to face images and derived information.

The **Face** (p. 109) package gathers all face related matters, including classes to represent face information and helper functions for conversion between biometric representations. Contained within this namespace are classes to represent specific record formats, such as ISO 19794-5.

F.3.2 Typedef Documentation

F.3.2.1 PropertySet

```
typedef std::vector< BiometricEvaluation::Face::Property> BiometricEvaluation::Face::Property←  
Set
```

A set of properties.

F.4 BiometricEvaluation::Feature Namespace Reference

Biometric information relating to biometric features not specific to any type of biometric record.

Namespaces

- **Sort**

Classes

- class **AN2K7Minutiae**
A class to represent a set of minutiae in an ANSI/NIST record.
- struct **CorePoint**
Representation of the core.
- struct **DeltaPoint**
Representation of the delta.
- struct **FrictionRidgeGeneralizedPosition**
Representation of the position (Finger/Palm/Plantar) used in this class and child classes.
- class **INCITSMinutiae**
A class to represent a set of minutiae in an ANSI/INCITS record.
- class **Minutiae**
A class to represent a set of minutiae data points.
- struct **MinutiaPoint**
Representation of a finger minutiae data point.
- struct **MPEGFacePoint**
Representation of a feature point and a set of points.
- struct **RidgeCountItem**
Representation of ridge count data, which is the number of ridges between any two minutia data points, each represented by its index number.

Typedefs

- using **FGP** = struct **FrictionRidgeGeneralizedPosition**
- using **FGPSet** = std::vector< **FGP** >
- using **AN2K7MinutiaeSet** = std::vector< std::shared_ptr< **AN2K7Minutiae** > >
- using **MinutiaPoint** = struct **MinutiaPoint**
- using **MinutiaPointSet** = std::vector< **MinutiaPoint** >
- using **RidgeCountItem** = struct **RidgeCountItem**
- using **RidgeCountItemSet** = std::vector< **RidgeCountItem** >
- using **CorePoint** = struct **CorePoint**
- using **CorePointSet** = std::vector< **CorePoint** >
- using **DeltaPoint** = struct **DeltaPoint**
- using **DeltaPointSet** = std::vector< **DeltaPoint** >
- using **MinutiaeSet** = std::vector< std::shared_ptr< **Minutiae** > >
- typedef std::vector< **MPEGFacePoint** > **MPEGFacePointSet**

Enumerations

- enum **PositionType** { **Finger** = 0, **Palm** = 1, **Plantar** = 2 }
Enumeration of the types of position classes used in this class and child classes.
- enum **MinutiaeFormat** {
 AN2K7 = 0, **IAFIS**, **Cogent**, **Motorola**,
 Sagem, **NEC**, **Identix**, **M1** }
Enumerate the minutiae format standards.
- enum **MinutiaeType** {
 RidgeEnding = 0, **Bifurcation**, **Compound**, **NoDistinction**,
 Other }
Enumerate the types of minutiae: Ridge Ending, Bifurcation, Compound, or other.
- enum **RidgeCountExtractionMethod** { **NonSpecific** = 0, **FourNeighbor** = 1, **EightNeighbor** = 2,
 Other = 3 }
Enumerate the types of extraction methods for ridge counts.

Functions

- std::ostream & **operator**<< (std::ostream &s, const **Feature::FGP** &fgp)
*Output stream overload for **FrictionRidgeGeneralizedPosition** (p. 346).*
- std::ostream & **operator**<< (std::ostream &, const **AN2K7Minutiae::FingerprintReadingSystem** &)
*Output stream overload for **FingerprintReadingSystem**.*
- std::ostream & **operator**<< (std::ostream &, const **MinutiaPoint** &)
- std::ostream & **operator**<< (std::ostream &, const **RidgeCountItem** &)
- std::ostream & **operator**<< (std::ostream &, const **CorePoint** &)
- std::ostream & **operator**<< (std::ostream &, const **DeltaPoint** &)

F.4.1 Detailed Description

Biometric information relating to biometric features not specific to any type of biometric record.

Definition of an MPEG4 **Face** (p. 109) feature point. See ISO/IEC 14496-2.

F.4.2 Function Documentation

F.4.2.1 operator<<()

```
std::ostream& BiometricEvaluation::Feature::operator<< (
    std::ostream & s,
    const Feature::FGP & fgp )
```

Output stream overload for **FrictionRidgeGeneralizedPosition** (p. 346).

Parameters

in	<i>s</i>	Stream on which to append formatted information.
in	<i>fgp</i>	FrictionRidgeGeneralizedPosition (p. 346) information to append to stream.

Returns

stream with a fgp textual representation appended.

F.5 BiometricEvaluation::Feature::Sort Namespace Reference

Classes

- class **Angle**
- class **Polar**
 - Sort* (p. 113) by increasing distance from center and angle (theta).
- class **Quality**
- class **XY**
- class **YX**

Enumerations

- enum **Kind** {
 Kind::XYAscending, **Kind::XYDescending**, **Kind::YXAscending**, **Kind::YXDescending**,
Kind::QualityAscending, **Kind::QualityDescending**, **Kind::AngleAscending**, **Kind::AngleDescending**,
Kind::PolarCOMAscending, **Kind::PolarCOMDescending**, **Kind::PolarCOIAscending**, **Kind::PolarCOIDescending**,
Kind::Unknown }

Functions

- void **updateIndicies** (BiometricEvaluation::Feature::MinutiaPointSet &mps)
 - Renumber index numbers in a MinutiaPointSet in place.*
- std::vector< **Feature::MinutiaPoint** > **sort** (std::vector< **Feature::MinutiaPoint** > &minutia, const **Kind** &sortOrder)
 - Sort* (p. 113) minutia.
- std::vector< **Feature::MinutiaPoint** > **stableSort** (std::vector< **Feature::MinutiaPoint** > &minutia, const **Kind** &sortOrder)
 - Sort* (p. 113) minutia, maintaining existing order if elements are otherwise deemed equal.

F.5.1 Detailed Description

Utilities for sorting MinutiaPointSets.

F.5.2 Enumeration Type Documentation

F.5.2.1 Kind

enum **BiometricEvaluation::Feature::Sort::Kind** [strong]
Sort (p. 113) order of MinutiaPointSets.

Enumerator

XYAscending	Lowest to highest X value, followed by Y value.
XYDescending	Highest to lowest X value, followed by Y value.
YXAscending	Lowest to highest Y value, followed by X value.
YXDescending	Highest to lowest Y value, followed by X value.
QualityAscending	Lowest to highest quality value.
QualityDescending	Highest to lowest quality value.
AngleAscending	Lowest to highest angle (theta) value.
AngleDescending	Highest to lowest angle (theta) value.
PolarCOMAscending	Lowest to highest distance from center of minutia mass, followed by angle (theta).
PolarCOMDescending	Highest to lowest distance from center of minutia mass, followed by angle (theta).
PolarCOIAscending	Lowest to highest distance from center of image, followed by angle (theta).
PolarCOIDescending	Highest to lowest distance from center of img, followed by angle (theta).
Unknown	Sort (p. 113) order cannot be determined.

F.5.3 Function Documentation

F.5.3.1 sort()

```
std::vector< Feature::MinutiaPoint> BiometricEvaluation::Feature::Sort::sort (
    std::vector< Feature::MinutiaPoint > & minutia,
    const Kind & sortOrder )
Sort (p. 113) minutia.
```

Parameters

<i>minutia</i>	Minutia to be sorted.
<i>sortOrder</i>	Order in which to sort minutia.

Exceptions

Error::NotImplemented (p. 456)	sortOrder is not implemented.
Error::StrategyError (p. 567)	Center of mass is specified, but no minutia.

F.5.3.2 stableSort()

```
std::vector< Feature::MinutiaPoint> BiometricEvaluation::Feature::Sort::stableSort (
    std::vector< Feature::MinutiaPoint > & minutia,
    const Kind & sortOrder )
```

Sort (p. 113) minutia, maintaining existing order if elements are otherwise deemed equal.

Parameters

<i>minutia</i>	Minutia to be sorted.
<i>sortOrder</i>	Order in which to sort minutia.

Exceptions

Error::NotImplemented (p. 456)	sortOrder is not implemented.
Error::StrategyError (p. 567)	Center of mass is specified, but no minutia.

F.6 BiometricEvaluation::Finger Namespace Reference

Biometric information relating to finger images and derived information.

Classes

- class **AN2KMinutiaeDataRecord**
Representation of a Type-9 Record from an AN2K file.
- class **AN2KView**
A class to represent single finger view and derived information.
- class **AN2KViewCapture**
Represents an ANSI/NIST variable-resolution finger image.
- class **AN2KViewFixedResolution**
A class to represent single finger view and derived information.
- class **ANSI2004View**
A class to represent single finger view and derived information.
- class **ANSI2007View**
A class to represent single finger view and derived information.
- class **INCITSView**
A class to represent single finger view and derived information.
- class **ISO2005View**
A class to represent single finger view and derived information.

Typedefs

- using **PositionSet** = std::vector< **Position** >
- using **PositionDescriptors** = std::map< **Position**, **FingerImageCode** >

Enumerations

- enum **PatternClassification** {
PlainArch = 0, **TentedArch**, **RadialLoop**, **UlnarLoop**,
PlainWhorl, **CentralPocketLoop**, **DoubleLoop**, **AccidentalWhorl**,
Whorl, **RightSlantLoop**, **LeftSlantLoop**, **Scar**,
Amputation, **Unknown** }
- enum **Position** {
Unknown = 0, **RightThumb** = 1, **RightIndex** = 2, **RightMiddle** = 3,
RightRing = 4, **RightLittle** = 5, **LeftThumb** = 6, **LeftIndex** = 7,
LeftMiddle = 8, **LeftRing** = 9, **LeftLittle** = 10, **PlainRightThumb** = 11,
PlainLeftThumb = 12, **PlainRightFourFingers** = 13, **PlainLeftFourFingers** = 14, **LeftRightThumbs**
= 15,
RightExtraDigit = 16, **LeftExtraDigit** = 17, **UnknownFrictionRidge** = 18, **EJI** = 19,
RightIndexMiddle = 40, **RightMiddleRing** = 41, **RightRingLittle** = 42, **LeftIndexMiddle** = 43,
LeftMiddleRing = 44, **LeftRingLittle** = 45, **RightIndexLeftIndex** = 46, **RightIndexMiddleRing** =
47,
RightMiddleRingLittle = 48, **LeftIndexMiddleRing** = 49, **LeftMiddleRingLittle** = 50, **PlainRight**↵
FourTips = 51,
PlainLeftFourTips = 52, **PlainRightFiveTips** = 53, **PlainLeftFiveTips** = 54 }
- *Finger position codes.*
- enum **Impression** {
LiveScanPlain = 0, **LiveScanRolled** = 1, **NonLiveScanPlain** = 2, **NonLiveScanRolled** = 3,
LatentImpression = 4, **LatentTracing** = 5, **LatentPhoto** = 6, **LatentLift** = 7,
LiveScanVerticalSwipe = 8, **LiveScanPalm** = 10, **NonLiveScanPalm** = 11, **LatentPalmImpression** =
12,
LatentPalmTracing = 13, **LatentPalmPhoto** = 14, **LatentPalmLift** = 15, **LiveScanOpticalContact**↵
Plain = 20,
LiveScanOpticalContactRolled = 21, **LiveScanNonOpticalContactPlain** = 22, **LiveScanNonOptical**↵
ContactRolled = 23, **LiveScanOpticalContactlessPlain** = 24,
LiveScanOpticalContactlessRolled = 25, **LiveScanNonOpticalContactlessPlain** = 26, **LiveScan**↵
NonOpticalContactlessRolled = 27, **Other** = 28,
Unknown = 29 }
- enum **FingerImageCode** {
EJI = 0, **RolledTip**, **FullFingerRolled**, **FullFingerPlainLeft**,
FullFingerPlainCenter, **FullFingerPlainRight**, **ProximalSegment**, **DistalSegment**,
MedialSegment, **NA** }

Functions

- std::ostream & **operator<<** (std::ostream &stream, const **AN2KViewCapture::FingerSegment**↵
Position &fsp)

Output stream overload for FingerSegmentPosition.

F.6.1 Detailed Description

Biometric information relating to finger images and derived information.

The **Finger** (p. 115) package gathers all finger related matters, including classes to represent finger minutiae and helper functions for conversion between biometric representations. Contained within this namespace are classes to represent specific record formats, such as ANSI/NIST finger image records.

F.6.2 Enumeration Type Documentation

F.6.2.1 FingerImageCode

```
enum BiometricEvaluation::Finger::FingerImageCode [strong]
```

Joint and tip codes.

F.6.2.2 Impression

```
enum BiometricEvaluation::Finger::Impression [strong]
```

Finger (p. 115), palm, and latent impression types.

F.6.2.3 PatternClassification

```
enum BiometricEvaluation::Finger::PatternClassification [strong]
```

Pattern classification codes.

F.6.2.4 Position

```
enum BiometricEvaluation::Finger::Position [strong]
```

Finger (p. 115) position codes.
These codes match those in ANSI/NIST. Other minutiae formats may have to map codes into this set.

F.7 BiometricEvaluation::Framework Namespace Reference

Information about the framework.

Classes

- class **API**
*A convenient way to execute biometric technology evaluation **API** (p. 221) methods safely.*
- class **Status**

Enumerations

- enum **APICurrentState** {
APICurrentState::NeverCalled, APICurrentState::WatchdogExpired, APICurrentState::Signal←
Caught, APICurrentState::ExceptionCaught,
APICurrentState::Running, APICurrentState::Completed }

Functions

- unsigned int **getMajorVersion** ()
Framework (p. 117) major version.
- unsigned int **getMinorVersion** ()
Framework (p. 117) minor version.
- std::string **getCompiler** ()
Compiler used to compile this framework.
- std::string **getCompileDate** ()
Date when this framework was compiled.
- std::string **getCompileTime** ()
Time (p. 161) when this framework was compiled.
- std::string **getCompilerVersion** ()
Version string of compiler used to compile this framework.
- std::string **to_string** (const **Status** &status)
Obtain a textual representation of a Status (p. 565).
- std::ostream & **operator<<** (std::ostream &s, const **Status** &status)
Output stream operator overload.

F.7.1 Detailed Description

Information about the framework.

F.7.2 Enumeration Type Documentation

F.7.2.1 APICurrentState

enum **BiometricEvaluation::Framework::APICurrentState** [strong]
Reasons operations could not complete.

Enumerator

NeverCalled	Operation was never executed.
WatchdogExpired	Watchdog timer expired.
SignalCaught	Signal handler was invoked.
ExceptionCaught	An exception was caught.
Running	Operation is running.
Completed	Operation has returned.

F.7.3 Function Documentation

F.7.3.1 getCompileDate()

std::string BiometricEvaluation::Framework::getCompileDate ()

Date when this framework was compiled.

Returns

Date when this framework was compiled, in the form "MMM DD YYYY"

F.7.3.2 getCompiler()

```
std::string BiometricEvaluation::Framework::getCompiler ( )
```

Compiler used to compile this framework.

Returns

The name of the compiler used to compile this framework.

F.7.3.3 getCompilerVersion()

```
std::string BiometricEvaluation::Framework::getCompilerVersion ( )
```

Version string of compiler used to compile this framework.

Returns

Major, minor, and patch level of the compiler used.

F.7.3.4 getCompileTime()

```
std::string BiometricEvaluation::Framework::getCompileTime ( )
```

Time (p. 161) when this framework was compiled.

Returns

Time (p. 161) when this framework was compiled, in the form "HH:MM:SS"

F.7.3.5 getMajorVersion()

```
unsigned int BiometricEvaluation::Framework::getMajorVersion ( )
```

Framework (p. 117) major version.

Returns

The major version number of the BiometricFramework

F.7.3.6 getMinorVersion()

```
unsigned int BiometricEvaluation::Framework::getMinorVersion ( )
```

Framework (p. 117) minor version.

Returns

The minor version of the **BiometricEvaluation** (p. 107) framework.

F.7.3.7 operator<<()

```
std::ostream& BiometricEvaluation::Framework::operator<< (
    std::ostream & s,
    const Status & status )
```

Output stream operator overload.

Parameters

<i>s</i>	Output stream.
<i>status</i>	Status (p. 565) object to output.

Returns

s appended with string representation of status.

F.7.3.8 to_string()

```
std::string BiometricEvaluation::Framework::to_string (
    const Status & status )
```

Obtain a textual representation of a **Status** (p. 565).

Parameters

<i>status</i>	Status (p. 565) object to convert.
---------------	---

Returns

Textual representation of status.

F.8 BiometricEvaluation::Image Namespace Reference

Basic information relating to images.

Classes

- class **BMP**
A BMP-encoded image.
- struct **Coordinate**
A structure to contain a two-dimensional coordinate without a specified origin.
- class **Image**
Represent attributes common to all images.
- class **JPEG**
A JPEG-encoded image.
- class **JPEG2000**
A JPEG-2000-encoded image.
- class **JPEGL**

- A Lossless JPEG-encoded image.*
- class **NetPBM**
 - A NetPBM-encoded image.*
- class **PNG**
 - A PNG-encoded image.*
- class **Raw**
 - An image with no encoding or compression.*
- struct **Resolution**
 - A structure to represent the resolution of an image.*
- struct **ROI**
 - A structure to represent a region of interest (**ROI** (p. 539)), which is a bounding box and a set of coordinates.*
- struct **Size**
 - A structure to represent the size of an image, in pixels.*
- class **TIFF**
- class **WSQ**
 - A WSQ-encoded image.*

Typedefs

- using **Coordinate** = struct **Coordinate**
- using **CoordinateSet** = std::vector< **Image::Coordinate** >
- using **Size** = struct **Size**
- using **Resolution** = struct **Resolution**
- using **ROI** = struct **ROI**

Enumerations

- enum **CompressionAlgorithm** {
None = 0, **Facsimile** = 1, **WSQ20** = 2, **JPEGB** = 3,
JPEGL = 4, **JP2** = 5, **JP2L** = 6, **PNG** = 7,
NetPBM = 8, **BMP** = 9, **TIFF** = 10 }
- enum **PixelFormat** { **PixelFormat::MonoWhite** = 0, **PixelFormat::MonoBlack** = 1, **PixelFormat::Gray8** = 2, **PixelFormat::RGB24** = 3 }

Functions

- std::string **to_string** (const **Coordinate** &c)
*Convert **Coordinate** (p. 284) to std::string.*
- std::ostream & **operator**<< (std::ostream &, const **Coordinate** &)
- bool **operator**== (const **Coordinate** &lhs, const **Coordinate** &rhs)
- bool **operator**!= (const **Coordinate** &lhs, const **Coordinate** &rhs)
- std::string **to_string** (const **CoordinateSet** &coordinates)
*Convert **CoordinateSet** to std::string.*
- std::ostream & **operator**<< (std::ostream &stream, const **CoordinateSet** &coordinates)
*Output stream overload for **CoordinateSet**.*
- std::string **to_string** (const **Size** &s)
*Convert **Size** (p. 549) to std::string.*
- std::ostream & **operator**<< (std::ostream &, const **Size** &)

- bool **operator==** (const **Size** &lhs, const **Size** &rhs)
- bool **operator!=** (const **Size** &lhs, const **Size** &rhs)
- std::string **to_string** (const **Resolution** &r)
 - Convert **Resolution** (p. 531) to std::string.
- std::ostream & **operator<<** (std::ostream &, const **Resolution** &)
- bool **operator==** (const **Resolution** &lhs, const **Resolution** &rhs)
- bool **operator!=** (const **Resolution** &lhs, const **Resolution** &rhs)
- float **distance** (const **Coordinate** &p1, const **Coordinate** &p2)
 - Calculate the distance between two points.
- **BiometricEvaluation::Memory::uint8Array removeComponents** (const **BiometricEvaluation::Memory::uint8Array** &rawData, const uint8_t bitDepth, const std::vector< bool > &components)
 - Remove components from a decompressed image's raw byte representation.
- std::string **to_string** (const **ROI** &r)
 - Convert **ROI** (p. 539) to std::string.
- std::ostream & **operator<<** (std::ostream &, const **ROI** &)
- bool **operator==** (const **ROI** &lhs, const **ROI** &rhs)
- bool **operator!=** (const **ROI** &lhs, const **ROI** &rhs)

Variables

- const double **CentimetersPerInch** = 2.54
- const double **MillimetersPerInch** = **CentimetersPerInch** * 10

F.8.1 Detailed Description

Basic information relating to images.

Classes and methods for manipulating images.

The **Image** (p. 355) package gathers all image related matters, including classes to represent an image, coordinates, and functions for conversion between biometric representations.

F.8.2 Enumeration Type Documentation

F.8.2.1 CompressionAlgorithm

enum **BiometricEvaluation::Image::CompressionAlgorithm** [strong]
Image (p. 355) compression algorithms.

F.8.2.2 PixelFormat

enum **BiometricEvaluation::Image::PixelFormat** [strong]
Image (p. 355) pixel formats.

Enumerator

MonoWhite	1 bit/pixel, 0 is white, 1 = black
MonoBlack	1 bit/pixel, 0 is black, 1 = white
Gray8	8-bit gray
RGB24	8-bit red/8-bit blue/8-bit green

F.8.3 Function Documentation

F.8.3.1 distance()

```
float BiometricEvaluation::Image::distance (
    const Coordinate & p1,
    const Coordinate & p2 )
```

Calculate the distance between two points.

Parameters

in	<i>p1</i>	First point.
in	<i>p2</i>	Second point.

Returns

Distance between *p1* and *p2*.

F.8.3.2 operator<<()

```
std::ostream& BiometricEvaluation::Image::operator<< (
    std::ostream & stream,
    const CoordinateSet & coordinates )
```

Output stream overload for **CoordinateSet**.

Parameters

in	<i>stream</i>	Stream on which to append formatted CoordinateSet information.
in	<i>coordinates</i>	CoordinateSet information to append to stream.

Returns

stream with a coordinates textual representation appended.

F.8.3.3 removeComponents()

```
BiometricEvaluation::Memory::uint8Array BiometricEvaluation::Image::removeComponents (
    const BiometricEvaluation::Memory::uint8Array & rawData,
    const uint8_t bitDepth,
    const std::vector< bool > & components )
```

Remove components from a decompressed image's raw byte representation.

Parameters

in	<i>rawData</i>	Raw (p. 497) byte representation of an image.
----	----------------	--

Parameters

in	<i>bitDepth</i>	The number of bits that represents a single component in <code>rawData</code> (only 8 and 16 are supported).
in	<i>components</i>	A bitset representing the components of the image, where true values represent components to be removed. For example, in a 4-component image where fourth component should be removed, this parameter would be {false, false, false, true}.

Returns

Copy of `rawData` with true components removed.

Exceptions

<i>BiometricEvaluation::Error::ParameterError</i> (p. 474)	Invalid <code>bitDepth</code> parameter.
<i>BiometricEvaluation::Error::StrategyError</i> (p. 567)	<code>rawData</code> does not appear to be sized large enough for the <code>bitsPerComponent</code> and <code>components</code> provided.

F.8.3.4 `to_string()` [1/5]

```
std::string BiometricEvaluation::Image::to_string (
    const Coordinate & c )
```

Convert **Coordinate** (p. 284) to `std::string`.

Parameters

<i>c</i>	Coordinate (p. 284) to convert to <code>std::string</code> .
----------	---

Returns

`std::string` representation of `c`.

F.8.3.5 `to_string()` [2/5]

```
std::string BiometricEvaluation::Image::to_string (
    const CoordinateSet & coordinates )
```

Convert **CoordinateSet** to `std::string`.

Parameters

<i>coordinates</i>	CoordinateSet to convert to <code>std::string</code> .
--------------------	---

Returns

std::string representation of coordinates.

F.8.3.6 to_string() [3/5]

```
std::string BiometricEvaluation::Image::to_string (
    const Size & s )
```

Convert **Size** (p. 549) to std::string.

Parameters

<i>s</i>	Size (p. 549) to convert to std::string.
----------	---

Returns

std::string representation of s.

F.8.3.7 to_string() [4/5]

```
std::string BiometricEvaluation::Image::to_string (
    const Resolution & r )
```

Convert **Resolution** (p. 531) to std::string.

Parameters

<i>r</i>	Resolution (p. 531) to convert to std::string.
----------	---

Returns

std::string representation of r.

F.8.3.8 to_string() [5/5]

```
std::string BiometricEvaluation::Image::to_string (
    const ROI & r )
```

Convert **ROI** (p. 539) to std::string.

Parameters

<i>r</i>	ROI (p. 539) to convert to std::string.
----------	--

Returns

std::string representation of r.

F.8.4 Variable Documentation

F.8.4.1 CentimetersPerInch

```
const double BiometricEvaluation::Image::CentimetersPerInch = 2.54
```

Number of centimeters in one inch

F.8.4.2 MillimetersPerInch

```
const double BiometricEvaluation::Image::MillimetersPerInch = CentimetersPerInch * 10
```

Number of millimeters in one inch

F.9 BiometricEvaluation::IO Namespace Reference

Input/Output functionality.

Namespaces

- **Utility**

Classes

- class **ArchiveRecordStore**
*This class implements the **IO::RecordStore** (p. 504) interface by storing data items in single file, with an associated manifest file.*
- class **CompressedRecordStore**
*Sibling-implemented **RecordStore** (p. 504) with Compression.*
- class **Compressor**
- class **DBRecordStore**
*A class that implements **IO::RecordStore** (p. 504) using a Berkeley DB database as the underlying record storage system.*
- class **FileLogCabinet**
- class **FileLogsheet**
A class to represent a single logging mechanism with a file as the backing store.
- class **FileRecordStore**
- class **GZip**
Compressor (p. 270) for gzip compression from zlib.
- class **ListRecordStore**
***RecordStore** (p. 504) that reads a list of keys from a text file, and retrieves the data from another **RecordStore** (p. 504).*
- class **Logsheet**
A class to represent a logging mechanism.
- class **PersistentRecordStoreUnion**
- class **Properties**

Maintain key/value pairs of strings, with each property matched to one value.

- class **PropertiesFile**
A *Properties* (p. 486) object persisted in an file on disk.
- class **RecordStore**
A class to represent a data storage mechanism.
- class **RecordStoreIterator**
Generic ForwardIterator for all RecordStores.
- class **RecordStoreUnion**
A collection of *N* related read-only RecordStores, operated on simultaneously.
- class **SQLiteRecordStore**
A *RecordStore* (p. 504) implementation using a SQLite database as the underlying record storage system.
- class **SysLogsheet**
A class to represent a single logging mechanism to a logging service on the network.

Enumerations

- enum **Mode** { **Mode::ReadWrite** = 0, **Mode::ReadOnly** = 1 }

F.9.1 Detailed Description

Input/Output functionality.

The **IO** (p. 126) package contains classes and functions used to abstract input and output operations and provide for robust error handling on behalf of the application.

F.9.2 Enumeration Type Documentation

F.9.2.1 Mode

enum **BiometricEvaluation::IO::Mode** [strong]
Accessibility of object.

Enumerator

ReadWrite	Constant indicating the state of an object that manages some underlying file is accessible for reading and writing.
ReadOnly	Constant indicating the state of an object that manages some underlying file is accessible for reading only.

F.10 BiometricEvaluation::IO::Utility Namespace Reference

Functions

- void **removeDirectory** (const std::string &directory, const std::string &prefix)
Remove a directory using directory name and parent pathname.
- void **removeDirectory** (const std::string &pathname)
Remove a directory using a complete pathname.

- void **copyDirectoryContents** (const std::string &sourcepath, const std::string &targetpath, const bool removesource=false)
Copy the contents of a directory, optionally deleting the source directory contents when done.
- void **setAsideName** (const std::string &name)
Set aside a file or directory name.
- uint64_t **getFileSize** (const std::string &pathname)
- uint64_t **sumDirectoryUsage** (const std::string &pathname)
- bool **fileExists** (const std::string &pathname)
- bool **pathIsDirectory** (const std::string &pathname)
- int **makePath** (const std::string &path, const mode_t mode)
Create an entire directory tree.
- **Memory::uint8Array** **readFile** (const std::string &path, std::ios_base::openmode mode=std::ios_base::binary)
Read the contents of a file into an 8-bit AutoArray.
- void **writeFile** (const uint8_t *data, const size_t size, const std::string &path, std::ios_base::openmode mode=std::ios_base::binary)
Write the contents of a buffer to a file.
- void **writeFile** (const **Memory::uint8Array** data, const std::string &path, std::ios_base::openmode mode=std::ios_base::binary)
Write the contents of an 8-bit AutoArray to a file.
- void **readPipe** (void *data, size_t size, int pipeFD)
Read from an open pipe into a buffer.
- void **readPipe** (**Memory::uint8Array** &data, int pipeFD)
Read from an open pipe into an 8-bit AutoArray.
- void **writePipe** (const void *data, size_t size, int pipeFD)
Write the contents of a buffer to a pipe.
- void **writePipe** (const **Memory::uint8Array** &data, int pipeFD)
Write the contents of an 8-bit AutoArray to a pipe.
- bool **isReadable** (const std::string &pathname)
Determine if the real user has read access permissions to this file.
- bool **isWritable** (const std::string &pathname)
Determine if the real user has write access permissions to this file.
- std::string **createTemporaryFile** (const std::string &prefix="", const std::string &parentDir="/tmp")
Create a temporary file.
- FILE * **createTemporaryFile** (std::string &path, const std::string &prefix="", const std::string &parentDir="/tmp")
Create a temporary file.
- uint64_t **countLines** (const std::string &path)
Count the number of newline characters in a text file.
- uint64_t **countLines** (const **Memory::uint8Array** &textBuffer)
Count the number of newline characters in a buffer of a text file.

F.10.1 Detailed Description

A class containing utility functions used for **IO** (p. 126) operations. These functions are class methods.

F.10.2 Function Documentation

F.10.2.1 copyDirectoryContents()

```
void BiometricEvaluation::IO::Utility::copyDirectoryContents (
    const std::string & sourcepath,
    const std::string & targetpath,
    const bool removesource = false )
```

Copy the contents of a directory, optionally deleting the source directory contents when done.

Parameters

in	<i>sourcepath</i>	The name of the directory whose contents are to be moved.
in	<i>targetpath</i>	The name of the directory where the contents of the sourcepath are to be moved.
in	<i>removesource</i>	Flag indicating whether to remove the source directory after the copy is complete.

Exceptions

<i>Error::ObjectDoesNotExist</i> (p. 457)	The source named directory does not exist.
<i>Error::StrategyError</i> (p. 567)	An error occurred when using the underlying storage system, or the directory name or prefix is malformed.

F.10.2.2 countLines() [1/2]

```
uint64_t BiometricEvaluation::IO::Utility::countLines (
    const std::string & path )
```

Count the number of newline characters in a text file.

Parameters

<i>path</i>	Path to text file.
-------------	--------------------

Returns

Number of newline characters in file at path.

Exceptions

<i>Error::FileError</i> (p. 316)	Could not open path.
--	----------------------

F.10.2.3 countLines() [2/2]

```
uint64_t BiometricEvaluation::IO::Utility::countLines (
```

```
const Memory::uint8Array & textBuffer )
```

Count the number of newline characters in a buffer of a text file.

Parameters

<i>path</i>	Buffer of text file that has been read in.
-------------	--

Returns

Number of newline characters in buffer.

F.10.2.4 **createTemporaryFile()** [1/2]

```
std::string BiometricEvaluation::IO::Utility::createTemporaryFile (
    const std::string & prefix = "",
    const std::string & parentDir = "/tmp" )
```

Create a temporary file.

Parameters

in	<i>prefix</i>	String to be prefixed to the random temporary name.
in	<i>parentDir</i>	Where to place the temporary file.

Exceptions

Error::FileError (p. 316)	Could not create or close temporary file.
Error::MemoryError (p. 436)	Error (p. 108) allocating memory for file name.

Returns

Path to temporary file.

Note

Exclusivity is not guaranteed for the path returned, since the exclusive descriptor is closed before returning.

F.10.2.5 **createTemporaryFile()** [2/2]

```
FILE* BiometricEvaluation::IO::Utility::createTemporaryFile (
    std::string & path,
    const std::string & prefix = "",
    const std::string & parentDir = "/tmp" )
```

Create a temporary file.

Exclusivity to the file stream is guaranteed.

Parameters

out	<i>path</i>	Reference to a string that will hold the path to the opened temporary file.
in	<i>prefix</i>	String to be prefixed to the random temporary name.
in	<i>parentDir</i>	Where to place the temporary file.

Exceptions

Error::FileError (p. 316)	Could not create or close temporary file.
Error::MemoryError (p. 436)	Error (p. 108) allocating memory for file name.

Returns

Open file stream to path.

Note

Caller must fclose(3) the returned stream.

F.10.2.6 fileExists()

```
bool BiometricEvaluation::IO::Utility::fileExists (
    const std::string & pathname )
    Indicate whether a file exists.
```

Parameters

in	<i>pathname</i>	The name of the file to be checked; can be a complete path.
----	-----------------	---

Returns

true if the file exists, false otherwise.

Exceptions

Error::StrategyError (p. 567)	An error occurred when using the underlying storage system, or pathname is malformed.
---	---

F.10.2.7 getFileSize()

```
uint64_t BiometricEvaluation::IO::Utility::getFileSize (
    const std::string & pathname )
    Get the size of a file.
```

Parameters

in	<i>pathname</i>	The name of the file to be sized; can be a complete path.
----	-----------------	---

Returns

The file size.

Exceptions

<i>Error::ObjectDoesNotExist</i> (p. 457)	The named directory does not exist.
<i>Error::StrategyError</i> (p. 567)	An error occurred when using the underlying storage system, or <i>pathname</i> is malformed.

F.10.2.8 isReadable()

```
bool BiometricEvaluation::IO::Utility::isReadable (
    const std::string & pathname )
```

Determine if the real user has read access permissions to this file.

Parameters

in	<i>pathname</i>	Path to the file to check.
----	-----------------	----------------------------

Returns

true if the real user has read access permissions, false otherwise.

Warning

This function should **only** be called *after* failing to open a file, to determine a possible failure reason.

See also

BiometricEvaluation::IO::Utility::fileExists() (p. [131](#))

F.10.2.9 isWritable()

```
bool BiometricEvaluation::IO::Utility::isWritable (
    const std::string & pathname )
```

Determine if the real user has write access permissions to this file.

Parameters

in	<i>pathname</i>	Path to the file to check.
----	-----------------	----------------------------

Returns

true if the real user has write access permissions, false otherwise.

Warning

This function should **only** be called *after* failing to write to a file, to determine a possible failure reason.

See also

BiometricEvaluation::IO::Utility::fileExists() (p. 131)

F.10.2.10 makePath()

```
int BiometricEvaluation::IO::Utility::makePath (
    const std::string & path,
    const mode_t mode )
```

Create an entire directory tree.

All intermediate nodes are created if they don't exist.

Parameters

in	<i>path</i>	The path to create.
in	<i>mode</i>	The permission mode of each element in the path. See chmod(2).

Returns

0 on success, non-zero otherwise, and errno can be checked.

F.10.2.11 readFile()

```
Memory::uint8Array BiometricEvaluation::IO::Utility::readFile (
    const std::string & path,
    std::ios_base::openmode mode = std::ios_base::binary )
```

Read the contents of a file into an 8-bit AutoArray.

Parameters

<i>path</i>	Path to a file to be read.
<i>mode</i>	Bitwise OR'd arguments to send to the file stream constructor.

Returns

Contents of path in an AutoArray.

Exceptions

Error::ObjectDoesNotExist (p. 457)	path does not exist.
Error::StrategyError (p. 567)	An error occurred when using the underlying storage system.

F.10.2.12 readPipe() [1/2]

```
void BiometricEvaluation::IO::Utility::readPipe (
    void * data,
    size_t size,
    int pipeFD )
```

Read from an open pipe into a buffer.

Wraps the read(2) system call by reading the requested amount of data from a pipe file descriptor, handling all errors and signals.

Parameters

<i>data</i>	Data buffer to store the data being read.
<i>size</i>	Size of data to read.
<i>pipeFD</i>	The file descriptor of the pipe.

Exceptions

<i>ObjectDoesNotExist</i>	The writing end of the pipe has been closed.
<i>FileError</i>	The data could not be written in the entirety; Error::errorStr() (p. 109) may contain more information.

F.10.2.13 readPipe() [2/2]

```
void BiometricEvaluation::IO::Utility::readPipe (
    Memory::uint8Array & data,
    int pipeFD )
```

Read from an open pipe into an 8-bit AutoArray.

Wraps the read(2) system call by reading the requested amount of data from a pipe file descriptor, * handling all errors and signals.

Parameters

<i>data</i>	Data array to read into.
<i>pipeFD</i>	The file descriptor of the pipe.

Exceptions

<i>ObjectDoesNotExist</i>	The reading end of the pipe has been closed.
---------------------------	--

Exceptions

<i>FileError</i>	The data could not be written in the entirety; Error::errorStr() (p. 109) may contain more information.
------------------	--

F.10.2.14 removeDirectory() [1/2]

```
void BiometricEvaluation::IO::Utility::removeDirectory (
    const std::string & directory,
    const std::string & prefix )
```

Remove a directory using directory name and parent pathname.

Parameters

in	<i>directory</i>	The name of the directory to be removed, without a preceding path.
in	<i>prefix</i>	The path leading to the directory.

Exceptions

Error::ObjectDoesNotExist (p. 457)	The named directory does not exist.
Error::StrategyError (p. 567)	An error occurred when using the underlying storage system, or the directory name or prefix is malformed.

F.10.2.15 removeDirectory() [2/2]

```
void BiometricEvaluation::IO::Utility::removeDirectory (
    const std::string & pathname )
```

Remove a directory using a complete pathname.

Parameters

in	<i>pathname</i>	The complete path name of the directory to be removed,
----	-----------------	--

Exceptions

Error::ObjectDoesNotExist (p. 457)	The named directory does not exist.
Error::StrategyError (p. 567)	An error occurred when using the underlying storage system, or the path name is malformed.

F.10.2.16 setAsideName()

```
void BiometricEvaluation::IO::Utility::setAsideName (
```

```
const std::string & name )
```

Set aside a file or directory name.

A file or directory is renamed in a sequential manner. For example, if directory foo is set aside, it will be renamed foo.1. If foo is recreated by the application, and again set aside, it will be renamed foo.2. There is a limit of `uint16_t` max attempts at creating a set aside name.

Parameters

in	<i>name</i>	The path name of the file or directory to be set aside.
----	-------------	---

Exceptions

<i>Error::ObjectDoesNotExist</i> (p. 457)	The named object does not exist.
<i>Error::StrategyError</i> (p. 567)	An error occurred when using the underlying storage system, the name or prefix is malformed, or the maximum number of attempts was reached.

F.10.2.17 sumDirectoryUsage()

```
uint64_t BiometricEvaluation::IO::Utility::sumDirectoryUsage (
    const std::string & pathname )
```

Get the sum of the sizes of all files and directories in a given path.

Parameters

in	<i>pathname</i>	The name of the directory to be sized.
----	-----------------	--

Returns

The sum of file and directory sizes.

Exceptions

<i>Error::ObjectDoesNotExist</i> (p. 457)	The named directory does not exist.
<i>Error::StrategyError</i> (p. 567)	An error occurred when using the underlying storage system, or pathname is malformed.

F.10.2.18 writeFile() [1/2]

```
void BiometricEvaluation::IO::Utility::writeFile (
    const uint8_t * data,
    const size_t size,
    const std::string & path,
    std::ios_base::openmode mode = std::ios_base::binary )
```

Write the contents of a buffer to a file.

A thin wrapper around `std::ofstream`. The mode parameter has the same semantics as that for `std::ofstream` and applications must set mode for append or truncate when writing to an existing file.

Parameters

<i>data</i>	Data buffer to write.
<i>size</i>	Size of data.
<i>path</i>	Path to file to create with contents of data.
<i>mode</i>	Bitwise OR'd arguments to send to the file stream constructor.

Exceptions

<i>ObjectExists</i>	path exists and is a directory.
<i>StrategyError</i>	An error occurred when using the underlying storage system.

F.10.2.19 writeFile() [2/2]

```
void BiometricEvaluation::IO::Utility::writeFile (
    const Memory::uint8Array data,
    const std::string & path,
    std::ios_base::openmode mode = std::ios_base::binary )
```

Write the contents of an 8-bit AutoArray to a file.

A thin wrapper around `std::ofstream`. The mode parameter has the same semantics as that for `std::ofstream` and applications must set mode for append or truncate when writing to an existing file.

Parameters

<i>data</i>	Data array to write.
<i>path</i>	Path to file to create with contents of data.
<i>mode</i>	Bitwise OR'd arguments to send to the file stream constructor.

Exceptions

<i>ObjectExists</i>	path exists and is a directory.
<i>StrategyError</i>	An error occurred when using the underlying storage system.

F.10.2.20 writePipe() [1/2]

```
void BiometricEvaluation::IO::Utility::writePipe (
    const void * data,
    size_t size,
    int pipeFD )
```

Write the contents of a buffer to a pipe.

Wraps the write(2) system call by writing all data to a pipe file descriptor, handling all errors and signals.

Parameters

<i>data</i>	Data buffer to write.
<i>size</i>	Size of data.
<i>pipeFD</i>	The file descriptor of the pipe.

Exceptions

<i>ObjectDoesNotExist</i>	The reading end of the pipe has been closed.
<i>FileError</i>	The data could not be written in the entirety; Error::errorStr() (p. 109) may contain more information.

F.10.2.21 writePipe() [2/2]

```
void BiometricEvaluation::IO::Utility::writePipe (
    const Memory::uint8Array & data,
    int pipeFD )
```

Write the contents of an 8-bit AutoArray to a pipe.

Wraps the write(2) system call by writing all data to a pipe file descriptor, handling all errors and signals.

Parameters

<i>data</i>	Data array to write.
<i>pipeFD</i>	The file descriptor of the pipe.

Exceptions

<i>ObjectDoesNotExist</i>	The reading end of the pipe has been closed.
<i>FileError</i>	The data could not be written in the entirety; Error::errorStr() (p. 109) may contain more information.

F.11 BiometricEvaluation::Iris Namespace Reference

Biometric information relating to iris images and derived information.

Classes

- class **INCITSView**
A class to represent single iris view and derived information.
- class **ISO2011View**
A class to represent single iris view and derived information.

Enumerations

- enum **CaptureDeviceTechnology** { **Unknown** = 0, **CMOSCCD** = 1 }
Capture device technology identifiers.
- enum **EyeLabel** { **Undefined** = 0, **Right** = 1, **Left** = 2 }
Eye label.
- enum **ImageType** { **Uncropped** = 1, **VGA** = 2, **Cropped** = 3, **CroppedMasked** = 7 }
Iris image type classification codes.
- enum **Orientation** { **Undefined** = 0, **Base** = 1, **Flipped** = 2 }
Iris horizontal orientation classification codes.
- enum **ImageCompression** { **Undefined** = 0, **LosslessNone** = 1, **Lossy** = 2 }
Iris image compression type.
- enum **CameraRange** { **Unassigned** = 0, **Failed** = 1, **Overflow** = 2 }
Range from camera lens center to subject iris.

F.11.1 Detailed Description

Biometric information relating to iris images and derived information.

The **Iris** (p. 138) package gathers all iris related matters, including classes to represent iris information and helper functions for conversion between biometric representations. Contained within this namespace are classes to represent specific record formats, such as ISO 19794-6.

F.12 BiometricEvaluation::Memory Namespace Reference

Support for memory-related operations.

Namespaces

- **AutoArrayUtility**

Classes

- class **AutoArray**
A C-style array wrapped in the facade of a C++ STL container.
- class **AutoArrayIterator**
*RandomAccessIterator for any **AutoArray** (p. 233).*
- class **AutoBuffer**
- class **IndexedBuffer**
Wrap a memory buffer with an index.
- class **MutableIndexedBuffer**
- class **OrderedMap**
- class **OrderedMapConstIterator**
- class **OrderedMapIterator**
- struct **unique_if**
Define a type that is visible when T is not an array.
- struct **unique_if< T[]>**
Define a type that is visible when T is an unknown-bound array.
- struct **unique_if< T[S]>**
Define a type that is visible when T is a known-bound array.

Typedefs

- using **uint8Array** = **AutoArray**< uint8_t >
- using **uint16Array** = **AutoArray**< uint16_t >
- using **uint32Array** = **AutoArray**< uint32_t >

Functions

- template<typename T , typename... Ts>
unique_if< T >::unique_single **make_unique** (Ts &&... params)
Framework (p. 117) version of *std::make_unique* for non-array types.
- template<class T >
unique_if< T >::unique_array_unknown_bound **make_unique** (size_t size)
Framework (p. 117) version of *std::make_unique* for unknown-bound arrays.
- template<class T , class... Ts>
unique_if< T >::unique_array_known_bound **make_unique** (Ts &&...)=delete
Framework (p. 117) version of *std::make_unique* for known-bound arrays.
- bool **isLittleEndian** ()
Determine endianness of current platform.
- template<typename T >
bool **operator==** (const **AutoArray**< T > &lhs, const **AutoArray**< T > &rhs)
- template<typename T >
bool **operator!=** (const **AutoArray**< T > &lhs, const **AutoArray**< T > &rhs)
- template<typename T >
bool **operator<** (const **AutoArray**< T > &lhs, const **AutoArray**< T > &rhs)
- template<typename T >
bool **operator<=** (const **AutoArray**< T > &lhs, const **AutoArray**< T > &rhs)
- template<typename T >
bool **operator>** (const **AutoArray**< T > &lhs, const **AutoArray**< T > &rhs)
- template<typename T >
bool **operator>=** (const **AutoArray**< T > &lhs, const **AutoArray**< T > &rhs)

F.12.1 Detailed Description

Support for memory-related operations.

The **Memory** (p. 139) package contains templates and classes that are used to manage memory, auto-sizing arrays, for example.

F.12.2 Function Documentation

F.12.2.1 isLittleEndian()

```
bool BiometricEvaluation::Memory::isLittleEndian ( ) [inline]
```

Determine endianness of current platform.

Returns

true if current platform is little endian. false otherwise.

F.12.2.2 make_unique() [1/3]

```
template<typename T , typename... Ts>
unique_if<T>::unique_single BiometricEvaluation::Memory::make_unique (
    Ts &&... params )
```

Framework (p. 117) version of std::make_unique for non-array types.

Note

Coming in C++14. This implementation is taken from "Effective Modern C++" by Scott Meyers, modified to participate in the overload resolution only when T is not an array.

This function shall not participate in overload resolution unless T is not an array.

F.12.2.3 make_unique() [2/3]

```
template<class T >
unique_if<T>::unique_array_unknown_bound BiometricEvaluation::Memory::make_unique (
    size_t size )
```

Framework (p. 117) version of std::make_unique for unknown-bound arrays.

Note

Coming in C++14. This implementation is taken from the LLVM implementation.

This function shall not participate in overload resolution unless T is an array of unknown bound.

F.12.2.4 make_unique() [3/3]

```
template<class T , class... Ts>
unique_if<T>::unique_array_known_bound BiometricEvaluation::Memory::make_unique (
    Ts && ... ) [delete]
```

Framework (p. 117) version of std::make_unique for known-bound arrays.

Note

Coming in C++14. This implementation is taken from the LLVM implementation.

This function shall not participate in overload resolution unless T is an array of known bound.

F.12.2.5 operator"!=()

```
template<typename T >
bool BiometricEvaluation::Memory::operator!= (
    const AutoArray< T > & lhs,
    const AutoArray< T > & rhs )
```

Returns

Whether size or any accessible entries differ.

F.12.2.6 operator<()

```
template<typename T >
bool BiometricEvaluation::Memory::operator< (
    const AutoArray< T > & lhs,
    const AutoArray< T > & rhs )
```

Returns

Lexicographical comparison of accessible entries.

F.12.2.7 operator<=()

```
template<typename T >
bool BiometricEvaluation::Memory::operator<= (
    const AutoArray< T > & lhs,
    const AutoArray< T > & rhs )
```

Returns

Lexicographical comparison of accessible entries.

F.12.2.8 operator==(())

```
template<typename T >
bool BiometricEvaluation::Memory::operator==( (
    const AutoArray< T > & lhs,
    const AutoArray< T > & rhs )
```

Returns

Equivalence of all accessible entries and size.

F.12.2.9 operator>()

```
template<typename T >
bool BiometricEvaluation::Memory::operator> (
    const AutoArray< T > & lhs,
    const AutoArray< T > & rhs )
```

Returns

Lexicographical comparison of accessible entries.

F.12.2.10 operator>=()

```
template<typename T >
bool BiometricEvaluation::Memory::operator>= (
    const AutoArray< T > & lhs,
    const AutoArray< T > & rhs )
```


Returns

Lexicographical comparison of accessible entries.

F.13 BiometricEvaluation::Memory::AutoArrayUtility Namespace Reference

Functions

- `template<typename T, typename = typename std::enable_if<std::is_same<T, uint8_t>::value || std::is_same<T, char>::value>::type>`
`char * cstr (const AutoArray< T > &rahc)`
*Cast an **AutoArray** (p. 233) of uint8_t or char to a char*.*
- `template<typename T, typename = typename std::enable_if<std::is_same<T, uint8_t>::value || std::is_same<T, char>::value>::type>`
`std::string getString (const AutoArray< T > &aa, typename AutoArray< T >::size_type count)`
*Convert a uint8_t or char **AutoArray** (p. 233) to a string.*
- `template<typename T, typename = typename std::enable_if<std::is_same<T, uint8_t>::value || std::is_same<T, char>::value>::type>`
`void setString (AutoArray< T > &aa, const std::string &str)`
Copy a string into an AutoArray of uint8_t or char.
- `template<typename T, typename = typename std::enable_if<std::is_same<T, uint8_t>::value || std::is_same<T, char>::value>::type>`
`void setString (AutoArray< T > &aa, const char *str,...)`
Copy a string into an AutoArray of uint8_t or char.

F.13.1 Detailed Description

Convenience functions for AutoArrays.

F.13.2 Function Documentation

F.13.2.1 cstr()

```
template<typename T , typename = typename std::enable_if<std::is_same<T, uint8_t>::value || std::is_same<T, char>::value>::type>
char* BiometricEvaluation::Memory::AutoArrayUtility::cstr (
    const AutoArray< T > & rahc ) [inline]
    Cast an AutoArray (p. 233) of uint8_t or char to a char*.
```

Parameters

<i>rahc</i>	AutoArray (p. 233) to cast.
-------------	------------------------------------

Returns

rahc casted as a char*.

F.13.2.2 getString()

```
template<typename T , typename = typename std::enable_if<std::is_same<T, uint8_t>::value || std::is_same<T, char>::value>::type>
std::string BiometricEvaluation::Memory::AutoArrayUtility::getString (
    const AutoArray< T > & aa,
    typename AutoArray< T >::size_type count ) [inline]
    Convert a uint8_t or char AutoArray (p. 233) to a string.
```

Parameters

<i>aa</i>	AutoArray (p. 233) to stringify.
<i>count</i>	Last byte of aa to include in the returned string.

Returns

String representation of aa.

Exceptions

Error::MemoryError (p. 436)	count > aa.size()
------------------------------------	-------------------

F.13.2.3 setString() [1/2]

```
template<typename T , typename = typename std::enable_if<std::is_same<T, uint8_t>::value || std::is_same<T, char>::value>::type>
void BiometricEvaluation::Memory::AutoArrayUtility::setString (
    AutoArray< T > & aa,
    const std::string & str ) [inline]
    Copy a string into an AutoArray of uint8_t or char.
```

Parameters

<i>aa</i>	AutoArray (p. 233) whose contents will be replaced with str.
<i>str</i>	String to assign to AutoArray (p. 233).

F.13.2.4 setString() [2/2]

```
template<typename T , typename = typename std::enable_if<std::is_same<T, uint8_t>::value || std::is_same<T, char>::value>::type>
void BiometricEvaluation::Memory::AutoArrayUtility::setString (
    AutoArray< T > & aa,
    const char * str,
    ... ) [inline]
    Copy a string into an AutoArray of uint8_t or char.
```

Parameters

<i>aa</i>	AutoArray (p. 233) whose contents will be replaced with str.
<i>str</i>	printf-style format string.
...	Variable list of arguments for printf formatting.

F.14 BiometricEvaluation::MPI Namespace Reference

Common declarations and functions for the MPI-based functionality.

Classes

- class **CSVDistributor**
An implementation of the MPI::Distrbutor abstraction that distribute lines of a text file via work packages.
- class **CSVProcessor**
An implementation of a work package processor that will extract lines (and optionally tokenize) a line from a CSV text file.
- class **CSVResources**
- class **Distributor**
A class to represent an MPI (p. 145) task that distributes work to other tasks.
- class **Exception**
- class **Receiver**
*A class to represent an MPI (p. 145) task that receives WorkPackages containers from the **Distributor** (p. 305).*
- class **RecordProcessor**
*An implementation of a work package processor that will extract record store keys, and optionally, values, from a **WorkPackage** (p. 606).*
- class **RecordStoreDistributor**
An implementation of the MPI::Distrbutor abstraction that uses a record store for input to create the work packages.
- class **RecordStoreResources**
A class to represent a set of resources needed by an MPI (p. 145) program using a RecordStore for input.
- class **Resources**
- class **Runtime**
Runtime (p. 540) support for the startup/shutdown of MPI (p. 145) jobs.
- class **TerminateJob**
*An exception that when thrown from a Task should result in the entire job (all tasks) being shut down by the **Distributor** (p. 305).*
- class **WorkPackage**
A class to represent a piece of work to be acted upon by a processor.
- class **WorkPackageProcessor**
Represents an object that processes the contents of a work package.

Typedefs

- using **taskcmd_t** = std::underlying_type< **TaskCommand** >::type
- using **taskstat_t** = std::underlying_type< **TaskStatus** >::type
- using **msgtag_t** = std::underlying_type< **MessageTag** >::type

Enumerations

- enum **TaskCommand** : int32_t {
TaskCommand::Continue = 0, **TaskCommand::Ignore** = 1, **TaskCommand::Exit** = 2, **TaskCommand::QuickExit** = 3,
TaskCommand::TermExit = 4 }
- enum **TaskStatus** : int32_t { **TaskStatus::OK** = 0, **TaskStatus::Failed** = 1, **TaskStatus::Exit** = 2,
TaskStatus::RequestJobTermination = 3 }
- enum **MessageTag** : int32_t { **MessageTag::Control** = 0, **MessageTag::Data** = 1, **MessageTag::OOB** = 2 }

Functions

- std::string **generateUniqueID** ()
Obtain a unique ID for the current process.
- void **printStatus** (const std::string &message)
Print a status message to stdout.
- void **logEntry** (**IO::Logsheet** &logsheet)
Send the current log stream to the log device as a debug entry.
- void **logMessage** (**IO::Logsheet** &logsheet, const std::string &message)
Send a log message to the given Logsheet as a debug entry.
- std::shared_ptr< **BiometricEvaluation::IO::Logsheet** > **openLogsheet** (const std::string &url, const std::string &description)
*Open a Logsheet object for a component of the **MPI** (p. 145) framework.*

Variables

- bool **Exit**
- bool **QuickExit**
- bool **TermExit**
- bool **checkpointEnable**
- bool **doCheckpointRestore**

F.14.1 Detailed Description

Common declarations and functions for the MPI-based functionality.

F.14.2 Typedef Documentation

F.14.2.1 msgtag_t

using **BiometricEvaluation::MPI::msgtag_t** = typedef std::underlying_type< **MessageTag**>::type
Storage type for MessageTag.

F.14.2.2 taskcmd_t

using **BiometricEvaluation::MPI::taskcmd_t** = typedef std::underlying_type< **TaskCommand**>::type
Storage type for TaskCommand.

F.14.2.3 taskstat.t

using **BiometricEvaluation::MPI::taskstat.t** = typedef std::underlying_type< **TaskStatus**>::type
Storage type for TaskStatus.

F.14.3 Enumeration Type Documentation

F.14.3.1 MessageTag

enum **BiometricEvaluation::MPI::MessageTag** : int32_t [strong]
The types of messages sent between **MPI** (p. 145) task processes.

Enumerator

Control	A control message (start, exit, etc.)
Data	A data message.
OOB	An out-of-band message, used when the normal control/data messaging cannot be used.

F.14.3.2 TaskCommand

enum **BiometricEvaluation::MPI::TaskCommand** : int32_t [strong]
The command given to an **MPI** (p. 145) task.

Enumerator

Continue	Normal operation.
Ignore	Ignore the message.
Exit	Transition to the normal shutdown state.
QuickExit	Transition to the quick shutdown state.
TermExit	Transition to the immediate shutdown state.

F.14.3.3 TaskStatus

enum **BiometricEvaluation::MPI::TaskStatus** : int32_t [strong]
The status of an **MPI** (p. 145) distributor or receiver task.

Enumerator

OK	Normal operation.
Failed	Failed to complete an operation.
Exit	Transitioned to the shutdown state.
RequestJobTermination	Requesting that Distributor (p. 305) stops the job.

F.14.4 Function Documentation

F.14.4.1 generateUniqueID()

```
std::string BiometricEvaluation::MPI::generateUniqueID ( )
```

Obtain a unique ID for the current process.

The ID is a string that is based on the host name, **MPI** (p. 145) rank, and process ID, formatted in a manner that can be used to uniquely name files.

Returns

The unique ID for the process.

F.14.4.2 logEntry()

```
void BiometricEvaluation::MPI::logEntry (
    IO::Logsheet & logsheet )
```

Send the current log stream to the log device as a debug entry.

Log messages may be streamed into the Logsheet and written as debug messages to aid tracing. In order to prevent log errors interfering with the **MPI** (p. 145) job, errors are managed, and therefore, log messages may stop if the Logsheet has failed.

Parameters

in	<i>logsheet</i>	The open Logsheet to write into.
----	-----------------	----------------------------------

F.14.4.3 logMessage()

```
void BiometricEvaluation::MPI::logMessage (
    IO::Logsheet & logsheet,
    const std::string & message )
```

Send a log message to the given Logsheet as a debug entry.

In order to prevent log errors interfering with the **MPI** (p. 145) job, errors are managed, and therefore, log messages may stop if the Logsheet has failed.

Parameters

in	<i>logsheet</i>	The open Logsheet to write into.
in	<i>message</i>	The log message.

F.14.4.4 openLogsheet()

```
std::shared_ptr< BiometricEvaluation::IO::Logsheet> BiometricEvaluation::MPI::openLogsheet (
    const std::string & url,
    const std::string & description )
```

Open a Logsheet object for a component of the **MPI** (p. 145) framework.
 If the empty string is passed in as the URL, then a Null Logsheet object is returned.

Parameters

in	<i>url</i>	The Uniform Resource Locator for the Logsheet.
in	<i>description</i>	The description of the Logsheet.

Returns

Shared pointer to the Logsheet object.

Exceptions

Error::ParameterError (p. 474)	Invalid URL.
Error::Exception (p. 310)	Failed to create the Logsheet object. The exception string will contain more information.

F.14.4.5 printStatus()

```
void BiometricEvaluation::MPI::printStatus (
    const std::string & message )
    Print a status message to stdout.
```

Parameters

in	<i>message</i>	The message to be printed.
----	----------------	----------------------------

F.15 BiometricEvaluation::Palm Namespace Reference

Biometric information relating to palm images and derived information.

Classes

- class **AN2KView**
*A class to represent a single **Palm** (p. 149) view and derived information.*

Enumerations

- enum **Position** {
Unknown = 20, **RightFull** = 21, **RightWriters** = 22, **LeftFull** = 23,
LeftWriters = 24, **RightLower** = 25, **RightUpper** = 26, **LeftLower** = 27,
LeftUpper = 28, **RightOther** = 29, **LeftOther** = 30, **RightInterdigital** = 31,
RightThenar = 32, **RightHypothenar** = 33, **LeftInterdigital** = 34, **LeftThenar** = 35,
LeftHypothenar = 36, **RightGrasp** = 37, **LeftGrasp** = 38, **RightCarpelDelta** = 81,

LeftCarpelDelta = 82, **RightFullWithWriters** = 83, **LeftFullWithWriters** = 84, **RightWristBracelet** = 85,
LeftWristBracelet = 86 }

Palm position codes.

F.15.1 Detailed Description

Biometric information relating to palm images and derived information.

The **Palm** (p. 149) package gathers all palm related matters,

F.15.2 Enumeration Type Documentation

F.15.2.1 Position

```
enum BiometricEvaluation::Palm::Position [strong]
```

Palm (p. 149) position codes.

These codes match those in ANSI/NIST. Other data formats may have to map codes into this set.

F.16 BiometricEvaluation::Plantar Namespace Reference

Biometric information relating to plantar images and derived information.

Enumerations

- enum **Position** {
UnknownSole = 60, **RightSole** = 61, **LeftSole** = 62, **UnknownToe** = 63,
RightBigToe = 64 }

Plantar position codes.

F.16.1 Detailed Description

Biometric information relating to plantar images and derived information.

F.16.2 Enumeration Type Documentation

F.16.2.1 Position

```
enum BiometricEvaluation::Plantar::Position [strong]
```

Plantar (p. 150) position codes.

These codes match those in ANSI/NIST. Other minutiae formats may have to map codes into this set.

F.17 BiometricEvaluation::Process Namespace Reference

Process (p. 150) information and controls.

Classes

- class **CommandCenter**
- class **CommandParser**
- class **ForkManager**
 - Manager (p. 430) implementation that starts Workers by calling fork(2).*
- class **ForkWorkerController**
 - Wrapper of a Worker (p. 595) returned from a Process::ForkManager (p. 335).*
- class **Manager**
 - An interface for intranode process management classes.*
- class **MessageCenter**
- class **MessageCenterListener**
- class **MessageCenterReceiver**
 - Receives message from a client, forwarding to the central MessageCenter (p. 436).*
- class **POSIXThreadManager**
 - Manager (p. 430) implementation that starts Workers in POSIX threads.*
- class **POSIXThreadWorkerController**
 - Decorated Worker (p. 595) returned from a Process::POSIXThreadManager (p. 480).*
- class **Semaphore**
 - Represent a semaphore that can be used for interprocess communication.*
- class **Statistics**
 - The Statistics (p. 560) class provides an interface for gathering process statistics, such as memory usage, system time, etc.*
- class **Worker**
 - An abstraction of an instance that performs work on given data.*
- class **WorkerController**
 - Wrapper of a Worker (p. 595) returned from a Process::Manager (p. 430).*

Typedefs

- using **ParameterList** = std::map< std::string, std::shared_ptr< void > >

F.17.1 Detailed Description

Process (p. 150) information and controls.

The **Process** (p. 150) package gathers all process related matters, including a class to obtain resource usage statistics.

F.17.2 Typedef Documentation

F.17.2.1 ParameterList

```
using BiometricEvaluation::Process::ParameterList = typedef std::map<std::string, std::shared_ptr<void> >
```

Convenience alias for parameter lists to child routines

F.18 BiometricEvaluation::System Namespace Reference

Operating system, hardware, etc.

Functions

- `uint32_t getCPUCount ()`
Obtain the number of central processing units that are online. Typically, this is the total logical CPU count for the system, often called hyperthreads.
- `uint32_t getCPUCoreCount ()`
Obtain the number of CPU cores that are online.
- `uint32_t getCPUSocketCount ()`
Obtain the number of CPU sockets that are online.
- `uint64_t getRealMemorySize ()`
Obtain the amount of real memory in the system.
- `double getLoadAverage ()`
Obtain the system load average for the last minute.

F.18.1 Detailed Description

Operating system, hardware, etc.

The **System** (p. 152) package gathers all system related matters, such as the operating system name, number of CPUs, etc.

F.18.2 Function Documentation

F.18.2.1 getCPUCoreCount()

`uint32_t BiometricEvaluation::System::getCPUCoreCount ()`
Obtain the number of CPU cores that are online.
Obtain the number of central processing units that are online. This is the total CPU core count for the system.

Returns

The number of CPU cores.

Exceptions

<i>Error::NotImplemented</i> (p. 456)	Not implemented for this operating system, or the underlying OS feature is not installed.
---------------------------------------	---

F.18.2.2 getCPUCount()

`uint32_t BiometricEvaluation::System::getCPUCount ()`
Obtain the number of central processing units that are online. Typically, this is the total logical CPU count for the system, often called hyperthreads.

Returns

The number of processing units.

Exceptions

Error::NotImplemented (p. 456)	Not implemented for this operating system, or the underlying OS feature is not installed.
---------------------------------------	---

F.18.2.3 getCPUSocketCount()

```
uint32_t BiometricEvaluation::System::getCPUSocketCount ( )
```

Obtain the number of CPU sockets that are online.

The hierarchy is CPU (thread) -> Core -> Socket, where there are 1..n hyperthreads per core and 1..n cores per socket.

Returns

The number of CPU sockets.

Exceptions

Error::NotImplemented (p. 456)	Not implemented for this operating system, or the underlying OS feature is not installed.
---------------------------------------	---

F.18.2.4 getLoadAverage()

```
double BiometricEvaluation::System::getLoadAverage ( )
```

Obtain the system load average for the last minute.

Returns

The system load average.

Exceptions

Error::NotImplemented (p. 456)	Not implemented for this operating system, or the underlying OS feature is not installed.
---------------------------------------	---

F.18.2.5 getRealMemorySize()

```
uint64_t BiometricEvaluation::System::getRealMemorySize ( )
```

Obtain the amount of real memory in the system.

Returns

The real memory size, in kibibytes.

Exceptions

Error::NotImplemented (p. 456)	Not implemented for this operating system, or the underlying OS feature is not installed.
---------------------------------------	---

F.19 BiometricEvaluation::Text Namespace Reference

Text (p. 154) processing for string objects.

Functions

- `std::string trimWhitespace` (const `std::string` &s, const `std::locale` &locale=`std::locale()`)
Remove leading and trailing whitespace from a string.
- `std::string ltrimWhitespace` (const `std::string` &s, const `std::locale` &locale=`std::locale()`)
Remove leading whitespace from a string.
- `std::string rtrimWhitespace` (const `std::string` &s, const `std::locale` &locale=`std::locale()`)
Remove trailing whitespace from a string.
- `std::string trim` (const `std::string` &s, const `char` trimChar)
Remove leading and trailing characters from a string.
- `std::string ltrim` (const `std::string` &s, const `char` trimChar)
Remove leading characters from a string.
- `std::string rtrim` (const `std::string` &s, const `char` trimChar)
Remove trailing characters from a string.
- `std::string digest` (const `std::string` &s, const `std::string` &digest="md5")
Compute the digest of a string.
- `std::string digest` (const `void` *buffer, const `size_t` buffer_size, const `std::string` &digest="md5")
Compute the digest of a memory buffer.
- `std::vector< std::string > split` (const `std::string` &str, const `char` delimiter, bool escape=true)
Return tokens bound by delimiters and the beginning and end of a string.
- `std::string basename` (const `std::string` &path)
Extract the filename component of a pathname.
- `std::string dirname` (const `std::string` &path)
Extract the directory component of a pathname.
- `bool caseInsensitiveCompare` (const `std::string` &str1, const `std::string` &str2)
Compare two ASCII-encoded strings.
- `std::string toUppercase` (const `std::string` &str, const `std::locale` &locale=`std::locale()`)
Uppercase a string, respecting locale.
- `std::string toLowercase` (const `std::string` &str, const `std::locale` &locale=`std::locale()`)
Lowercase a string, respecting locale.
- `std::string encodeBase64` (const **BiometricEvaluation::Memory::uint8Array** &data)
Perform Base64 encoding.
- **BiometricEvaluation::Memory::uint8Array** `decodeBase64` (const `std::string` &data)
Perform Base64 decoding.

F.19.1 Detailed Description

Text (p. 154) processing for string objects.

The **Text** (p. 154) package contains a set of functions for the processing of strings: removing leading and trailing whitespace, computing a digest, and other utility functions.

F.19.2 Function Documentation

F.19.2.1 basename()

```
std::string BiometricEvaluation::Text::basename (
    const std::string & path )
```

Extract the filename component of a pathname.

Returns the component following the final '/'. Trailing '/' characters are not counted as part of the path-name.

Parameters

in	<i>path</i>	Path from which to extract the filename portion.
----	-------------	--

Returns

Filename portion of path.

F.19.2.2 caseInsensitiveCompare()

```
bool BiometricEvaluation::Text::caseInsensitiveCompare (
    const std::string & str1,
    const std::string & str2 )
```

Compare two ASCII-encoded strings.

Parameters

<i>str1</i>	First string to compare.
<i>str2</i>	Second string to compare.

Returns

true if str1 and str2 are equal other than case, false otherwise.

F.19.2.3 decodeBase64()

```
BiometricEvaluation::Memory::uint8Array BiometricEvaluation::Text::decodeBase64 (
    const std::string & data )
```

Perform Base64 decoding.

Parameters

<i>data</i>	Base64 data to decode.
-------------	------------------------

Returns

Base64 decoding of data.

F.19.2.4 digest() [1/2]

```
std::string BiometricEvaluation::Text::digest (
    const std::string & s,
    const std::string & digest = "md5" )
```

Compute the digest of a string.

Parameters

in	<i>s</i>	The string of which a digest should be computed.
in	<i>digest</i>	The digest to use. Any digest supported by OpenSSL is valid, and the default is MD5.

Exceptions

<i>Error::MemoryError</i> (p. 436)	Could not allocate memory to store digest.
<i>Error::NotImplemented</i> (p. 456)	The value of digest is not a supported digest.
<i>Error::StrategyError</i> (p. 567)	An error occurred while obtaining the digest.

Returns

An ASCII representation of the hex digits composing the digest.

F.19.2.5 digest() [2/2]

```
std::string BiometricEvaluation::Text::digest (
    const void * buffer,
    const size_t buffer_size,
    const std::string & digest = "md5" )
```

Compute the digest of a memory buffer.

Parameters

in	<i>buffer</i>	The buffer of which a digest should be computed.
in	<i>buffer_size</i>	The size of buffer.
in	<i>digest</i>	The digest to use. Any digest supported by OpenSSL is valid, and the default is MD5.

Exceptions

Error::MemoryError (p. 436)	Could not allocate memory to store digest.
Error::NotImplemented (p. 456)	The value of digest is not a supported digest.
Error::StrategyError (p. 567)	An error occurred while obtaining the digest.

Returns

An ASCII representation of the hex digits composing the digest.

F.19.2.6 `dirname()`

```
std::string BiometricEvaluation::Text::dirname (  
    const std::string & path )
```

Extract the directory component of a pathname.

Returns the string up to, but not including, the final '/'.

Parameters

<i>in</i>	<i>path</i>	Path from which to extract the directory portion.
-----------	-------------	---

Returns

Directory portion of path.

F.19.2.7 `encodeBase64()`

```
std::string BiometricEvaluation::Text::encodeBase64 (  
    const BiometricEvaluation::Memory::uint8Array & data )
```

Perform Base64 encoding.

Parameters

<i>data</i>	Data to encoded.
-------------	------------------

Returns

Base64 encoding of data.

F.19.2.8 `ltrim()`

```
std::string BiometricEvaluation::Text::ltrim (  
    const std::string & s,  
    const char trimChar )
```

Remove leading characters from a string.

Parameters

<i>s</i>	String object whose leading trimChar should be removed.
<i>trimChar</i>	Character to remove from the beginning of s.

Returns

Copy of s without leading trimChar.

F.19.2.9 ltrimWhitespace()

```
std::string BiometricEvaluation::Text::ltrimWhitespace (
    const std::string & s,
    const std::locale & locale = std::locale() )
```

Remove leading whitespace from a string.

Parameters

<i>s</i>	String object whose leading whitespace should be removed.
<i>locale</i>	Locale to be considered when determining whitespace characters.

Returns

Copy of s without leading whitespace.

F.19.2.10 rtrim()

```
std::string BiometricEvaluation::Text::rtrim (
    const std::string & s,
    const char trimChar )
```

Remove trailing characters from a string.

Parameters

<i>s</i>	String object whose trailing trimChar should be removed.
<i>trimChar</i>	Character to remove from the end of s.

Returns

Copy of s without trailing trimChar.

F.19.2.11 rtrimWhitespace()

```
std::string BiometricEvaluation::Text::rtrimWhitespace (
    const std::string & s,
    const std::locale & locale = std::locale() )
```


Remove trailing whitespace from a string.

Parameters

<i>s</i>	String object whose trailing whitespace should be removed.
<i>locale</i>	Locale to be considered when determining whitespace characters.

Returns

Copy of *s* without trailing whitespace.

F.19.2.12 split()

```
std::vector<std::string> BiometricEvaluation::Text::split (
    const std::string & str,
    const char delimiter,
    bool escape = true )
```

Return tokens bound by delimiters and the beginning and end of a string.

Parameters

in	<i>str</i>	String to tokenize.
in	<i>delimiter</i>	Character that defines the end of a token. Any are valid, except '\ '.
in	<i>escape</i>	If the delimiter is prefixed with '\ ' in the string, do not split at that point and remove the '\ '.

Returns

Vector of string tokens, in order of appearance.

Note

If delimiter does not appear in string, the returned vector *vector* will still contain one item, *str*.

F.19.2.13 toLowercase()

```
std::string BiometricEvaluation::Text::toLowercase (
    const std::string & str,
    const std::locale & locale = std::locale() )
```

Lowercase a string, respecting locale.

Parameters

<i>str</i>	String to lowercase.
<i>locale</i>	Locale to use when lowercasing <i>str</i> .

Returns

Lowercase copy of str.

F.19.2.14 toUppercase()

```
std::string BiometricEvaluation::Text::toUppercase (
    const std::string & str,
    const std::locale & locale = std::locale() )
```

Uppercase a string, respecting locale.

Parameters

<i>str</i>	String to uppercase.
<i>locale</i>	Locale to use when uppercasing str.

Returns

Uppercase copy of str.

F.19.2.15 trim()

```
std::string BiometricEvaluation::Text::trim (
    const std::string & s,
    const char trimChar )
```

Remove leading and trailing characters from a string.

Parameters

<i>s</i>	String object whose leading and trailing trimChar should be removed.
<i>trimChar</i>	Character to remove from the beginning and ending of s.

Returns

Copy of s without leading or trailing trimChar.

F.19.2.16 trimWhitespace()

```
std::string BiometricEvaluation::Text::trimWhitespace (
    const std::string & s,
    const std::locale & locale = std::locale() )
```

Remove leading and trailing whitespace from a string.

Parameters

<i>s</i>	String object whose leading and trailing whitespace should be removed.
<i>locale</i>	Locale to be considered when determining whitespace characters.

Returns

Copy of s without leading or trailing whitespace.

F.20 BiometricEvaluation::Time Namespace Reference

Support for time and timers.

Classes

- class **Timer**

This class can be used by applications to report the amount of time a block of code takes to execute.

- class **Watchdog**

*A **Watchdog** (p. 591) object can be used by applications to limit the amount of processing time taken by a block of code.*

Functions

- std::string **getCurrentTime** ()
- std::string **getCurrentDate** ()
- std::string **getCurrentDateAndTime** ()
- std::string **getCurrentCalendarInformation** (const std::string &formatString)

Obtain customized calendar information.

- std::string **put_time** (const struct tm *tmb, const char *fmt)

Manual implementation of std::put_time.

- std::ostream & **operator<<** (std::ostream &s, const **Timer** &timer)

*Output stream operator overload for **Timer** (p. 578).*

- void **WatchdogSignalHandler** (int signo, siginfo_t *info, void *uap)

Variables

- const uint64_t **OneSecond** = 1000000
- const uint64_t **OneHalfSecond** = 500000
- const uint64_t **OneQuarterSecond** = 250000
- const uint64_t **OneEighthSecond** = 125000
- const int **NanosecondsPerMicrosecond** = 1000
- const int **MicrosecondsPerSecond** = 1000000
- const int **MicrosecondsPerMillisecond** = 1000
- const int **MillisecondsPerSecond** = 1000

F.20.1 Detailed Description

Support for time and timers.

The **Time** (p. 161) package gathers all timing relating matters, such as Timers, **Watchdog** (p. 591) timers, etc. **Time** (p. 161) values are in microsecond units.

F.20.2 Function Documentation

F.20.2.1 getCurrentCalendarInformation()

```
std::string BiometricEvaluation::Time::getCurrentCalendarInformation (
    const std::string & formatString )
```

Obtain customized calendar information.

Parameters

<i>formatString</i>	A C++11 <code>put_time</code> -compatible format string.
---------------------	--

Returns

The current calendar information formatted as specified in `formatString`.

Note

Return value is undefined if format string is invalid.

F.20.2.2 getCurrentDate()

```
std::string BiometricEvaluation::Time::getCurrentDate ( )
```

Returns

The current ISO 8601 date as a string.

F.20.2.3 getCurrentDateAndTime()

```
std::string BiometricEvaluation::Time::getCurrentDateAndTime ( )
```

Returns

The standard locale current date and time as a string.

F.20.2.4 getCurrentTime()

```
std::string BiometricEvaluation::Time::getCurrentTime ( )
```

Returns

The current ISO 8601 time as a string.

F.20.2.5 operator<<()

```
std::ostream& BiometricEvaluation::Time::operator<< (
    std::ostream & s,
    const Timer & timer )
```

Output stream operator overload for **Timer** (p. 578).

Parameters

<i>s</i>	Stream to append.
<i>timer</i>	Timer (p. 578) whose elapsed time in microseconds should be appended to s.

Returns

s with value of elapsedStr() appended.

Exceptions

<i>BE::Error::StrategyError</i>	Propagated from elapsedStr().
---------------------------------	-------------------------------

F.20.2.6 put_time()

```
std::string BiometricEvaluation::Time::put_time (
    const struct tm * tmb,
    const char * fmt )
```

Manual implementation of std::put_time.

Note

Exists because g++ does not currently implement put_time (http://gcc.gnu.org/bugzilla/show_bug.cgi?id=54354)

F.21 BiometricEvaluation::Video Namespace Reference

Basic information relating to video and streams.

Classes

- class **Container**
Representation of a video container.
- struct **Frame**
- class **Stream**

Enumerations

- enum **CodingFormat** {
 None = 0, **MPEG1** = 1, **MPEG2** = 2, **MPEG4** = 3,
 H264 = 4 }
- enum **ContainerFormat** { **MPEG1PS** = 1, **MPEG2TS** = 2, **MPEG4PS** = 3, **AVI** = 4 }

F.21.1 Detailed Description

Basic information relating to video and streams.

Common representation of a video stream. **Stream** (p. 567) objects can only be obtained from **Container** (p. 281) objects.

The **Video** (p. 163) package gathers all video related matters, including classes to represent a video stream and video containers.

F.21.2 Enumeration Type Documentation

F.21.2.1 CodingFormat

enum **BiometricEvaluation::Video::CodingFormat** [strong]
Video (p. 163) coding formats.

F.21.2.2 ContainerFormat

enum **BiometricEvaluation::Video::ContainerFormat** [strong]
Container (p. 281) formats

F.22 BiometricEvaluation::View Namespace Reference

View (p. 587) information.

Classes

- class **AN2KView**
A class to represent single biometric view and derived information.
- class **AN2KViewVariableResolution**
A class to represent single view based on an ANSI/NIST record.
- class **View**
A class to represent single biometric element view.

Functions

- `std::ostream & operator<< (std::ostream &stream, const AN2KView::DeviceMonitoringMode &kind)`
Output stream overload for DeviceMonitoringMode.
- `std::ostream & operator<< (std::ostream &s, const AN2KViewVariableResolution::AN2KQualityMetric &qm)`
Output stream overload for AN2KQualityMetric.
- `std::ostream & operator<< (std::ostream &stream, const AN2KViewVariableResolution::PrintPositionCoordinate &ppc)`
Output stream overload for PrintPositionCoordinate.

F.22.1 Detailed Description

View (p. 587) information.

The **View** (p. 587) package gathers all classes and other items that are related to a biometric view, which represents an image and all information derived from that image, such as fingerprint minutiae.

F.22.2 Function Documentation

F.22.2.1 operator<<() [1/2]

```
std::ostream& BiometricEvaluation::View::operator<< (
    std::ostream & s,
    const AN2KViewVariableResolution::AN2KQualityMetric & qm )
```

Output stream overload for AN2KQualityMetric.

Parameters

in	<i>s</i>	Stream on which to append formatted AN2KQualityMetric information.
in	<i>qm</i>	AN2KQualityMetric information to append to stream.

Returns

stream with a qm textual representation appended.

F.22.2.2 operator<<() [2/2]

```
std::ostream& BiometricEvaluation::View::operator<< (
    std::ostream & stream,
    const AN2KViewVariableResolution::PrintPositionCoordinate & ppc )
```

Output stream overload for PrintPositionCoordinate.

Parameters

in	<i>stream</i>	Stream on which to append formatted PrintPositionCoordinate information.
in	<i>ppc</i>	PrintPositionCoordinate information to append to stream.

Returns

Stream with a ppc textual representation appended.

Appendix G

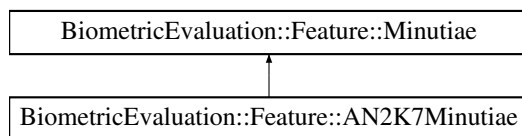
Class Documentation

G.1 BiometricEvaluation::Feature::AN2K7Minutiae Class Reference

A class to represent a set of minutiae in an ANSI/NIST record.

```
#include <be_feature_an2k7minutiae.h>
```

Inheritance diagram for BiometricEvaluation::Feature::AN2K7Minutiae:



Classes

- struct **FingerprintReadingSystem**
Representation of information about a fingerprint reader system.
- class **PatternClassification**
Pattern classification codes.

Public Types

- enum **EncodingMethod** { **EncodingMethod::Automatic** = 0, **EncodingMethod::AutomaticUnedited**, **EncodingMethod::AutomaticEdited**, **Manual** }
Methods for encoding minutiae data in an AN2K record.
- using **PatternClassificationSet** = std::vector< **PatternClassification::Entry** >
- using **FingerprintReadingSystem** = struct **FingerprintReadingSystem**

Public Member Functions

- **AN2K7Minutiae** (const std::string &filename, int recordNumber)
*Construct an AN2K7 **Minutiae** (p. 442) object from file data.*
- **AN2K7Minutiae** (**Memory::uint8Array** &buf, int recordNumber)
*Construct an AN2K7 **Minutiae** (p. 442) object from data contained in a memory buffer.*
- **PatternClassificationSet** **getPatternClassificationSet** () const

- Obtain the set fingerprint pattern classifications.*
- **FingerprintReadingSystem** **getOriginatingFingerprintReadingSystem** () const
- **MinutiaeFormat** **getFormat** () const
 - Obtain the minutiae format kind.*
- **MinutiaPointSet** **getMinutiaPoints** () const
 - Obtain the set of finger minutiae data points. The set may be empty.*
- **RidgeCountItemSet** **getRidgeCountItems** () const
 - Obtain the set of ridge count data items. The set may be empty.*
- **CorePointSet** **getCores** () const
 - Obtains the set of core positions. The set may be empty.*
- **DeltaPointSet** **getDeltas** () const
 - Obtains the set of delta positions. The set may be empty.*

Static Public Member Functions

- static **Finger::PatternClassification** **convertPatternClassification** (const char *fpc)
 - Convert string read from AN2K record into a **PatternClassification** (p. 474).*
- static **Finger::PatternClassification** **convertPatternClassification** (const **PatternClassification::↔Entry** &entry)
 - Convert a standard **PatternClassification::Entry** (p. 309) to a **PatternClassification::Kind**.*
- static **EncodingMethod** **convertEncodingMethod** (const char *mem)
 - Convert string read from AN2K record into a **EncodingMethod**.*
- static **Image::Coordinate** **convertCoordinate** (const char *str, bool calculateDistance=true)
 - Obtain a **Coordinate** given an AN2K entry.*

G.1.1 Detailed Description

A class to represent a set of minutiae in an ANSI/NIST record.

Each minutiae point, ridge count item, core, and delta is represented in the native ANSI/NIST format.

G.1.2 Member Enumeration Documentation

G.1.2.1 EncodingMethod

enum **BiometricEvaluation::Feature::AN2K7Minutiae::EncodingMethod** [strong]

Methods for encoding minutiae data in an AN2K record.

Enumerator

Automatic	No possible human interaction
AutomaticUnedited	Editing possible, but not performed
AutomaticEdited	Editing possible and was performed

G.1.3 Constructor & Destructor Documentation

G.1.3.1 AN2K7Minutiae() [1/2]

```
BiometricEvaluation::Feature::AN2K7Minutiae::AN2K7Minutiae (
    const std::string & filename,
    int recordNumber )
```

Construct an AN2K7 **Minutiae** (p. 442) object from file data.

The file contains a complete ANSI/NIST record, and an object of this class represents a single fingerprint minutiae record.

Parameters

in	<i>filename</i>	The name of the file containing the complete ANSI/NIST record.
in	<i>recordNumber</i>	Which fingerprint minutiae record to read from the complete AN2K record.

Exceptions

Error::FileError (p. 316)	An error occurred when opening or reading from the file.
Error::DataError (p. 295)	An error occurred reading the AN2K record, or there is no fingerprint minutiae record for the requested number.

G.1.3.2 AN2K7Minutiae() [2/2]

```
BiometricEvaluation::Feature::AN2K7Minutiae::AN2K7Minutiae (
    Memory::uint8Array & buf,
    int recordNumber )
```

Construct an AN2K7 **Minutiae** (p. 442) object from data contained in a memory buffer.

The buffer contains a complete ANSI/NIST record, and an object of this class represents a single fingerprint minutiae record.

Parameters

in	<i>buf</i>	The memory buffer containing the complete ANSI/NIST record.
in	<i>recordNumber</i>	Which fingerprint minutiae record to read from the complete AN2K record.

Exceptions

Error::DataError (p. 295)	An error occurred reading the AN2K record, or there is no fingerprint minutiae record for the requested number.
----------------------------------	---

G.1.4 Member Function Documentation

G.1.4.1 convertCoordinate()

```
static Image::Coordinate BiometricEvaluation::Feature::AN2K7Minutiae::convertCoordinate (
    const char * str,
    bool calculateDistance = true ) [static]
```

Obtain a **Coordinate** given an AN2K entry.

This AN2K entry is formatted as "XXXXYYYYY".

Parameters

in	<i>str</i>	Coordinate string from an AN2K record.
in	<i>calculateDistance</i>	Whether or not to calculate the [xy]Distance portion of the Coordinate.

Returns

Image::Coordinate (p. 284) representation of *str*.

Exceptions

Error::DataError (p. 295)	Invalid format of <i>str</i> .
----------------------------------	--------------------------------

G.1.4.2 convertEncodingMethod()

```
static EncodingMethod BiometricEvaluation::Feature::AN2K7Minutiae::convertEncodingMethod (
    const char * mem ) [static]
```

Convert string read from AN2K record into a **EncodingMethod**.

Parameters

in	<i>mem</i>	Value for minutiae encoding method read from AN2K record.
----	------------	---

Exceptions

Error::DataError (p. 295)	Invalid value for <i>mem</i> .
----------------------------------	--------------------------------

G.1.4.3 convertPatternClassification() [1/2]

```
static Finger::PatternClassification BiometricEvaluation::Feature::AN2K7Minutiae::convert↵
PatternClassification (
    const char * fpc ) [static]
```

Convert string read from AN2K record into a **PatternClassification** (p. 474).

Parameters

in	<i>fpc</i>	Value for pattern classification read from AN2K record.
----	------------	---

Exceptions

Error::DataError (p. 295)	Invalid value for fpc.
----------------------------------	------------------------

G.1.4.4 convertPatternClassification() [2/2]

```
static Finger::PatternClassification BiometricEvaluation::Feature::AN2K7Minutiae::convert←
PatternClassification (
    const PatternClassification::Entry & entry ) [static]
    Convert a standard PatternClassification::Entry (p. 309) to a PatternClassification::Kind.
```

Parameters

in	entry	A standard pattern classification entry
----	-------	---

Exceptions

Error::DataError (p. 295)	Non-standard pattern classification entry.
----------------------------------	--

G.1.4.5 getOriginatingFingerprintReadingSystem()

```
FingerprintReadingSystem BiometricEvaluation::Feature::AN2K7Minutiae::getOriginatingFingerprint←
ReadingSystem ( ) const
    Obtain the originating fingerprint reading system.
```

Exceptions

Error::ObjectDoesNotExist (p. 457)	The optional OFR field has been excluded.
---	---

G.1.4.6 getPatternClassificationSet()

```
PatternClassificationSet BiometricEvaluation::Feature::AN2K7Minutiae::getPatternClassification←
Set ( ) const
    Obtain the set fingerprint pattern classifications.
    The code returned may be a standard code or user-defined. Applications should call isPatternClassification←
Standard() to check.
```

G.2 BiometricEvaluation::Finger::AN2KMinutiaeDataRecord Class Reference

Representation of a Type-9 Record from an AN2K file.

```
#include <be_finger_an2kminutiae_data_record.h>
```

Public Member Functions

- **AN2KMinutiaeDataRecord** (const std::string &filename, int recordNumber)
*Construct an **AN2KMinutiaeDataRecord** (p. 171) object from data contained in a file on disk.*
- **AN2KMinutiaeDataRecord** (**Memory::uint8Array** &buf, int recordNumber)
*Construct an **AN2KMinutiaeDataRecord** (p. 171) object from data contained in a memory buffer.*
- std::shared_ptr< **Feature::AN2K7Minutiae** > **getAN2K7Minutiae** () const
Obtain the "standard" minutiae data from this Type-9 Record (fields 9.005 - 9.012).
- std::shared_ptr< **Feature::AN2K11EFS::ExtendedFeatureSet** > **getAN2K11EFS** () const
Obtain the extended feature set data from this Type-9 Record (fields 9.300 - 9.399).
- **Impression** **getImpressionType** () const
Return impression type field from Type-9 Record.
- std::map< uint16_t, **Memory::uint8Array** > **getRegisteredVendorBlock** (**Feature::Minutiae**↔
Format vendor) const
Obtain data recorded in a registered vendor minutiae block found in this Type-9 Record.

G.2.1 Detailed Description

Representation of a Type-9 Record from an AN2K file.

Type-9 Records may contain only "standard" minutiae data (fields 9.005 - 9.012) or any combination of "standard" minutiae data, registered vendor minutiae data (several vendors from fields 9.013 - 9.175), and extended feature set data (fields 9.300 - 9.399), although not all fields are supported.

G.2.2 Constructor & Destructor Documentation

G.2.2.1 AN2KMinutiaeDataRecord() [1/2]

```
BiometricEvaluation::Finger::AN2KMinutiaeDataRecord::AN2KMinutiaeDataRecord (
    const std::string & filename,
    int recordNumber )
```

Construct an **AN2KMinutiaeDataRecord** (p. 171) object from data contained in a file on disk.

The file contains a complete ANSI/NIST record, and an object of this class represents a single fingerprint minutiae record.

Parameters

in	<i>filename</i>	The name of the file containing the complete ANSI/NIST record.
in	<i>recordNumber</i>	Which fingerprint minutiae record to read from the complete AN2K record.

Exceptions

Error::FileError (p. 316)	An error occurred when opening or reading from the file.
Error::DataError (p. 295)	An error occurred reading the AN2K record, or there is no fingerprint minutiae record for the requested number.

G.2.2.2 AN2KMinutiaeDataRecord() [2/2]

```
BiometricEvaluation::Finger::AN2KMinutiaeDataRecord::AN2KMinutiaeDataRecord (
    Memory::uint8Array & buf,
    int recordNumber )
```

Construct an **AN2KMinutiaeDataRecord** (p. 171) object from data contained in a memory buffer.

The buffer contains a complete ANSI/NIST record, and an object of this class represents a single fingerprint minutiae record.

Parameters

in	<i>buf</i>	The memory buffer containing the complete ANSI/NIST record.
in	<i>recordNumber</i>	Which fingerprint minutiae record to read from the complete AN2K record.

Exceptions

Error::DataError (p. 295)	An error occurred reading the AN2K record, or there is no fingerprint minutiae record for the requested number.
----------------------------------	---

G.2.3 Member Function Documentation

G.2.3.1 getAN2K11EFS()

```
std::shared_ptr< Feature::AN2K11EFS::ExtendedFeatureSet> BiometricEvaluation::Finger::AN2K↔
MinutiaeDataRecord::getAN2K11EFS ( ) const
```

Obtain the extended feature set data from this Type-9 Record (fields 9.300 - 9.399).

Returns

Shared pointer to an AN2K11ExtendedFeatureSet object if present in the record. The managed pointer will nulptr if there is no extended feature data.

G.2.3.2 getAN2K7Minutiae()

```
std::shared_ptr< Feature::AN2K7Minutiae> BiometricEvaluation::Finger::AN2KMinutiaeDataRecord↔
::getAN2K7Minutiae ( ) const
```

Obtain the "standard" minutiae data from this Type-9 Record (fields 9.005 - 9.012).

Returns

Shared pointer to an AN2KMinutiae object containing the standard format minutiae data found in this Type-9 Record.

G.2.3.3 getImpressionType()

Impression BiometricEvaluation::Finger::AN2KMinutiaeDataRecord::getImpressionType () const
Return impression type field from Type-9 Record.

Returns

Impression type of the image from which minutiae points were generated.

G.2.3.4 getRegisteredVendorBlock()

std::map<uint16_t, **Memory::uint8Array**> BiometricEvaluation::Finger::AN2KMinutiaeDataRecord↵
::getRegisteredVendorBlock (
 Feature::MinutiaeFormat vendor) const

Obtain data recorded in a registered vendor minutiae block found in this Type-9 Record.

Parameters

in	vendor	The vendor whose registered minutiae blocks are being requested.
----	--------	--

Returns

A map of the registered vendor minutiae block fields. The map key is the AN2K Field number. The value is a uint8Array of the ASCII data found at that field. All Fields will be present as keys even if there was no data recorded in that Field.

Exceptions

Error::NotImplemented (p. 456)	Cannot return a map of fields for vendor, likely because there exists a better, native implementation of accessing minutiae data in AN2KMinutiaeDataRecord (p. 171).
---------------------------------------	---

G.3 BiometricEvaluation::View::AN2KViewVariableResolution::AN2↵ KQualityMetric Struct Reference

A structure to represent an AN2K quality metric.

```
#include <be_view_an2kview_varres.h>
```

Public Attributes

- **Feature::FGP** fgp
- uint8_t score
- uint16_t vendorID
- uint16_t productCode

G.3.1 Detailed Description

A structure to represent an AN2K quality metric.

The quality metric is an optional field in the Type-13 (Latent), Type-14 (Fingerprint and Segmentation) and Type-15 (Palmprint). The NIST Quality Metric is also returned via this structure.

G.4 BiometricEvaluation::DataInterchange::AN2KRecord Class Reference

A class to represent an entire ANSI/NIST record.

```
#include <be_data_interchange_an2k.h>
```

Classes

- struct **CharacterSet**
- struct **DomainName**

Representation of a domain name for the user-defined Type-2 logical record implementation.

Public Types

- using **DomainName** = struct **DomainName**
- using **CharacterSet** = struct **CharacterSet**

Public Member Functions

- **AN2KRecord** (const std::string filename)
Constructor taking an AN2K record from a file.
- **AN2KRecord** (**Memory::uint8Array** &buf)
Constructor taking an AN2K record from a buffer.
- std::string **getVersionNumber** () const
- std::string **getDate** () const
- std::string **getDestinationAgency** () const
- std::string **getOriginatingAgency** () const
- std::string **getTransactionControlNumber** () const
- std::string **getNativeScanningResolution** () const
- std::string **getNominalTransmittingResolution** () const
- uint32_t **getFingerLatentCount** () const
Obtain the count of latent (Type-13) finger views.
- std::vector< **Latent::AN2KView** > **getFingerLatents** () const
Obtain all latent (Type-13) finger views.
- uint32_t **getFingerCaptureCount** () const
Obtain the count of capture (Type-14) finger views.
- std::vector< **Finger::AN2KViewCapture** > **getFingerCaptures** () const
Obtain all capture (Type-14) finger views.
- std::vector< **Finger::AN2KMinutiaeDataRecord** > **getMinutiaeDataRecordSet** () const
Obtain all minutiae (Type-9) data.
- uint8_t **getPriority** () const
Obtain the urgency with which a response is required.
- **DomainName** **getDomainName** () const
Obtain the identifier of the domain name for the user-defined Type-2 logical record implementation.

- struct tm **getGreenwichMeanTime** () const
Obtain the date and time of encoding in terms of GMT units.
- std::vector< **CharacterSet** > **getDirectoryOfCharacterSets** () const
Obtain the list of character sets other than 7-bit ASCII that may appear in the transaction.

Static Public Member Functions

- static std::set< int > **recordLocations** (**Memory::uint8Array** &buf, const **View::AN2KView::RecordType** recordType)
Find the position within a buffer of all Records of a particular type.
- static std::set< int > **recordLocations** (const ANSI_NIST *an2k, const **View::AN2KView::RecordType** recordType)
Find the position within an ANSI_NIST struct of all Records of a particular type.

G.4.1 Detailed Description

A class to represent an entire ANSI/NIST record.

An object of this class can be used to retrieve all the general record information, finger views, and other components of the ANSI/NIST record.

G.4.2 Member Typedef Documentation

G.4.2.1 CharacterSet

using **BiometricEvaluation::DataInterchange::AN2KRecord::CharacterSet** = struct **CharacterSet**
Convenience alias for struct **CharacterSet** (p. 254)

G.4.2.2 DomainName

using **BiometricEvaluation::DataInterchange::AN2KRecord::DomainName** = struct **DomainName**
Convenience alias for struct **DomainName** (p. 308)

G.4.3 Constructor & Destructor Documentation

G.4.3.1 AN2KRecord() [1/2]

```
BiometricEvaluation::DataInterchange::AN2KRecord::AN2KRecord (
    const std::string filename )
```

Constructor taking an AN2K record from a file.

Parameters

in	<i>filename</i>	The name of the file containing the complete ANSI/NIST record.
----	-----------------	--

Exceptions

Exceptions

Error::FileError (p. 316)	An error occurred when opening or reading the file.
Error::DataError (p. 295)	An error occurred when processing the AN2K record.

G.4.3.2 AN2KRecord() [2/2]

```
BiometricEvaluation::DataInterchange::AN2KRecord::AN2KRecord (
    Memory::uint8Array & buf )
```

Constructor taking an AN2K record from a buffer.

Parameters

in	buf	The memory buffer containing the complete ANSI/NIST record.
----	-----	---

Exceptions

Error::DataError (p. 295)	An error occurred when processing the AN2K record.
---	--

G.4.4 Member Function Documentation**G.4.4.1 getDate()**

```
std::string BiometricEvaluation::DataInterchange::AN2KRecord::getDate ( ) const
```

Returns

The date field in the Type-1 record.

G.4.4.2 getDestinationAgency()

```
std::string BiometricEvaluation::DataInterchange::AN2KRecord::getDestinationAgency ( ) const
```

Returns

The destination agency ID.

G.4.4.3 getDirectoryOfCharacterSets()

```
std::vector< CharacterSet> BiometricEvaluation::DataInterchange::AN2KRecord::getDirectoryOfCharacterSets ( ) const
```

Obtain the list of character sets other than 7-bit ASCII that may appear in the transaction.

Returns

Vector of **CharacterSet** (p. 254) structs representing other character sets that may appear in the transaction.

G.4.4.4 getDomainName()

```
DomainName BiometricEvaluation::DataInterchange::AN2KRecord::getDomainName ( ) const
```

Obtain the identifier of the domain name for the user-defined Type-2 logical record implementation.

Returns

DomainName (p. 308) struct with identifier and version information (if defined).

G.4.4.5 getFingerCaptureCount()

```
uint32_t BiometricEvaluation::DataInterchange::AN2KRecord::getFingerCaptureCount ( ) const
```

Obtain the count of capture (Type-14) finger views.

Returns

The number of captures in the AN2K record.

G.4.4.6 getFingerCaptures()

```
std::vector< Finger::AN2KViewCapture> BiometricEvaluation::DataInterchange::AN2KRecord::getFingerCaptures ( ) const
```

Obtain all capture (Type-14) finger views.

The returned vector will be empty when no capture views are present in the **AN2KRecord** (p. 175).

Returns

A vector of **AN2KViewCapture** objects, each representing a single capture finger view.

G.4.4.7 getFingerLatentCount()

```
uint32_t BiometricEvaluation::DataInterchange::AN2KRecord::getFingerLatentCount ( ) const
```

Obtain the count of latent (Type-13) finger views.

Returns

The number of latents in the AN2K record.

G.4.4.8 getFingerLatents()

```
std::vector< Latent::AN2KView> BiometricEvaluation::DataInterchange::AN2KRecord::getFinger←  
Latents ( ) const
```

Obtain all latent (Type-13) finger views.

The returned vector will be empty when no latent views are present in the **AN2KRecord** (p. 175).

Returns

A vector of AN2KViewLatent objects, each representing a single latent finger view.

G.4.4.9 getGreenwichMeanTime()

```
struct tm BiometricEvaluation::DataInterchange::AN2KRecord::getGreenwichMeanTime ( ) const
```

Obtain the date and time of encoding in terms of GMT units.

Returns

struct tm encoding of the GMT field.

G.4.4.10 getMinutiaeDataRecordSet()

```
std::vector< Finger::AN2KMinutiaeDataRecord> BiometricEvaluation::DataInterchange::AN2KRecord←  
::getMinutiaeDataRecordSet ( ) const
```

Obtain all minutiae (Type-9) data.

Returns

A vector of AN2KMinutiaeDataRecord objects, each representing a single Type-9 Record.

G.4.4.11 getNativeScanningResolution()

```
std::string BiometricEvaluation::DataInterchange::AN2KRecord::getNativeScanningResolution ( )  
const
```

Returns

The native scanning resolution.

G.4.4.12 getNominalTransmittingResolution()

```
std::string BiometricEvaluation::DataInterchange::AN2KRecord::getNominalTransmittingResolution  
( ) const
```

Returns

The nominal transmitting resolution.

G.4.4.13 getOriginatingAgency()

```
std::string BiometricEvaluation::DataInterchange::AN2KRecord::getOriginatingAgency ( ) const
```

Returns

The originating agency ID.

G.4.4.14 getPriority()

```
uint8_t BiometricEvaluation::DataInterchange::AN2KRecord::getPriority ( ) const
```

Obtain the urgency with which a response is required.

Returns

Priority (1:High - 9:Low)

G.4.4.15 getTransactionControlNumber()

```
std::string BiometricEvaluation::DataInterchange::AN2KRecord::getTransactionControlNumber ( ) const
```

Returns

The transaction control number.

G.4.4.16 getVersionNumber()

```
std::string BiometricEvaluation::DataInterchange::AN2KRecord::getVersionNumber ( ) const
```

Returns

The record version field in the Type-1 record.

G.4.4.17 recordLocations() [1/2]

```
static std::set<int> BiometricEvaluation::DataInterchange::AN2KRecord::recordLocations (
    Memory::uint8Array & buf,
    const View::AN2KView::RecordType recordType ) [static]
```

Find the position within a buffer of all Records of a particular type.

Parameters

in	<i>buf</i>	AN2K Buffer to search.
in	<i>recordType</i>	The ID of the Record to search for.

Returns

Set of integer positions within buf where a recordType Record is located.

Exceptions

Error::DataError (p. 295)	An error occurred when processing the AN2K record.
----------------------------------	--

G.4.4.18 recordLocations() [2/2]

```
static std::set<int> BiometricEvaluation::DataInterchange::AN2KRecord::recordLocations (
    const ANSI_NIST * an2k,
    const View::AN2KView::RecordType recordType ) [static]
```

Find the position within an ANSI_NIST struct of all Records of a particular type.

Parameters

in	<i>an2k</i>	ANSI_NIST struct to search.
in	<i>recordType</i>	The ID of the Record to search for.

Returns

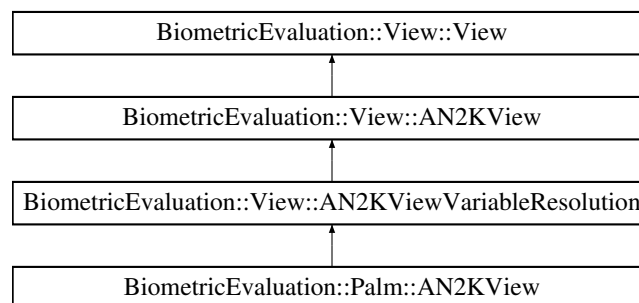
Set of integer positions within the ANSI_NIST struct where a recordType Record is located.

G.5 BiometricEvaluation::Palm::AN2KView Class Reference

A class to represent a single **Palm** (p. 149) view and derived information.

```
#include <be_palm_an2kview.h>
```

Inheritance diagram for BiometricEvaluation::Palm::AN2KView:



Public Member Functions

- **AN2KView** (const std::string &filename, const uint32_t recordNumber)
Construct an AN2K palm view from a file.
- **AN2KView** (**BiometricEvaluation::Memory::uint8Array** &buf, const uint32_t recordNumber)
Construct an AN2K palm view from a memory buffer.

- **Palm::Position** `getPosition ()` const
Obtain the palm position.
- **QualityMetricSet** `getPalmQualityMetric ()` const
Obtain the palm quality metric.

Additional Inherited Members

G.5.1 Detailed Description

A class to represent a single **Palm** (p. 149) view and derived information.

A **Palm::AN2KView** (p. 181) object represents an ANSI/NIST Type-15 record, and can return the image as well as the other information associated with that image, such as the minutiae from the corresponding Type-9 record.

G.5.2 Constructor & Destructor Documentation

G.5.2.1 AN2KView() [1/2]

```
BiometricEvaluation::Palm::AN2KView::AN2KView (
    const std::string & filename,
    const uint32_t recordNumber )
```

Construct an AN2K palm view from a file.

The file must contain the entire AN2K record, not just the palm image and/or minutiae records.

G.5.2.2 AN2KView() [2/2]

```
BiometricEvaluation::Palm::AN2KView::AN2KView (
    BiometricEvaluation::Memory::uint8Array & buf,
    const uint32_t recordNumber )
```

Construct an AN2K palm view from a memory buffer.

The buffer must contain the entire AN2K record, not just the palm image and/or minutiae records.

G.5.3 Member Function Documentation

G.5.3.1 getPalmQualityMetric()

```
QualityMetricSet BiometricEvaluation::Palm::AN2KView::getPalmQualityMetric ( ) const
```

Obtain the palm quality metric.

Returns

QualityMetricSet containing the set of metrics the palm image.

G.5.3.2 getPosition()

Palm::Position BiometricEvaluation::Palm::AN2KView::getPosition () const
Obtain the palm position.

Returns

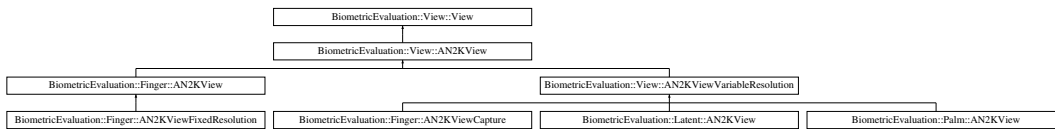
The palm position.

G.6 BiometricEvaluation::View::AN2KView Class Reference

A class to represent single biometric view and derived information.

```
#include <be_view_an2kview.h>
```

Inheritance diagram for BiometricEvaluation::View::AN2KView:



Public Types

- enum **RecordType** : uint16_t {
 Type_1 = 1, **Type_2** = 2, **Type_3** = 3, **Type_4** = 4,
 Type_5 = 5, **Type_6** = 6, **Type_7** = 7, **Type_8** = 8,
 Type_9 = 9, **Type_10** = 10, **Type_11** = 11, **Type_12** = 12,
 Type_13 = 13, **Type_14** = 14, **Type_15** = 15, **Type_16** = 16,
 Type_17 = 17, **Type_99** = 99 }
- enum **DeviceMonitoringMode** {
 DeviceMonitoringMode::Controlled, **DeviceMonitoringMode::Assisted**, **DeviceMonitoringMode::Observed**, **DeviceMonitoringMode::Unattended**,
 DeviceMonitoringMode::Unknown, **DeviceMonitoringMode::NA** }

The level of human monitoring for the image capture device.

Public Member Functions

- AN2KView** (const std::string filename, const **RecordType** typeId, const uint32_t recordNumber)
Construct an AN2K view from a file.
- AN2KView** (**Memory::uint8Array** &buf, const **RecordType** typeId, const uint32_t recordNumber)
Construct an AN2K view from a buffer.
- std::vector< **Finger::AN2KMinutiaeDataRecord** > **getMinutiaeDataRecordSet** () const
Obtain the set of minutiae records.
- RecordType** **getRecordType** () const
Obtain the ANSI-NIST record type.

Static Public Member Functions

- static **DeviceMonitoringMode** **convertDeviceMonitoringMode** (const char *dmm)
Convert a device monitoring mode indicator from an AN2K record.

- static **Image::CompressionAlgorithm** **convertCompressionAlgorithm** (const uint16_t recordType, const unsigned char *an2kValue)

Convert a compression algorithm indicator from an AN2K finger image record.

Static Public Attributes

- static const double **MinimumScanResolutionPPMM**
Constants to define the minimum resolution used for fingerprint images in an AN2k record.
- static const double **HalfMinimumScanResolutionPPMM**
- static const int **FixedResolutionBitDepth** = 8
The defined bit-depth for fixed-resolution images.

Protected Member Functions

- **Memory::AutoBuffer**< ANSI_NIST > **getAN2K** () const
Obtain the complete ANSI/NIST record set.
- RECORD * **getAN2KRecord** () const
Obtain a pointer to the single ANSI/NIST record.

G.6.1 Detailed Description

A class to represent single biometric view and derived information.

This abstraction represents the image and derived information taken from an ANSI/NIST record.

For these types of records, the image resolution and scan resolution are identical. For compressed images, applications can compare the image resolution and size taken from the Type-3/4/5/6 record to that returned by the **Image** (p. 120) object directly.

G.6.2 Member Enumeration Documentation

G.6.2.1 DeviceMonitoringMode

enum **BiometricEvaluation::View::AN2KView::DeviceMonitoringMode** [strong]

The level of human monitoring for the image capture device.

Enumerator

Controlled	Operator physically controls the subject to acquire biometric sample.
Assisted	Person available to provide assistance to the subject submitting the biometric.
Observed	Person present to observe the operation of the device but provides no assistance.
Unattended	No one present to observe or provide assistance.
Unknown	No information is known.
NA	Optional field – not specified

G.6.2.2 RecordType

enum BiometricEvaluation::View::AN2KView::RecordType : uint16_t [strong]
The type of AN2K record.

G.6.3 Constructor & Destructor Documentation

G.6.3.1 AN2KView() [1/2]

BiometricEvaluation::View::AN2KView::AN2KView (
const std::string filename,
const RecordType typeID,
const uint32_t recordNumber)
Construct an AN2K view from a file.
The file must contain the entire AN2K record, not just the image and other view-related records.

G.6.3.2 AN2KView() [2/2]

BiometricEvaluation::View::AN2KView::AN2KView (
Memory::uint8Array & buf,
const RecordType typeID,
const uint32_t recordNumber)
Construct an AN2K view from a buffer.
The buffer must contain the entire AN2K record, not just the image and other view-related records.

G.6.4 Member Function Documentation

G.6.4.1 convertCompressionAlgorithm()

static Image::CompressionAlgorithm BiometricEvaluation::View::AN2KView::convertCompression↔
Algorithm (
const uint16_t recordType,
const unsigned char * an2kValue) [static]
Convert a compression algorithm indicator from an AN2K finger image record.

Parameters

recordType	The AN2K record type as an integer, allowing the value taken directly from the AN2K record or a RecordType::Kind to be passed in.
an2kValue	Compression type data as read from an AN2K record.

Returns

The compression algorithm.

Exceptions

Error::DataError (p. 295)	Invalid compression algorithm for record type.
---------------------------	--

Exceptions

Error::ParameterError (p. 474)	Invalid record type.
---------------------------------------	----------------------

G.6.4.2 convertDeviceMonitoringMode()

```
static DeviceMonitoringMode BiometricEvaluation::View::AN2KView::convertDeviceMonitoringMode
(
    const char * dmm ) [static]
    Convert a device monitoring mode indicator from an AN2K record.
```

Parameters

<i>dmm</i>	Item value for device monitoring mode from an AN2K record.
------------	--

Returns

DeviceMonitoringMode representation of dmm.

Exceptions

Error::DataError (p. 295)	Invalid format of dmm.
----------------------------------	------------------------

G.6.4.3 getAN2KRecord()

```
RECORD* BiometricEvaluation::View::AN2KView::getAN2KRecord ( ) const [protected]
    Obtain a pointer to the single ANSI/NIST record.
    Child classes use this method to obtain a pointer to the specific ANSI/NIST record that was searched for
    by this class object.
```

G.6.4.4 getMinutiaeDataRecordSet()

```
std::vector< Finger::AN2KMinutiaeDataRecord> BiometricEvaluation::View::AN2KView::getMinutiae↵
DataRecordSet ( ) const
    Obtain the set of minutiae records.
    Each AN2KViewVariableResolution (p. 200) may have more than one associated Type-9 record and each
    Type-9 record may have more than one minutiae format.
```

Returns

A vector of minutiae data records.

G.6.4.5 getRecordType()

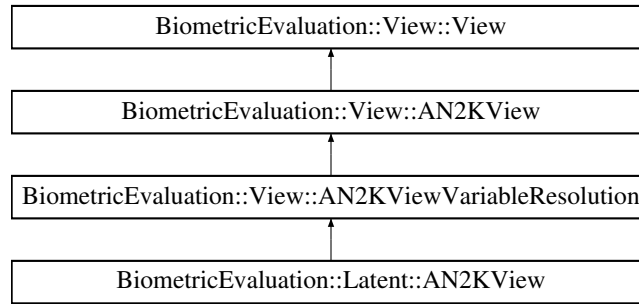
```
RecordType BiometricEvaluation::View::AN2KView::getRecordType ( ) const
    Obtain the ANSI-NIST record type.
```

Returns

The type of record used to construct this object.

G.7 BiometricEvaluation::Latent::AN2KView Class Reference

Inheritance diagram for BiometricEvaluation::Latent::AN2KView:



Public Member Functions

- **AN2KView** (const std::string &filename, const uint32_t recordNumber)
Construct an AN2K finger view from a file.
- **AN2KView** (**Memory::uint8Array** &buf, const uint32_t recordNumber)
Construct an AN2K finger view using from a memory buffer.
- Feature::FGPSet **getPositions** () const
Obtain the set of finger positions.
- QualityMetricSet **getLatentQualityMetric** () const
Obtain metrics for latent image quality score data for the image stored in this record.
- Finger::PositionDescriptors **getSearchPositionDescriptors** () const
Return search position descriptors.
- PrintPositionCoordinateSet **getPrintPositionCoordinates** () const
Obtain print position coordinates.

Additional Inherited Members

G.7.1 Constructor & Destructor Documentation

G.7.1.1 AN2KView() [1/2]

```

BiometricEvaluation::Latent::AN2KView::AN2KView (
    const std::string & filename,
    const uint32_t recordNumber )

```

Construct an AN2K finger view from a file.

The file must contain the entire AN2K record, not just the finger image and/or minutiae records.

G.7.1.2 AN2KView() [2/2]

```
BiometricEvaluation::Latent::AN2KView::AN2KView (
    Memory::uint8Array & buf,
    const uint32_t recordNumber )
```

Construct an AN2K finger view using from a memory buffer.

The buffer must contain the entire AN2K record, not just the finger image and/or minutiae records.

G.7.2 Member Function Documentation

G.7.2.1 getLatentQualityMetric()

```
QualityMetricSet BiometricEvaluation::Latent::AN2KView::getLatentQualityMetric ( ) const
```

Obtain metrics for latent image quality score data for the image stored in this record.

Returns

Latent quality metrics

G.7.2.2 getPositions()

```
Feature::FGPSet BiometricEvaluation::Latent::AN2KView::getPositions ( ) const
```

Obtain the set of finger positions.

An AN2K latent image record contains a set of possible finger positions. This method returns that set as read from the image record. Any minutiae record (Type-9) associated with this image will have its own set of positions.

G.7.2.3 getPrintPositionCoordinates()

```
PrintPositionCoordinateSet BiometricEvaluation::Latent::AN2KView::getPrintPositionCoordinates
( ) const
```

Obtain print position coordinates.

Returns

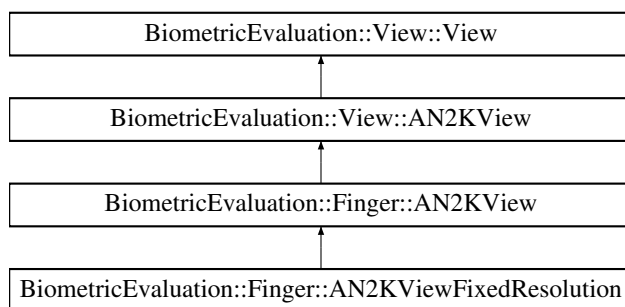
Set of all PrintPositionCoordinates

G.8 BiometricEvaluation::Finger::AN2KView Class Reference

A class to represent single finger view and derived information.

```
#include <be_finger_an2kview.h>
```

Inheritance diagram for BiometricEvaluation::Finger::AN2KView:



Public Member Functions

- `std::vector< AN2KMinutiaeDataRecord > getMinutiaeDataRecordSet () const`
Obtain the set of minutiae records.
- `Finger::PositionSet getPositions () const`
Obtain the set of finger positions.
- `Finger::Impression getImpressionType () const`
Obtain the finger impression code.

Static Public Member Functions

- static `Finger::Position convertPosition (int an2kFGP)`
Convert a compression algorithm indicator from an AN2K finger image record.
- static `Finger::PositionSet populateFGP (FIELD *field)`
Read the finger positions from an AN2K record.
- static `Finger::Impression convertImpression (const unsigned char *str)`
Convert an impression code from a string.
- static `Finger::FingerImageCode convertFingerImageCode (const char *str)`
Convert an finger image code from a string.

Protected Member Functions

- `AN2KView (const std::string filename, const RecordType typeID, const uint32_t recordNumber)`
Construct an AN2K finger view from a file.
- `AN2KView (Memory::uint8Array &buf, const RecordType typeID, const uint32_t recordNumber)`
Construct an AN2K finger view from a buffer.
- void `addMinutiaeDataRecord (Finger::AN2KMinutiaeDataRecord &mdr)`
Add a minutiae data record to the AN2KMinutiaeDataRecord (p. 171) set.
- void `setPositions (Finger::PositionSet &ps)`
Add a position set to the collection of position sets.
- void `setImpressionType (Finger::Impression &imp)`
Mutator for the impression type.

Additional Inherited Members

G.8.1 Detailed Description

A class to represent single finger view and derived information.

A base **Finger::AN2KView** (p. 188) object represents an ANSI/NIST Type-3/4/5/6 record, and can return the image as well as the other information associated with that image, such as the minutiae from the corresponding Type-9 record.

For these types of records, the image resolution and scan resolution are identical. For compressed images, applications can compare the image resolution and size taken from the Type-3/4/5/6 record to that returned by the **Image** (p. 120) object directly.

G.8.2 Constructor & Destructor Documentation

G.8.2.1 AN2KView() [1/2]

```
BiometricEvaluation::Finger::AN2KView::AN2KView (
    const std::string filename,
    const RecordType typeId,
    const uint32_t recordNumber ) [protected]
```

Construct an AN2K finger view from a file.

The file must contain the entire AN2K record, not just the finger image and/or minutiae records.

Parameters

in	<i>filename</i>	The name of the file containing the AN2K record.
in	<i>typeID</i>	The type of AN2K finger view: Type-3/Type-4/etc.
in	<i>recordNumber</i>	Which finger record to read as there may be multiple finger views of the same type within a single AN2K record.

Exceptions

Error::ParameterError (p. 474)	An invalid parameter was passed in.
Error::DataError (p. 295)	An error occurred when parsing the AN2K record.
Error::FileError (p. 316)	An error occurred when reading the file.

G.8.2.2 AN2KView() [2/2]

```
BiometricEvaluation::Finger::AN2KView::AN2KView (
    Memory::uint8Array & buf,
    const RecordType typeId,
    const uint32_t recordNumber ) [protected]
```

Construct an AN2K finger view from a buffer.

The buffer must contain the entire AN2K record, not just the finger image and/or minutiae records.

Parameters

in	<i>buf</i>	The buffer containing the AN2K record.
in	<i>typeID</i>	The type of AN2K finger view: Type-3/Type-4/etc.
in	<i>recordNumber</i>	Which finger record to read as there may be multiple finger views of the same type within a single AN2K record.

Exceptions

Error::ParameterError (p. 474)	An invalid parameter was passed in.
Error::DataError (p. 295)	An error occurred when parsing the AN2K record.

G.8.3 Member Function Documentation

G.8.3.1 addMinutiaeDataRecord()

```
void BiometricEvaluation::Finger::AN2KView::addMinutiaeDataRecord (
    Finger::AN2KMinutiaeDataRecord & mdr ) [protected]
    Add a minutiae data record to the AN2KMinutiaeDataRecord (p. 171) set.
```

Parameters

in	<i>mdr</i>	The minutiae data record to be added.
----	------------	---------------------------------------

G.8.3.2 convertFingerImageCode()

```
static Finger::FingerImageCode BiometricEvaluation::Finger::AN2KView::convertFingerImageCode
(
    const char * str ) [static]
    Convert an finger image code from a string.
```

Parameters

in	<i>str</i>	The character string containing the image code.
----	------------	---

Returns

A FingerImageCode value.

Exceptions

Error::DataError (p. 295)	The string contains an invalid image code.
----------------------------------	--

G.8.3.3 convertPosition()

```
static Finger::Position BiometricEvaluation::Finger::AN2KView::convertPosition (
    int an2kFGP ) [static]
```

Convert a compression algorithm indicator from an AN2K finger image record.

Parameters

in	<i>an2kFGP</i>	A finger position code as defined by the AN2K standard.
----	----------------	---

Exceptions

Error::DataError (p. 295)	The position code is invalid.
----------------------------------	-------------------------------

G.8.3.4 getImpressionType()

```
Finger::Impression BiometricEvaluation::Finger::AN2KView::getImpressionType ( ) const
```

Obtain the finger impression code.

Returns

The finger impression code.

G.8.3.5 getMinutiaeDataRecordSet()

```
std::vector< AN2KMinutiaeDataRecord> BiometricEvaluation::Finger::AN2KView::getMinutiaeData↵
RecordSet ( ) const
```

Obtain the set of minutiae records.

Because it is possible to have more than one Type-9 record associated with a finger view, this method returns a set of objects, each one representing a single Type-9 record.

Returns

The vector of minutiae data records.

G.8.3.6 getPositions()

```
Finger::PositionSet BiometricEvaluation::Finger::AN2KView::getPositions ( ) const
```

Obtain the set of finger positions.

An AN2K finger image record contains a set of possible finger positions. This method returns that set as read from the image record. Any minutiae record (Type-9) associated with this image will have its own set of positions.

G.8.3.7 populateFGP()

```
static Finger::PositionSet BiometricEvaluation::Finger::AN2KView::populateFGP (
    FIELD * field ) [static]
```

Read the finger positions from an AN2K record.
An AN2K finger image record can have multiple values * for the finger position. Pull them out of the position field and return them as a set.

Exceptions

<i>Error::DataError</i> (p. 295)	The data contains an invalid value.
----------------------------------	-------------------------------------

G.8.3.8 setImpressionType()

```
void BiometricEvaluation::Finger::AN2KView::setImpressionType (
    Finger::Impression & imp ) [protected]
```

Mutator for the impression type.

Parameters

in	<i>imp</i>	The impression type for this finger view.
----	------------	---

G.8.3.9 setPositions()

```
void BiometricEvaluation::Finger::AN2KView::setPositions (
    Finger::PositionSet & ps ) [protected]
```

Add a position set to the collection of position sets.

Parameters

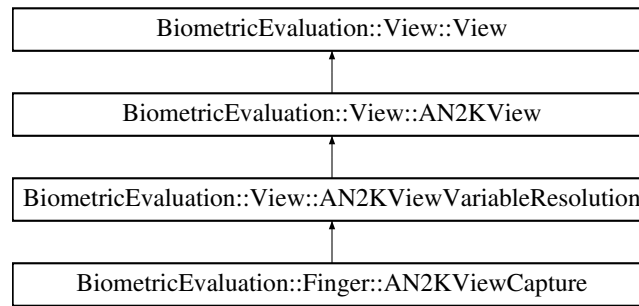
in	<i>ps</i>	The position set to be added.
----	-----------	-------------------------------

G.9 BiometricEvaluation::Finger::AN2KViewCapture Class Reference

Represents an ANSI/NIST variable-resolution finger image.

```
#include <be_finger_an2kview_capture.h>
```

Inheritance diagram for BiometricEvaluation::Finger::AN2KViewCapture:



Classes

- struct **FingerSegmentPosition**
Locations of an individual finger segment in a slap.

Public Types

- enum **AmputatedBandaged** { **AmputatedBandaged::Amputated**, **AmputatedBandaged::Bandaged**, **AmputatedBandaged::NA** }
Enumeration of the finger amputated or bandaged code, a reason that a capture could not be made.
- using **FingerSegmentPosition** = struct **FingerSegmentPosition**
- using **FingerSegmentPositionSet** = std::vector< **FingerSegmentPosition** >

Public Member Functions

- **AN2KViewCapture** (const std::string &filename, const uint32_t recordNumber)
Construct an AN2K finger view from a file.
- **AN2KViewCapture** (**Memory::uint8Array** &buf, const uint32_t recordNumber)
Construct an AN2K finger view using from a memory buffer.
- **QualityMetricSet** **extractNISTQuality** (const FIELD *field)
Extract the NQM information from an AN2K FIELD.
- **Finger::Position** **getPosition** () const
Obtain the finger position.
- **PositionDescriptors** **getPrintPositionDescriptors** () const
Return search position descriptors.
- **PrintPositionCoordinateSet** **getPrintPositionCoordinates** () const
Obtain print position coordinates.
- **QualityMetricSet** **getNISTQualityMetric** () const
Obtain the NIST quality metric for all segmented finger images.
- **QualityMetricSet** **getSegmentationQualityMetric** () const
Obtain the segmentation quality metric for all segmented finger images.
- **AmputatedBandaged** **getAmputatedBandaged** () const
- **FingerSegmentPositionSet** **getFingerSegmentPositionSet** () const
- **FingerSegmentPositionSet** **getAlternateFingerSegmentPositionSet** () const
- **QualityMetricSet** **getFingerprintQualityMetric** () const
Obtain metrics for fingerprint image quality score data for the image stored in this record.

Additional Inherited Members

G.9.1 Detailed Description

Represents an ANSI/NIST variable-resolution finger image.

If the complete ANSI/NIST record contains a corresponding Type-9 (finger minutiae) record, an object of this class can be used to retrieve the minutiae set(s).

G.9.2 Member Enumeration Documentation

G.9.2.1 AmputatedBandaged

enum **BiometricEvaluation::Finger::AN2KViewCapture::AmputatedBandaged** [strong]

Enumeration of the finger amputated or bandaged code, a reason that a capture could not be made.

Enumerator

Amputated	Amputation
Bandaged	Unable to print (e.g., bandaged)
NA	Optional field – not specified

G.9.3 Constructor & Destructor Documentation

G.9.3.1 AN2KViewCapture() [1/2]

```
BiometricEvaluation::Finger::AN2KViewCapture::AN2KViewCapture (
    const std::string & filename,
    const uint32_t recordNumber )
```

Construct an AN2K finger view from a file.

The file must contain the entire AN2K record, not just the finger image and/or minutiae records. The object is constructed based on the nth variable resolution record found.

Parameters

in	<i>filename</i>	The name of the file containing the complete ANSI/NIST record.
in	<i>recordNumber</i>	The number of variable resolution record to read from the complete AN2K record.

Exceptions

Error::ParameterError (p. 474)	
Error::DataError (p. 295)	
Error::FileError (p. 316)	An error occurred when opening or reading the file.

G.9.3.2 AN2KViewCapture() [2/2]

```
BiometricEvaluation::Finger::AN2KViewCapture::AN2KViewCapture (
    Memory::uint8Array & buf,
    const uint32_t recordNumber )
```

Construct an AN2K finger view using from a memory buffer.

The buffer must contain the entire AN2K record, not just the finger image and/or minutiae records.

G.9.4 Member Function Documentation

G.9.4.1 extractNISTQuality()

```
QualityMetricSet BiometricEvaluation::Finger::AN2KViewCapture::extractNISTQuality (
    const FIELD * field )
```

Extract the NQM information from an AN2K FIELD.

Parameters

<i>field</i>	FIELD containing properly formatted NQM data
--------------	--

Returns

QualityMetricSet representation of field.

Exceptions

Error::DataError (p. 295)	Invalid format of field for NQM.
----------------------------------	----------------------------------

G.9.4.2 getAlternateFingerSegmentPositionSet()

```
FingerSegmentPositionSet BiometricEvaluation::Finger::AN2KViewCapture::getAlternateFingerSegment↔
PositionSet ( ) const
```

Returns

Optional set of polygonal finger segment positions for all finger segments.

G.9.4.3 getAmputatedBandaged()

```
AmputatedBandaged BiometricEvaluation::Finger::AN2KViewCapture::getAmputatedBandaged ( ) const
```

Returns

Optional amputated or bandaged code.

G.9.4.4 getFingerprintQualityMetric()

```
QualityMetricSet BiometricEvaluation::Finger::AN2KViewCapture::getFingerprintQualityMetric (
) const
```

Obtain metrics for fingerprint image quality score data for the image stored in this record.

Returns

Fingerprint quality metrics

G.9.4.5 getFingerSegmentPositionSet()

```
FingerSegmentPositionSet BiometricEvaluation::Finger::AN2KViewCapture::getFingerSegmentPosition↵
Set ( ) const
```

Returns

Optional set of rectangular finger segment positions for all finger segments.

G.9.4.6 getNISTQualityMetric()

```
QualityMetricSet BiometricEvaluation::Finger::AN2KViewCapture::getNISTQualityMetric ( ) const
```

Obtain the NIST quality metric for all segmented finger images.

Returns

QualityMetricSet containing the NIST quality metric for all segmented finger images.

Vendor ID and Product Code are undefined, as they are unused by NQM.

G.9.4.7 getPosition()

```
Finger::Position BiometricEvaluation::Finger::AN2KViewCapture::getPosition ( ) const
```

Obtain the finger position.

An AN2K finger image record contains a single finger positions. Any minutiae record (Type-9) associated with this image will have its own set of positions.

G.9.4.8 getPrintPositionCoordinates()

```
PrintPositionCoordinateSet BiometricEvaluation::Finger::AN2KViewCapture::getPrintPositionCoordinates
( ) const
```

Obtain print position coordinates.

Returns

Set of all PrintPositionCoordinates

G.9.4.9 getSegmentationQualityMetric()

```
QualityMetricSet BiometricEvaluation::Finger::AN2KViewCapture::getSegmentationQualityMetric (
) const
```

Obtain the segmentation quality metric for all segmented finger images.

Returns

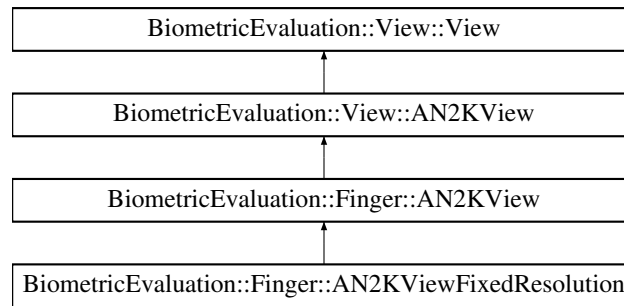
QualityMetricSet containing the segmentation quality metric for all segmented finger images.

G.10 BiometricEvaluation::Finger::AN2KViewFixedResolution Class Reference

A class to represent single finger view and derived information.

```
#include <be_finger_an2kview_fixedres.h>
```

Inheritance diagram for BiometricEvaluation::Finger::AN2KViewFixedResolution:



Public Member Functions

- **AN2KViewFixedResolution** (const std::string filename, const **RecordType** typeID, const uint32_t recordNumber)
Construct an AN2K finger view from a file.
- **AN2KViewFixedResolution** (**Memory::uint8Array** &buf, const **RecordType** typeID, const uint32_t recordNumber)
Construct an AN2K finger view from a buffer.

Additional Inherited Members

G.10.1 Detailed Description

A class to represent single finger view and derived information.

A base **Finger::AN2KView** (p. 188) object represents an ANSI/NIST Type-3/4/5/6 record, and can return the image as well as the other information associated with that image, such as the minutiae from the corresponding Type-9 record.

For these types of records, the image resolution and scan resolution are identical. For compressed images, applications can compare the image resolution and size taken from the Type-3/4/5/6 record to that returned by the **Image** (p. 120) object directly.

G.10.2 Constructor & Destructor Documentation

G.10.2.1 AN2KViewFixedResolution() [1/2]

```
BiometricEvaluation::Finger::AN2KViewFixedResolution::AN2KViewFixedResolution (
    const std::string filename,
    const RecordType typeID,
    const uint32_t recordNumber )
```

Construct an AN2K finger view from a file.

The file must contain the entire AN2K record, not just the finger image and/or minutiae records.

Parameters

in	<i>filename</i>	The name of the file containing the AN2K record.
in	<i>typeID</i>	The type of AN2K finger view: Type-3/Type-4/etc.
in	<i>recordNumber</i>	Which finger record to read as there may be multiple finger views of the same type within a single AN2K record.

Exceptions

Error::ParameterError (p. 474)	An invalid parameter was passed in.
Error::DataError (p. 295)	An error occurred when parsing the AN2K record.
Error::FileError (p. 316)	An error occurred when reading the file.

G.10.2.2 AN2KViewFixedResolution() [2/2]

```
BiometricEvaluation::Finger::AN2KViewFixedResolution::AN2KViewFixedResolution (
    Memory::uint8Array & buf,
    const RecordType typeID,
    const uint32_t recordNumber )
```

Construct an AN2K finger view from a buffer.

The buffer must contain the entire AN2K record, not just the finger image and/or minutiae records.

Parameters

in	<i>buf</i>	The buffer containing the AN2K record.
in	<i>typeID</i>	The type of AN2K finger view: Type-3/Type-4/etc.
in	<i>recordNumber</i>	Which finger record to read as there may be multiple finger views of the same type within a single AN2K record.

Exceptions

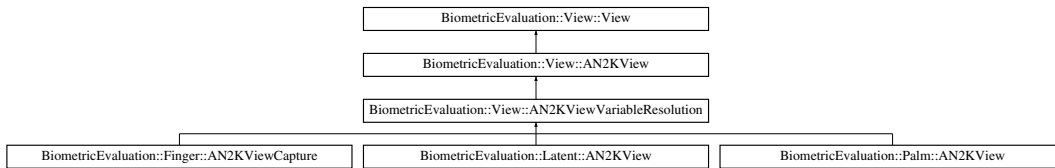
Error::ParameterError (p. 474)	An invalid parameter was passed in.
Error::DataError (p. 295)	An error occurred when parsing the AN2K record.

G.11 BiometricEvaluation::View::AN2KViewVariableResolution Class Reference

A class to represent single view based on an ANSI/NIST record.

```
#include <be_view_an2kview_varres.h>
```

Inheritance diagram for BiometricEvaluation::View::AN2KViewVariableResolution:



Classes

- struct **AN2KQualityMetric**
A structure to represent an AN2K quality metric.
- struct **PrintPositionCoordinate**
Offsets to the bounding boxes for the EJI, full finger views, or EJI segments.

Public Types

- using **AN2KQualityMetric** = struct **AN2KQualityMetric**
- using **QualityMetricSet** = std::vector< **AN2KQualityMetric** >
- using **PrintPositionCoordinate** = struct **PrintPositionCoordinate**
- using **PrintPositionCoordinateSet** = std::vector< **PrintPositionCoordinate** >

Public Member Functions

- **Finger::Impression** **getImpressionType** () const
- std::string **getSourceAgency** () const
- std::string **getCaptureDate** () const
- std::string **getComment** () const
Obtain the comment field.
- **Memory::uint8Array** **getUserDefinedField** (const uint16_t field) const
Obtain a user-defined field.

Static Public Member Functions

- static QualityMetricSet **extractQuality** (FIELD *field, **Feature::PositionType** type)
Read a Quality Metric Set from a variable resolution AN2K record.
- static **Memory::uint8Array** **parseUserDefinedField** (const RECORD *const record, int fieldID)
Read raw bytes from a user-defined AN2K field.

Protected Member Functions

- **AN2KViewVariableResolution** (const std::string &filename, const **RecordType** typeID, const uint32_t recordNumber)
Construct an AN2K finger view from a file.
- **AN2KViewVariableResolution** (**Memory::uint8Array** &buf, const **RecordType** typeID, const uint32_t recordNumber)
Construct an AN2K finger view using from a memory buffer.
- Feature::FGPSet **getPositions** () const
Obtain the set of finger positions.
- Finger::PositionDescriptors **getPositionDescriptors** () const
Obtain the position descriptors.
- PrintPositionCoordinateSet **getPrintPositionCoordinates** () const
Obtain print position coordinates.
- QualityMetricSet **getQualityMetric** () const
Obtain quality metrics for associated image record.

Additional Inherited Members

G.11.1 Detailed Description

A class to represent single view based on an ANSI/NIST record.

The view represents a variable resolution (Type-13/14/15) AN2K record.

G.11.2 Constructor & Destructor Documentation

G.11.2.1 AN2KViewVariableResolution() [1/2]

```
BiometricEvaluation::View::AN2KViewVariableResolution::AN2KViewVariableResolution (
    const std::string & filename,
    const RecordType typeID,
    const uint32_t recordNumber ) [protected]
```

Construct an AN2K finger view from a file.

The file must contain the entire AN2K record, not just the finger image and/or minutiae records.

G.11.2.2 AN2KViewVariableResolution() [2/2]

```
BiometricEvaluation::View::AN2KViewVariableResolution::AN2KViewVariableResolution (
    Memory::uint8Array & buf,
    const RecordType typeID,
    const uint32_t recordNumber ) [protected]
```

Construct an AN2K finger view using from a memory buffer.

The buffer must contain the entire AN2K record, not just the finger image and/or minutiae records.

G.11.3 Member Function Documentation

G.11.3.1 extractQuality()

```
static QualityMetricSet BiometricEvaluation::View::AN2KViewVariableResolution::extractQuality
(
```

```
    FIELD * field,
```

```
    Feature::PositionType type ) [static]
```

Read a Quality Metric Set from a variable resolution AN2K record.

Parameters

in	<i>field</i>	A pointer to the field within the AN2K record.
in	<i>type</i>	The position type.

Exceptions

Error::DataError (p. 295)	The data contains an invalid value.
----------------------------------	-------------------------------------

G.11.3.2 getCaptureDate()

```
std::string BiometricEvaluation::View::AN2KViewVariableResolution::getCaptureDate ( ) const
```

Returns

The capture date.

G.11.3.3 getComment()

```
std::string BiometricEvaluation::View::AN2KViewVariableResolution::getComment ( ) const
```

Obtain the comment field.

The comment field is optional in an AN2K record.

Returns

The comment field, empty string if not present.

G.11.3.4 getImpressionType()

```
Finger::Impression BiometricEvaluation::View::AN2KViewVariableResolution::getImpressionType
( ) const
```

Returns

The finge/palmr impression code.

G.11.3.5 getPositionDescriptors()

```
Finger::PositionDescriptors BiometricEvaluation::View::AN2KViewVariableResolution::getPositionDescriptors ( ) const [protected]
```

Obtain the position descriptors.

Subclasses specialize the position descriptors based on the semantic meaning pertinent for that class.

Returns

The set of position descriptors.

G.11.3.6 getPositions()

```
Feature::FGPSet BiometricEvaluation::View::AN2KViewVariableResolution::getPositions ( ) const [protected]
```

Obtain the set of finger positions.

An AN2K variable resolution image record may contain a set of possible friction ridge positions. This method returns that set as read from the image record. Subclasses must retrieve the position information relevant to that class.

Returns

The set of friction ridge generalized positions.

G.11.3.7 getPrintPositionCoordinates()

```
PrintPositionCoordinateSet BiometricEvaluation::View::AN2KViewVariableResolution::getPrintPositionCoordinates ( ) const [protected]
```

Obtain print position coordinates.

Returns

Set of all PrintPositionCoordinates

G.11.3.8 getQualityMetric()

```
QualityMetricSet BiometricEvaluation::View::AN2KViewVariableResolution::getQualityMetric ( ) const [protected]
```

Obtain quality metrics for associated image record.

Returns

Quality metrics

G.11.3.9 getSourceAgency()

```
std::string BiometricEvaluation::View::AN2KViewVariableResolution::getSourceAgency ( ) const
```

Returns

The source agency.

G.11.3.10 getUserDefinedField()

Memory::uint8Array BiometricEvaluation::View::AN2KViewVariableResolution::getUserDefinedField (

const uint16_t field) const

Obtain a user-defined field.

Fields are retrieved on-demand and then cached.

Parameters

in	field	The field number to retrieve.
----	-------	-------------------------------

Returns

Raw bytes read from the field.

Exceptions

Error::ObjectDoesNotExist (p. 457)	There is no user-defined field with the requested field number.
Error::ParameterError (p. 474)	Invalid value for field.
Error::StrategyError (p. 567)	Field could not be cached.

G.11.3.11 parseUserDefinedField()

static **Memory::uint8Array** BiometricEvaluation::View::AN2KViewVariableResolution::parseUser←DefinedField (

const RECORD *const record,

int fieldID) [static]

Read raw bytes from a user-defined AN2K field.

Parameters

in	record	Pointer to a RECORD containing the user-defined field.
in	fieldID	The user-defined field number.

Returns

Raw bytes from field.

Exceptions

Error::ObjectDoesNotExist (p. 457)	There is no user-defined field with the requested field number.
Error::ParameterError (p. 474)	Invalid value for fieldID.

G.12 BiometricEvaluation::Feature::Sort::Angle Class Reference

```
#include <be_feature_sort.h>
```

Public Member Functions

- **bool operator()** (const **BiometricEvaluation::Feature::MinutiaPoint** &lhs, const **BiometricEvaluation::Feature::MinutiaPoint** &rhs) const

G.12.1 Detailed Description

Sort (p. 113) by increasing angle (theta)

G.12.2 Member Function Documentation

G.12.2.1 operator()

```
bool BiometricEvaluation::Feature::Sort::Angle::operator() (
    const BiometricEvaluation::Feature::MinutiaPoint & lhs,
    const BiometricEvaluation::Feature::MinutiaPoint & rhs ) const
MinutiaPoint (p. 444) angle ascending comparator.
```

G.13 BiometricEvaluation::DataInterchange::ANSI2004Record Class Reference

```
#include <be_data_interchange_ansi2004.h>
```

Public Member Functions

- **ANSI2004Record** (const **BiometricEvaluation::Memory::uint8Array** &fmr, const **BiometricEvaluation::Memory::uint8Array** &fir)
ANSI2004Record (p. 205) constructor using a pair of finger minutia and image records.
- **ANSI2004Record** (const std::string &fmrPath, const std::string &firPath)
ANSI2004Record (p. 205) constructor using a pair of finger minutia and image records.
- **ANSI2004Record** (const std::initializer_list< **BiometricEvaluation::Finger::ANSI2004View** > &views)
ANSI2004Record (p. 205) constructor using a set of finger view records.
- **Finger::ANSI2004View getView** (const uint64_t viewNumber) const
Obtain an ANSI2004View.
- uint64_t **insertView** (const **Finger::ANSI2004View** &view)
Insert a finger view to the record at a specific position.
- uint64_t **insertView** (const **Finger::ANSI2004View** &view, const uint64_t viewNumber)
Insert a finger view to the record at a specific position.
- uint64_t **updateView** (const **Finger::ANSI2004View** &view, const uint64_t viewNumber)
Update an entire finger view.
- void **removeView** (const uint64_t viewNumber)
Remove a view from the record.

- void **isolateView** (const uint64_t viewNumber)
Isolate a finger view from the record.
- std::vector< **BiometricEvaluation::Feature::INCITSMinutiae** > **getMinutia** () const
Obtain the INCITSMinutiae for all finger views.
- **BiometricEvaluation::Feature::INCITSMinutiae** **getMinutia** (uint32_t viewNumber) const
Obtain the INCITSMinutiae for a finger view.
- void **setMinutia** (const std::vector< **BiometricEvaluation::Feature::INCITSMinutiae** > &minutia)
Alter the minutia for every finger view.
- void **setMinutia** (uint32_t viewNumber, const **BiometricEvaluation::Feature::INCITSMinutiae** &minutia)
Alter the minutia for a single finger view.
- **BiometricEvaluation::Memory::uint8Array** **getFMR** () const
Obtain an ANSI/INCITS 378-2004 record.
- uint64_t **getNumFingerViews** () const
Obtain the number of finger views in this finger minutia record.

Protected Member Functions

- uint64_t **getFMRLength** () const
*Obtain the size of FMR that will be written by **getFMR**() (p. 207).*
- uint64_t **getEDBLength** () const
*Obtain the size of EDB that will be written by **getFMR**() (p. 207).*

G.13.1 Detailed Description

All finger views from a single finger minutiae record

G.13.2 Constructor & Destructor Documentation

G.13.2.1 ANSI2004Record() [1/3]

```
BiometricEvaluation::DataInterchange::ANSI2004Record::ANSI2004Record (
    const BiometricEvaluation::Memory::uint8Array & fmr,
    const BiometricEvaluation::Memory::uint8Array & fir )
```

ANSI2004Record (p. 205) constructor using a pair of finger minutia and image records.

One or both records can be the empty array. The data obtained from an empty record will be set to the zero-value.

Parameters

<i>fmr</i>	Finger (p. 115) minutia record.
<i>fir</i>	Finger (p. 115) image record.

G.13.2.2 ANSI2004Record() [2/3]

```
BiometricEvaluation::DataInterchange::ANSI2004Record::ANSI2004Record (
    const std::string & fmrPath,
    const std::string & firPath )
```

ANSI2004Record (p. 205) constructor using a pair of finger minutia and image records.

One or both records can be the empty string. The data obtained from an empty record will be set to the zero-value.

Parameters

<i>fmr</i>	Path to a finger minutia record.
<i>fir</i>	Path to a finger image record.

G.13.2.3 ANSI2004Record() [3/3]

```
BiometricEvaluation::DataInterchange::ANSI2004Record::ANSI2004Record (
    const std::initializer_list< BiometricEvaluation::Finger::ANSI2004View > & views
)
```

ANSI2004Record (p. 205) constructor using a set of finger view records.

Parameters

<i>views</i>	ANSI2004View objects.
--------------	-----------------------

G.13.3 Member Function Documentation

G.13.3.1 getEDBLength()

```
uint64_t BiometricEvaluation::DataInterchange::ANSI2004Record::getEDBLength ( ) const [protected]
```

Obtain the size of EDB that will be written by **getFMR()** (p. 207).

Even if unmodified after reading a record, this value may be different than expected because ANSI2004↵View does not support reading proprietary extended data blocks.

Returns

Size of EDB that will be returned from **getFMR()** (p. 207).

getFMR() (p. 207)

G.13.3.2 getFMR()

```
BiometricEvaluation::Memory::uint8Array BiometricEvaluation::DataInterchange::ANSI2004Record↵::getFMR ( ) const
```

Obtain an ANSI/INCITS 378-2004 record.

Note

Reflects the current state of the object contained within.

Returns

A well-formed ANSI/INCITS 378-2004 record.

G.13.3.3 getFMRLength()

```
uint64_t BiometricEvaluation::DataInterchange::ANSI2004Record::getFMRLength ( ) const [protected]
```

Obtain the size of FMR that will be written by **getFMR()** (p. 207).

Even if unmodified after reading a record, this value may be different than expected because ANSI2004↵
View does not support reading proprietary extended data blocks.

Returns

Size of FMR that will be returned from **getFMR()** (p. 207).

getFMR() (p. 207) **getEDBLength()** (p. 207)

G.13.3.4 getMinutiae() [1/2]

```
std::vector< BiometricEvaluation::Feature::INCITSMinutiae> BiometricEvaluation::DataInterchange↵  
::ANSI2004Record::getMinutiae ( ) const
```

Obtain the INCITSMinutiae for all finger views.

Returns

Vector of INCITSMinutiae for all finger views in this record.

G.13.3.5 getMinutiae() [2/2]

```
BiometricEvaluation::Feature::INCITSMinutiae BiometricEvaluation::DataInterchange::ANSI2004↵  
Record::getMinutiae (
```

```
uint32_t viewNumber ) const
```

Obtain the INCITSMinutiae for a finger view.

Parameters

<i>viewNumber</i>	1-based finger view whose minutia will be returned.
-------------------	---

Returns

INCITSMinutiae for finger view viewNumber.

G.13.3.6 getNumFingerViews()

```
uint64_t BiometricEvaluation::DataInterchange::ANSI2004Record::getNumFingerViews ( ) const
```

Obtain the number of finger views in this finger minutia record.

Returns

Number of finger views, as iterated over when constructing this object.

G.13.3.7 getView()

Finger::ANSI2004View BiometricEvaluation::DataInterchange::ANSI2004Record::getView (
const uint64_t viewNumber) const
Obtain an ANSI2004View.

Parameters

viewNumber	The position of the view to obtain.
------------	-------------------------------------

Returns

ANSI2004View for view number viewNumber.

Exceptions

Error::ObjectDoesNotExist (p. 457)	viewNumber does not exist.
---	----------------------------

G.13.3.8 insertView() [1/2]

uint64_t BiometricEvaluation::DataInterchange::ANSI2004Record::insertView (
const **Finger::ANSI2004View** & view)
Insert a finger view to the record at a specific position.

Parameters

view	Finger (p. 115) view to add.
------	-------------------------------------

Returns

View (p. 164) number for view in this record.

G.13.3.9 insertView() [2/2]

uint64_t BiometricEvaluation::DataInterchange::ANSI2004Record::insertView (
const **Finger::ANSI2004View** & view,
const uint64_t viewNumber)
Insert a finger view to the record at a specific position.

Parameters

view	Finger (p. 115) view to add.
viewNumber	View (p. 164) number to assign to this view.

Returns

The view number.

Exceptions

<i>BiometricEvaluation::Error::StrategyError</i> (p. 567)	viewNumber is not valid.
---	--------------------------

G.13.3.10 isolateView()

```
void BiometricEvaluation::DataInterchange::ANSI2004Record::isolateView (  
    const uint64_t viewNumber )
```

Isolate a finger view from the record.

Parameters

<i>viewNumber</i>	The view number to isolate.
-------------------	-----------------------------

Exceptions

<i>BiometricEvaluation::Error::ObjectDoesNotExist</i> (p. 457)	viewNumber does not exist.
--	----------------------------

Note

The remaining view becomes view 1.

G.13.3.11 removeView()

```
void BiometricEvaluation::DataInterchange::ANSI2004Record::removeView (  
    const uint64_t viewNumber )
```

Remove a view from the record.

Parameters

<i>viewNumber</i>	The view number to remove.
-------------------	----------------------------

Exceptions

<i>BiometricEvaluation::Error::ObjectDoesNotExist</i> (p. 457)	viewNumber does not exist.
--	----------------------------

Note

All views will be renumbered after removal.

G.13.3.12 setMinutia() [1/2]

```
void BiometricEvaluation::DataInterchange::ANSI2004Record::setMinutia (
    const std::vector< BiometricEvaluation::Feature::INCITSMinutiae > & minutia )
```

Alter the minutia for every finger view.

Parameters

<i>minutia</i>	A vector of INCITSMinutiae for each finger view.
----------------	--

Exceptions

Error::StrategyError (p. 567)	Size of minutia does not equal the number of finger views in this record.
---	---

G.13.3.13 setMinutia() [2/2]

```
void BiometricEvaluation::DataInterchange::ANSI2004Record::setMinutia (
    uint32_t viewNumber,
    const BiometricEvaluation::Feature::INCITSMinutiae & minutia )
```

Alter the minutia for a single finger view.

Parameters

<i>viewNumber</i>	1-based finger view whose minutia will be replaced.
<i>minutia</i>	INCITSMinutiae for finger view viewNumber.

Exceptions

Error::StrategyError (p. 567)	View (p. 164) number is invalid for this finger record.
---	---

G.13.3.14 updateView()

```
uint64_t BiometricEvaluation::DataInterchange::ANSI2004Record::updateView (
    const Finger::ANSI2004View & view,
    const uint64_t viewNumber )
```

Update an entire finger view.

Parameters

<i>view</i>	Updated finger view.
<i>viewNumber</i>	View (p. 164) number replaced by view.

Returns

The view number.

Exceptions

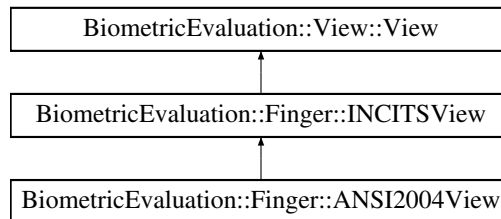
<i>BiometricEvaluation::Error::StrategyError</i> (p. 567)	viewNumber is not valid.
--	--------------------------

G.14 BiometricEvaluation::Finger::ANSI2004View Class Reference

A class to represent single finger view and derived information.

```
#include <be_finger_ansi2004view.h>
```

Inheritance diagram for BiometricEvaluation::Finger::ANSI2004View:



Public Member Functions

- **ANSI2004View** ()
Construct an empty ANSI finger view.
- **ANSI2004View** (const std::string &fmrFilename, const std::string &firFilename, const uint32_t view←
Number)
Construct an ANSI-2004 finger view from records contained in files.
- **ANSI2004View** (const **Memory::uint8Array** &fmrBuffer, const **Memory::uint8Array** &firBuffer,
const uint32_t viewNumber)
Construct an ANSI-2004 finger view from records contained in buffers.

Protected Member Functions

- void **readFMRHeader** (**Memory::IndexedBuffer** &buf)
- void **readCoreDeltaData** (**Memory::IndexedBuffer** &buf, uint32_t dataLength, Feature::CorePoint←
Set &cores, Feature::DeltaPointSet &deltas)
Read the core points data.

Static Protected Attributes

- static const uint32_t **BASE_SPEC_VERSION** = 0x20323000

Additional Inherited Members

G.14.1 Detailed Description

A class to represent single finger view and derived information.

A **Finger::ANSI2004View** (p. 212) object represents a finger view from a INCITS/ANSI-2004 **Finger** (p. 115) Minutiae Record.

G.14.2 Constructor & Destructor Documentation

G.14.2.1 ANSI2004View() [1/2]

```
BiometricEvaluation::Finger::ANSI2004View::ANSI2004View (
    const std::string & fmrFilename,
    const std::string & firFilename,
    const uint32_t viewNumber )
```

Construct an ANSI-2004 finger view from records contained in files.

A view can be constructed from a single record, with information missing as appropriate. For example, if a view is constructed with just the minutiae record, no image would be part of the view. However, the image size etc. would be present because that information is also present in the minutiae record.

Parameters

in	<i>fmrFilename</i>	The name of the file containing the complete finger minutiae record.
in	<i>firFilename</i>	The name of the file containing the complete finger image record.
in	<i>viewNumber</i>	The finger view number to use.

G.14.2.2 ANSI2004View() [2/2]

```
BiometricEvaluation::Finger::ANSI2004View::ANSI2004View (
    const Memory::uint8Array & fmrBuffer,
    const Memory::uint8Array & firBuffer,
    const uint32_t viewNumber )
```

Construct an ANSI-2004 finger view from records contained in buffers.

A view can be constructed from a single record, with information missing as appropriate. For example, if a view is constructed with just the minutiae record, no image would be part of the view. However, the image size etc. would be present because that information is also present in the minutiae record.

Parameters

in	<i>fmrBuffer</i>	The buffer containing the complete finger minutiae record.
in	<i>firBuffer</i>	The buffer containing the complete finger image record.
in	<i>viewNumber</i>	The finger view number to use.

G.14.3 Member Function Documentation

G.14.3.1 readCoreDeltaData()

```
void BiometricEvaluation::Finger::ANSI2004View::readCoreDeltaData (
    Memory::IndexedBuffer & buf,
    uint32_t dataLength,
    Feature::CorePointSet & cores,
    Feature::DeltaPointSet & deltas ) [protected], [virtual]
```

Read the core points data.

This method must be overridden by derived classes to read data in a specific record format.

Parameters

in, out	<i>buf</i>	The indexed buffer containing the record data. On function exit, the buffer index will be set to the location after the last core point data item.
out	<i>cores</i>	The set of core data items.
out	<i>deltas</i>	The set of delta data items.
in	<i>dataLength</i>	The length of the entire ridge count data block.

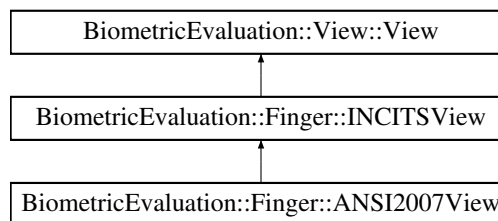
Implements **BiometricEvaluation::Finger::INCITSView** (p. 390).

G.15 BiometricEvaluation::Finger::ANSI2007View Class Reference

A class to represent single finger view and derived information.

```
#include <be_finger_ansi2007view.h>
```

Inheritance diagram for BiometricEvaluation::Finger::ANSI2007View:



Public Member Functions

- **ANSI2007View** (const std::string &fmrFilename, const std::string &firFilename, const uint32_t view↔ Number)

Construct an ANSI-2007 finger view from records contained in files.

- **ANSI2007View** (const **Memory::uint8Array** &fmrBuffer, const **Memory::uint8Array** &firBuffer, const uint32_t viewNumber)

Construct an ANSI-2007 finger view from records contained in buffers.

Protected Member Functions

- void **readFMRHeader** (**Memory::IndexedBuffer** &buf)
- void **readFVMR** (**Memory::IndexedBuffer** &buf)
- void **readCoreDeltaData** (**Memory::IndexedBuffer** &buf, uint32_t dataLength, Feature::CorePointSet &cores, Feature::DeltaPointSet &deltas)

Read the core points data.

Static Protected Attributes

- static const uint32_t **BASE_SPEC_VERSION** = 0x30333000

Additional Inherited Members

G.15.1 Detailed Description

A class to represent single finger view and derived information.

A **Finger::ANSI2007View** (p. 214) object represents a finger view from a INCITS/ANSI-2007 **Finger** (p. 115) Minutiae Record.

G.15.2 Constructor & Destructor Documentation

G.15.2.1 ANSI2007View() [1/2]

```
BiometricEvaluation::Finger::ANSI2007View::ANSI2007View (
    const std::string & fmrFilename,
    const std::string & firFilename,
    const uint32_t viewNumber )
```

Construct an ANSI-2007 finger view from records contained in files.

A view can be constructed from a single record, with information missing as appropriate. For example, if a view is constructed with just the minutiae record, no image would be part of the view. However, the image size etc. would be present because that information is also present in the minutiae record.

Parameters

in	<i>fmrFilename</i>	The name of the file containing the complete finger minutiae record.
in	<i>firFilename</i>	The name of the file containing the complete finger image record.
in	<i>viewNumber</i>	The finger view number to use.

Exceptions

Error::DataError (p. 295)	Invalid record format.
----------------------------------	------------------------

G.15.2.2 ANSI2007View() [2/2]

```
BiometricEvaluation::Finger::ANSI2007View::ANSI2007View (
```

```
const Memory::uint8Array & fmrBuffer,
const Memory::uint8Array & firBuffer,
const uint32_t viewNumber )
```

Construct an ANSI-2007 finger view from records contained in buffers.

A view can be constructed from a single record, with information missing as appropriate. For example, if a view is constructed with just the minutiae record, no image would be part of the view. However, the image size etc. would be present because that information is also present in the minutiae record.

Parameters

in	<i>fmrBuffer</i>	The buffer containing the complete finger minutiae record.
in	<i>firBuffer</i>	The buffer containing the complete finger image record.
in	<i>viewNumber</i>	The finger view number to use.

Exceptions

Error::DataError (p. 295)	Invalid record format.
----------------------------------	------------------------

G.15.3 Member Function Documentation

G.15.3.1 readCoreDeltaData()

```
void BiometricEvaluation::Finger::ANSI2007View::readCoreDeltaData (
    Memory::IndexedBuffer & buf,
    uint32_t dataLength,
    Feature::CorePointSet & cores,
    Feature::DeltaPointSet & deltas ) [protected], [virtual]
```

Read the core points data.

This method must be overridden by derived classes to read data in a specific record format.

Parameters

in, out	<i>buf</i>	The indexed buffer containing the record data. On function exit, the buffer index will be set to the location after the last core point data item.
out	<i>cores</i>	The set of core data items.
out	<i>deltas</i>	The set of delta data items.
in	<i>dataLength</i>	The length of the entire ridge count data block.

Implements **BiometricEvaluation::Finger::INCITSView** (p. 390).

G.16 BiometricEvaluation::Device::Smartcard::APDU Class Reference

Public Attributes

- uint8_t *cla*

- uint8_t **ins**
- uint8_t **p1**
- uint8_t **p2**
- uint16_t **lc**
- uint8_t **nc** [MAX_NC_SIZE]
- uint16_t **le**
- uint8_t **field_mask**

Static Public Attributes

- static const int **FIELD_LC** {0x00000001}
- static const int **FIELD_LE** {0x00000002}
- static const int **FLEN_CLA** {1}
- static const int **FLEN_INS** {1}
- static const int **FLEN_P1** {1}
- static const int **FLEN_P2** {1}
- static const int **FLEN_LC_SHORT** {1}
- static const int **FLEN_LC_EXTENDED** {3}
- static const int **FLEN_LE_SHORT** {1}
- static const int **FLEN_LE_EXTENDED** {3}
- static const int **FLEN_TRAILER** {2}
- static const int **FLAG_CLA_NOCHAIN** {0x00}
- static const int **FLAG_CLA_CHAIN** {0x10}
- static const int **MAX_NC_SIZE** {0xFFFF}
- static const int **MAX_LE_SIZE** {0xFFFF}
- static const int **MAX_SHORT_LC** {255}
- static const int **MAX_SHORT_LE** {255}
- static const int **HEADER_LEN** {FLEN_CLA + FLEN_INS + FLEN_P1 + FLEN_P2}
- static const int **NORMAL_COMPLETE** {0x90}
- static const int **NORMAL_CHAINING** {0x61}
- static const int **WARN_NVM_UNCHANGED** {0x62}
- static const int **WARN_NVM_CHANGED** {0x63}
- static const int **EXEC_ERR_NVM_UNCHANGED** {0x64}
- static const int **EXEC_ERR_NVM_CHANGED** {0x65}
- static const int **EXEC_ERR_SECURITY** {0x66}
- static const int **CHECK_ERR_WRONG_LENGTH** {0x67}
- static const int **CHECK_ERR_CLA_FUNCTION** {0x68}
- static const int **CHECK_ERR_CMD_NOT_ALLOWED** {0x69}
- static const int **CHECK_ERR_WRONG_PARAM_QUAL** {0x6A}
- static const int **CHECK_ERR_WRONG_PARAM** {0x6B}
- static const int **CHECK_ERR_WRONG_LE** {0x6C}
- static const int **CHECK_ERR_INVALID_INS** {0x6D}
- static const int **CHECK_ERR_CLA_UNSUPPORTED** {0x6E}
- static const int **CHECK_ERR_NO_DIAGNOSIS** {0x6F}
- static const int **NO_INFORMATION** {0x00}
- static const int **INCORRECT_PARAMETERS** {0x80}
- static const int **FUNCTION_NOT_SUPPORTED** {0x81}
- static const int **FILE_OR_APP_NOT_FOUND** {0x82}
- static const int **RETRY_COUNTER_MASK** {0x0F}
- static const int **RETRY_COUNTER_INDICATOR** {0xC0}
- static const int **RETRY_COUNTER_INDICATOR_MASK** {0xF0}
- static const int **RETRY_COUNTER_MAX** {15}

G.16.1 Member Data Documentation

G.16.1.1 cla

```
uint8_t BiometricEvaluation::Device::Smartcard::APDU::cla
```

The class byte

G.16.1.2 FIELD_LC

```
const int BiometricEvaluation::Device::Smartcard::APDU::FIELD_LC {0x00000001} [static]
```

Lc field is present; Implies Nc present as well

G.16.1.3 FIELD_LE

```
const int BiometricEvaluation::Device::Smartcard::APDU::FIELD_LE {0x00000002} [static]
```

Le field is present, response data expected

G.16.1.4 field_mask

```
uint8_t BiometricEvaluation::Device::Smartcard::APDU::field_mask
```

Mask of optional fields; use field bit masks

G.16.1.5 ins

```
uint8_t BiometricEvaluation::Device::Smartcard::APDU::ins
```

Instruction byte

G.16.1.6 lc

```
uint16_t BiometricEvaluation::Device::Smartcard::APDU::lc
```

Lc, length of the Nc field

G.16.1.7 le

```
uint16_t BiometricEvaluation::Device::Smartcard::APDU::le
```

Le, expected response length

G.16.1.8 nc

```
uint8_t BiometricEvaluation::Device::Smartcard::APDU::nc[MAX_NC_SIZE]
```

Nc, command data

G.16.1.9 p1

```
uint8_t BiometricEvaluation::Device::Smartcard::APDU::p1
```

P1 byte

G.16.1.10 p2

```
uint8_t BiometricEvaluation::Device::Smartcard::APDU::p2
```

P2 byte

G.17 BiometricEvaluation::Device::Smartcard::APDUException Struct Reference

Exception thrown when a command fails.
#include <be_device_smartcard.h>

Public Member Functions

- **APDUException** ()=default
- **APDUException** (const **APDUResponse** & response, const **Memory::uint8Array** & apdu)

Public Attributes

- **APDUResponse** response
- **Memory::uint8Array** apdu

G.17.1 Detailed Description

Exception thrown when a command fails.
This object is thrown when the status words returned from the card indicate an error occurred when a command was sent to the card. Any data returned by the card and the **APDU** (p. 216) that was sent are contained within this object.

G.17.2 Constructor & Destructor Documentation

G.17.2.1 APDUException() [1/2]

BiometricEvaluation::Device::Smartcard::APDUException::APDUException () [default]
Constructor.

G.17.2.2 APDUException() [2/2]

BiometricEvaluation::Device::Smartcard::APDUException::APDUException (
const **APDUResponse** & response,
const **Memory::uint8Array** & apdu)
Constructor.

Parameters

<i>repines</i>	The partial response data and status
<i>apdu</i>	The raw APDU (p. 216) that was sent.

G.17.3 Member Data Documentation

G.17.3.1 apdu

Memory::uint8Array BiometricEvaluation::Device::Smartcard::APDUException::apdu

The raw **APDU** (p. 216) that was sent.

G.17.3.2 response

APDUResponse BiometricEvaluation::Device::Smartcard::APDUException::response

The partial response data and status words from the failed command.

G.18 BiometricEvaluation::Device::Smartcard::APDUResponse Struct Reference

The data and status words returned by the card in response to a command.

```
#include <be_device_smartcard.h>
```

Public Member Functions

- **APDUResponse** ()=default
- **APDUResponse** (const **Memory::uint8Array** & **data**, const uint8_t **sw1**, const uint8_t **sw2**)

Public Attributes

- uint8_t **sw1** {0}
- uint8_t **sw2** {0}
- **Memory::uint8Array** **data**

G.18.1 Detailed Description

The data and status words returned by the card in response to a command.

G.18.2 Constructor & Destructor Documentation

G.18.2.1 APDUResponse() [1/2]

```
BiometricEvaluation::Device::Smartcard::APDUResponse::APDUResponse ( ) [default]
```

Constructor

G.18.2.2 APDUResponse() [2/2]

```
BiometricEvaluation::Device::Smartcard::APDUResponse::APDUResponse (
    const Memory::uint8Array & data,
    const uint8_t sw1,
    const uint8_t sw2 )
```

Constructor

Parameters

<i>data</i>	The response data; may be empty.
<i>sw1</i>	Status word one.
<i>sw2</i>	Status word two.

G.18.3 Member Data Documentation

G.18.3.1 data

Memory: `uint8Array` `BiometricEvaluation::Device::Smartcard::APDUResponse::data`
 The response data, possibly incomplete

G.18.3.2 sw1

`uint8_t` `BiometricEvaluation::Device::Smartcard::APDUResponse::sw1` {0}
 status word one

G.18.3.3 sw2

`uint8_t` `BiometricEvaluation::Device::Smartcard::APDUResponse::sw2` {0}
 status word two

G.19 BiometricEvaluation::Framework::API< T > Class Template Reference

A convenient way to execute biometric technology evaluation **API** (p. 221) methods safely.

```
#include <be-framework_api.h>
```

Classes

- class **Result**

Public Member Functions

- **API** ()
- **Result** **call** (const std::function< T(void)> &operation, const std::function< void(const **Result** &)> &success={}, const std::function< void(const **Result** &)> &failure={}, const bool rethrowExceptions=false)
*Invoke an operation. Invoking operations within this method implicitly wraps the operation in a SignalManager, Watchdog, and Timer, and follows evaluation best practices for calling an **API** (p. 221) operation.*
- std::shared_ptr< **BiometricEvaluation::Time::Timer** > **getTimer** () noexcept
Obtain the timer object.
- std::shared_ptr< **BiometricEvaluation::Time::Watchdog** > **getWatchdog** () noexcept
Obtain the watchdog timer object.
- std::shared_ptr< **BiometricEvaluation::Error::SignalManager** > **getSignalManager** () noexcept
Obtain the signal manager object.

G.19.1 Detailed Description

```
template<typename T>
class BiometricEvaluation::Framework::API< T >
```

A convenient way to execute biometric technology evaluation **API** (p. 221) methods safely.

Note

One **API** (p. 221) object should be instantiated per process/thread.

G.19.2 Constructor & Destructor Documentation

G.19.2.1 API()

```
template<typename T >
BiometricEvaluation::Framework::API< T >:: API ( )
    Constructor
```

G.19.3 Member Function Documentation

G.19.3.1 call()

```
template<typename T >
BiometricEvaluation::Framework::API< T >:: Result BiometricEvaluation::Framework::API< T >↔
::call (
    const std::function< T(void)> & operation,
    const std::function< void(const Result &)> & success = {},
    const std::function< void(const Result &)> & failure = {},
    const bool rethrowExceptions = false )
```

Invoke an operation. Invoking operations within this method implicitly wraps the operation in a Signal↔ Manager, Watchdog, and Timer, and follows evaluation best practices for calling an **API** (p. 221) operation.

Parameters

<i>operation</i>	A reference to a function that returns a Status (p. 565). (i.e., an API (p. 221) method).
<i>success</i>	Operations invoked if operation returns.
<i>failure</i>	Operations invoked if we abort the operation.
<i>rethrowExceptions</i>	Whether or not to rethrow an exception caught from operation.

Returns

Analytics about the return of operation.

Exceptions

...	Exceptions raised from operation, if caught, are rethrown when rethrowExceptions is true.
-----	---

Note

success is called and `currentState == APICurrentState::Completed` (p. 118) if operation returns, regardless of the Code of operation's **Status** (p. 565).
 Exceptions caught are rethrown after calling `failure()`.

G.19.3.2 getSignalManager()

```
template<typename T >
std::shared_ptr< BiometricEvaluation::Error::SignalManager> BiometricEvaluation::Framework::API<
::API< T >::getSignalManager ( ) [inline], [noexcept]
```

Obtain the signal manager object.

Returns

Signal manager object.

G.19.3.3 getTimer()

```
template<typename T >
std::shared_ptr< BiometricEvaluation::Time::Timer> BiometricEvaluation::Framework::API< T >::getTimer ( ) [inline], [noexcept]
```

Obtain the timer object.

Returns

Timer object.

G.19.3.4 getWatchdog()

```
template<typename T >
std::shared_ptr< BiometricEvaluation::Time::Watchdog> BiometricEvaluation::Framework::API<
T >::getWatchdog ( ) [inline], [noexcept]
```

Obtain the watchdog timer object.

Returns

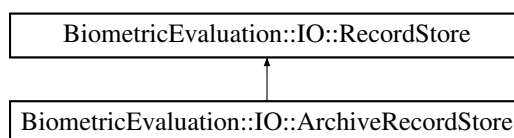
Watchdog timer object.

G.20 BiometricEvaluation::IO::ArchiveRecordStore Class Reference

This class implements the **IO::RecordStore** (p. 504) interface by storing data items in single file, with an associated manifest file.

```
#include <be_io_archiverecstore.h>
```

Inheritance diagram for BiometricEvaluation::IO::ArchiveRecordStore:



Public Member Functions

- **ArchiveRecordStore** (const std::string &pathname, const std::string &description)
- **ArchiveRecordStore** (const std::string &pathname, **IO::Mode** mode= **IO::Mode::ReadOnly**)
- **~ArchiveRecordStore** ()
- void **sync** () const override
- void **insert** (const std::string &key, const void *const data, const uint64_t size) override
- void **remove** (const std::string &key) override
- **Memory::uint8Array read** (const std::string &key) const override

Read a complete record from a store.
- uint64_t **length** (const std::string &key) const override
- void **flush** (const std::string &key) const override
- **RecordStore::Record sequence** (int cursor= **BE_RECSTORE_SEQ_NEXT**) override

*Sequence through a **RecordStore** (p. 504), returning the key/data pairs.*
- std::string **sequenceKey** (int cursor= **BE_RECSTORE_SEQ_NEXT**) override

*Sequence through a **RecordStore** (p. 504), returning the key.*
- void **setCursorAtKey** (const std::string &key) override
- void **move** (const std::string &pathname) override

*Move the **RecordStore** (p. 504).*
- uint64_t **getSpaceUsed** () const override

Obtain real storage utilization.
- unsigned int **getCount** () const override
- std::string **getPathname** () const override
- std::string **getDescription** () const override
- void **changeDescription** (const std::string &description) override
- bool **needsVacuum** ()
- std::string **getArchiveName** () const
- std::string **getManifestName** () const
- **ArchiveRecordStore** (const **ArchiveRecordStore** &)=delete
- **ArchiveRecordStore & operator=** (const **ArchiveRecordStore** &)=delete

Static Public Member Functions

- static bool **needsVacuum** (const std::string &pathname)
- static void **vacuum** (const std::string &pathname)

Static Public Attributes

- static const std::string **MANIFEST_FILE_NAME**
- static const std::string **ARCHIVE_FILE_NAME**
- static const long **OFFSET_RECORD_REMOVED** = -1

Additional Inherited Members

G.20.1 Detailed Description

This class implements the **IO::RecordStore** (p. 504) interface by storing data items in single file, with an associated manifest file.

Archives consist of binary records written back to back of each other. To pull information out of an archive, a manifest file is written in the same directory as the archive file.

Each record is assigned a string key, which will be required for retrieving the data. As the data is written, a plain text entry is entered into the manifest in the format:

key offset size

where offset is the offset into the archive file key's data chunk resides and size is the length of key's data chunk.

By default, information is not removed when updated in the archive, rather the old information is ignored. Therefore, it is possible to have multiple entries in the manifest for one key. The last entry for the key is considered accurate. If the last offset for a key is **ARCHIVE_RECORD_REMOVED**, the information is treated as unavailable.

G.20.2 Constructor & Destructor Documentation

G.20.2.1 ArchiveRecordStore() [1/2]

```
BiometricEvaluation::IO::ArchiveRecordStore::ArchiveRecordStore (
    const std::string & pathname,
    const std::string & description )
```

Create a new **ArchiveRecordStore** (p. 223), read/write mode.

Parameters

in	<i>pathname</i>	The directory where the store is to be created.
in	<i>description</i>	The store's description.

Exceptions

Error::ObjectExists (p. 458)	The store already exists.
Error::StrategyError (p. 567)	An error occurred when accessing the underlying file system.

G.20.2.2 ArchiveRecordStore() [2/2]

```
BiometricEvaluation::IO::ArchiveRecordStore::ArchiveRecordStore (
    const std::string & pathname,
    IO::Mode mode = IO::Mode::ReadOnly )
```

Open an existing **ArchiveRecordStore** (p. 223).

Parameters

in	<i>pathname</i>	The path name of the store.
in	<i>mode</i>	Open mode, read-only or read-write.

Exceptions

Error::ObjectDoesNotExist (p. 457)	The store does not exist.
Error::StrategyError (p. 567)	An error occurred when accessing the underlying file system.

G.20.2.3 ~ArchiveRecordStore()

BiometricEvaluation::IO::ArchiveRecordStore::~~ArchiveRecordStore ()
Destructor.

G.20.3 Member Function Documentation**G.20.3.1** changeDescription()

```
void BiometricEvaluation::IO::ArchiveRecordStore::changeDescription (
    const std::string & description ) [override], [virtual]
```

Change the description of the **RecordStore** (p. [504](#)).

Parameters

in	<i>description</i>	The new description.
----	--------------------	----------------------

Exceptions

Error::StrategyError (p. 567)	An error occurred when using the underlying storage system.
---	---

Implements **BiometricEvaluation::IO::RecordStore** (p. [506](#)).

G.20.3.2 flush()

```
void BiometricEvaluation::IO::ArchiveRecordStore::flush (
    const std::string & key ) const [override], [virtual]
```

Commit the record's data to storage.

Parameters

in	<i>key</i>	The key of the record to be flushed.
----	------------	--------------------------------------

Exceptions

Error::ObjectDoesNotExist (p. 457)	A record for the key does not exist.
Error::StrategyError (p. 567)	An error occurred when using the underlying storage system.

Implements **BiometricEvaluation::IO::RecordStore** (p. 508).

G.20.3.3 getArchiveName()

```
std::string BiometricEvaluation::IO::ArchiveRecordStore::getArchiveName ( ) const
```

Obtain the name of the file storing the data for this store.

Returns

Path to archive file.

G.20.3.4 getCount()

```
unsigned int BiometricEvaluation::IO::ArchiveRecordStore::getCount ( ) const [override], [virtual]
```

Obtain the number of items in the **RecordStore** (p. 504).

Returns

The number of items in the **RecordStore** (p. 504).

Implements **BiometricEvaluation::IO::RecordStore** (p. 508).

G.20.3.5 getDescription()

```
std::string BiometricEvaluation::IO::ArchiveRecordStore::getDescription ( ) const [override],  
[virtual]
```

Obtain a textual description of the **RecordStore** (p. 504).

Returns

The **RecordStore** (p. 504)'s description.

Implements **BiometricEvaluation::IO::RecordStore** (p. 508).

G.20.3.6 getManifestName()

```
std::string BiometricEvaluation::IO::ArchiveRecordStore::getManifestName ( ) const
```

Obtain the name of the file storing the manifest data data for this store.

Returns

Path to manifest file.

G.20.3.7 getPathname()

```
std::string BiometricEvaluation::IO::ArchiveRecordStore::getPathname ( ) const [override], [virtual]
```

Return the path name of the **RecordStore** (p. 504).

Returns

Where in the file system the **RecordStore** (p. 504) is located.

Implements **BiometricEvaluation::IO::RecordStore** (p. 509).

G.20.3.8 `getSpaceUsed()`

```
uint64_t BiometricEvaluation::IO::ArchiveRecordStore::getSpaceUsed ( ) const [override], [virtual]
```

Obtain real storage utilization.

The amount of disk space used, for example. This is the actual space allocated by the underlying storage mechanism, in bytes.

Returns

The amount of backing storage used by the **RecordStore** (p. 504).

Exceptions

Error::StrategyError (p. 567)	An error occurred when using the underlying storage system.
--------------------------------------	---

Implements **BiometricEvaluation::IO::RecordStore** (p. 509).

G.20.3.9 `insert()`

```
void BiometricEvaluation::IO::ArchiveRecordStore::insert (
    const std::string & key,
    const void *const data,
    const uint64_t size ) [override], [virtual]
```

Insert a record into the store.

Parameters

in	<i>key</i>	The key of the record to be inserted.
in	<i>data</i>	The data for the record.
in	<i>size</i>	The size of the record, in bytes.

Exceptions

Error::ObjectExists (p. 458)	A record with the given key is already present.
Error::StrategyError (p. 567)	The RecordStore (p. 504) is opened read-only, or an error occurred when using the underlying storage system.

Implements **BiometricEvaluation::IO::RecordStore** (p. 510).

G.20.3.10 `length()`

```
uint64_t BiometricEvaluation::IO::ArchiveRecordStore::length (
    const std::string & key ) const [override], [virtual]
```

Return the length of a record.

Parameters

in	<i>key</i>	The key of the record.
----	------------	------------------------

Returns

The record length.

Exceptions

Error::ObjectDoesNotExist (p. 457)	A record for the key does not exist.
Error::StrategyError (p. 567)	An error occurred when using the underlying storage system.

Implements **BiometricEvaluation::IO::RecordStore** (p. 510).

G.20.3.11 move()

```
void BiometricEvaluation::IO::ArchiveRecordStore::move (
    const std::string & pathname ) [override], [virtual]
```

Move the **RecordStore** (p. 504).

The **RecordStore** (p. 504) can be moved to a new path in the file system.

Parameters

in	<i>pathname</i>	The new path of the RecordStore (p. 504).
----	-----------------	--

Exceptions

Error::StrategyError (p. 567)	An error occurred when using the underlying storage system.
--------------------------------------	---

Implements **BiometricEvaluation::IO::RecordStore** (p. 511).

G.20.3.12 needsVacuum() [1/2]

```
bool BiometricEvaluation::IO::ArchiveRecordStore::needsVacuum ( )
```

See if the **ArchiveRecordStore** (p. 223) would benefit from calling **vacuum()** (p. 232) to remove deleted entries, since **vacuum()** (p. 232) is an expensive operation.

Returns

true if **vacuum()** (p. 232) would be beneficial false otherwise

G.20.3.13 needsVacuum() [2/2]

```
static bool BiometricEvaluation::IO::ArchiveRecordStore::needsVacuum (
    const std::string & pathname ) [static]
```

See if the **ArchiveRecordStore** (p. 223) would benefit from calling **vacuum()** (p. 232) to remove deleted entries, since **vacuum()** (p. 232) is an expensive operation.

Parameters

in	<i>pathname</i>	The path name of the existing RecordStore (p. 504).
----	-----------------	--

Exceptions

Error::ObjectDoesNotExist (p. 457)	A record with the given key does not exist.
Error::StrategyError (p. 567)	An error occurred when using the underlying storage system.

Returns

true if **vacuum()** (p. 232) would be beneficial false otherwise

G.20.3.14 read()

Memory::uint8Array BiometricEvaluation::IO::ArchiveRecordStore::read (
const std::string & key) const [override], [virtual]

Read a complete record from a store.

The AutoArray will be resized to match the size of the data.

Parameters

in	<i>key</i>	The key of the record to be read.
----	------------	-----------------------------------

Returns

The record associated with the key.

Exceptions

Error::ObjectDoesNotExist (p. 457)	A record for the key does not exist.
Error::StrategyError (p. 567)	An error occurred when using the underlying storage system.

Implements **BiometricEvaluation::IO::RecordStore** (p. 512).

G.20.3.15 remove()

void BiometricEvaluation::IO::ArchiveRecordStore::remove (
const std::string & key) [override], [virtual]

Remove a record from the store.

Parameters

in	<i>key</i>	The key of the record to be removed.
----	------------	--------------------------------------

Exceptions

Error::ObjectDoesNotExist (p. 457)	A record for the key does not exist.
Error::StrategyError (p. 567)	An error occurred when using the underlying storage system.

Implements **BiometricEvaluation::IO::RecordStore** (p. 513).

G.20.3.16 sequence()

```
RecordStore::Record BiometricEvaluation::IO::ArchiveRecordStore::sequence (
    int cursor = BE_RECSTORE_SEQ_NEXT ) [override], [virtual]
```

Sequence through a **RecordStore** (p. 504), returning the key/data pairs.

Sequencing means to start at some point in the store and return the record, then repeatedly calling the function to return the next record. The starting point is typically the first record, and is set to that when the **RecordStore** (p. 504) object is created. The starting point can be reset by calling this method with the cursor parameter set to **BE_RECSTORE_SEQ_START**.

Parameters

in	<i>cursor</i>	The location within the sequence of the key/data pair to return.
----	---------------	--

Returns

The record that is currently in sequence.

Exceptions

Error::ObjectDoesNotExist (p. 457)	End of sequencing.
Error::StrategyError (p. 567)	An error occurred when using the underlying storage system.

Implements **BiometricEvaluation::IO::RecordStore** (p. 515).

G.20.3.17 sequenceKey()

```
std::string BiometricEvaluation::IO::ArchiveRecordStore::sequenceKey (
    int cursor = BE_RECSTORE_SEQ_NEXT ) [override], [virtual]
```

Sequence through a **RecordStore** (p. 504), returning the key.

Sequencing means to start at some point in the store and return the key, then repeatedly calling the function to return the next key. The starting point is typically the first record, and is set to that when the **RecordStore** (p. 504) object is created. The starting point can be reset by calling this method with the cursor parameter set to **BE_RECSTORE_SEQ_START**.

Parameters

in	<i>cursor</i>	The location within the sequence of the key/data pair to return.
----	---------------	--

Returns

The key of the currently sequenced record.

Exceptions

<i>Error::ObjectDoesNotExist</i> (p. 457)	End of sequencing.
<i>Error::StrategyError</i> (p. 567)	An error occurred when using the underlying storage system.

Implements **BiometricEvaluation::IO::RecordStore** (p. 515).

G.20.3.18 setCursorAtKey()

```
void BiometricEvaluation::IO::ArchiveRecordStore::setCursorAtKey (
    const std::string & key ) [override], [virtual]
```

Set the sequence cursor to an arbitrary position within the **RecordStore** (p. 504), starting at key. Key will be the first record returned from the next call to **sequence()** (p. 231).

Parameters

in	key	The key of the record which will be returned by the first subsequent call to sequence() (p. 231).
----	-----	--

Exceptions

<i>Error::ObjectDoesNotExist</i> (p. 457)	A record for the key does not exist.
<i>Error::StrategyError</i> (p. 567)	An error occurred when using the underlying storage system.

Implements **BiometricEvaluation::IO::RecordStore** (p. 516).

G.20.3.19 sync()

```
void BiometricEvaluation::IO::ArchiveRecordStore::sync ( ) const [override], [virtual]
```

Synchronize the entire record store to persistent storage.

Exceptions

<i>Error::StrategyError</i> (p. 567)	An error occurred when using the underlying storage system.
---	---

Implements **BiometricEvaluation::IO::RecordStore** (p. 516).

G.20.3.20 vacuum()

```
static void BiometricEvaluation::IO::ArchiveRecordStore::vacuum (
    const std::string & pathname ) [static]
```

Remove deleted entries from the manifest and archive files to save space on disk.

Parameters

in	<i>pathname</i>	The pathname of the existing RecordStore (p. 504).
----	-----------------	---

Exceptions

Error::ObjectDoesNotExist (p. 457)	A record with the given key does not exist.
Error::StrategyError (p. 567)	An error occurred when using the underlying storage system.

Note

This is an expensive operation.

G.20.4 Member Data Documentation

G.20.4.1 ARCHIVE_FILE_NAME

```
const std::string BiometricEvaluation::IO::ArchiveRecordStore::ARCHIVE_FILE_NAME [static]
```

Name of the archive file on disk

G.20.4.2 MANIFEST_FILE_NAME

```
const std::string BiometricEvaluation::IO::ArchiveRecordStore::MANIFEST_FILE_NAME [static]
```

Name of the manifest file on disk

G.20.4.3 OFFSET_RECORD_REMOVED

```
const long BiometricEvaluation::IO::ArchiveRecordStore::OFFSET_RECORD_REMOVED = -1 [static]
```

Offset placeholder indicating a removed record

G.21 BiometricEvaluation::Memory::AutoArray< T > Class Template Reference

A C-style array wrapped in the facade of a C++ STL container.

```
#include <be_memory_autoarray.h>
```

Public Types

- using **value_type** = T
- using **size_type** = size_t
- using **iterator** = **AutoArrayIterator**< false, T >
- using **const_iterator** = **AutoArrayIterator**< true, T >
- using **reference** = T &
- using **const_reference** = const T &

Public Member Functions

- **operator T* ()**
Convert [AutoArray](#) (p. 233) to T array.
- **operator const T * () const**
Convert [AutoArray](#) (p. 233) to const T array.
- **reference operator[] (ptrdiff_t index)**
Subscripting operator overload with unchecked access.
- **const_reference operator[] (ptrdiff_t index) const**
Const subscripting operator overload with unchecked access.
- **reference at (ptrdiff_t index)**
Subscript into the [AutoArray](#) (p. 233) with checked access.
- **const_reference at (ptrdiff_t index) const**
Subscript into the [AutoArray](#) (p. 233) with checked access.
- **iterator begin ()**
Obtain an iterator to the beginning of the [AutoArray](#) (p. 233).
- **const_iterator begin () const**
Obtain an iterator to the beginning of the [AutoArray](#) (p. 233).
- **const_iterator cbegin () const**
Obtain an iterator to the beginning of the [AutoArray](#) (p. 233).
- **iterator end ()**
Obtain an iterator to the end of the [AutoArray](#) (p. 233).
- **const_iterator end () const**
Obtain an iterator to the end of the [AutoArray](#) (p. 233).
- **const_iterator cend () const**
Obtain an iterator to the end of the [AutoArray](#) (p. 233).
- **size_type size () const**
Obtain the number of accessible elements.
- void **resize (size_type new_size, bool free=false)**
Change the number of accessible elements.
- void **copy (const T *buffer)**
Deep-copy the contents of a buffer into this [AutoArray](#) (p. 233).
- void **copy (const T *buffer, size_type size)**
Deep-copy the contents of a buffer into this [AutoArray](#) (p. 233).
- std::vector< T > **to_vector () const**
Obtain a copy of elements in this [AutoArray](#) (p. 233) as a vector.
- **AutoArray (size_type size=0)**
Construct an [AutoArray](#) (p. 233).
- **AutoArray (const AutoArray & copy)**
Construct an [AutoArray](#) (p. 233).
- **AutoArray (AutoArray &&rvalue) noexcept**
Construct an [AutoArray](#) (p. 233).
- **AutoArray (std::initializer_list< T > ilist)**
Construct an [AutoArray](#) (p. 233).
- **AutoArray & operator= (const AutoArray &other)**

Copy assignment operator overload performing a deep copy.

- **AutoArray & operator=** (**AutoArray** &&other) noexcept(noexcept(std::swap(std::declval< **value_type** &>(), std::declval< **value_type** &>())) &&noexcept(std::swap(std::declval< **size_type** &>(), std::declval< **size_type** &>()))

Move assignment operator.

- **~AutoArray** ()

G.21.1 Detailed Description

```
template<class T>
class BiometricEvaluation::Memory::AutoArray< T >
```

A C-style array wrapped in the facade of a C++ STL container.

Objects of this type should be treated in the traditional manner for containers, where (size_type) construction creates an array of the given size, while {...} construction creates an array with the given elements.

Forward declaration.

G.21.2 Member Typedef Documentation

G.21.2.1 const_iterator

```
template<class T>
using BiometricEvaluation::Memory::AutoArray< T >:: const_iterator = AutoArrayIterator<true,
T>
    Const iterator of element
```

G.21.2.2 const_reference

```
template<class T>
using BiometricEvaluation::Memory::AutoArray< T >:: const_reference = const T&
    Const reference element
```

G.21.2.3 iterator

```
template<class T>
using BiometricEvaluation::Memory::AutoArray< T >:: iterator = AutoArrayIterator<false, T>
    Iterator of element
```

G.21.2.4 reference

```
template<class T>
using BiometricEvaluation::Memory::AutoArray< T >:: reference = T&
    Reference to element
```

G.21.2.5 size_type

```
template<class T>
using BiometricEvaluation::Memory::AutoArray< T >:: size_type = size_t
    Type of subscripts, counts, etc.
```

G.21.2.6 value_type

```
template<class T>
using BiometricEvaluation::Memory::AutoArray< T >:: value_type = T
    Type of element
```

G.21.3 Constructor & Destructor Documentation

G.21.3.1 AutoArray() [1/4]

```
template<class T >
BiometricEvaluation::Memory::AutoArray< T >:: AutoArray (
    size_type size = 0 ) [explicit]
    Construct an AutoArray (p. 233).
```

Parameters

in	size	The number of elements this AutoArray (p. 233) should initially hold.
----	------	--

Exceptions

Error::MemoryError (p. 436)	Could not allocate new memory.
------------------------------------	--------------------------------

G.21.3.2 AutoArray() [2/4]

```
template<class T >
BiometricEvaluation::Memory::AutoArray< T >:: AutoArray (
    const AutoArray< T > & copy )
    Construct an AutoArray (p. 233).
```

Parameters

in	copy	An AutoArray (p. 233) whose contents will be deep copied into the new AutoArray (p. 233).
----	------	---

Exceptions

Error::MemoryError (p. 436)	Could not allocate new memory.
------------------------------------	--------------------------------

G.21.3.3 AutoArray() [3/4]

```
template<class T >
BiometricEvaluation::Memory::AutoArray< T >:: AutoArray (
```

AutoArray< T > && *rvalue*) [noexcept]
Construct an **AutoArray** (p. 233).

Parameters

in	<i>rvalue</i>	An rvalue reference to an AutoArray (p. 233) whose contents will be moved and destroyed.
----	---------------	---

G.21.3.4 AutoArray() [4/4]

```
template<class T>
BiometricEvaluation::Memory::AutoArray< T >:: AutoArray (
    std::initializer_list< T > ilist )
Construct an AutoArray (p. 233).
```

Parameters

in	<i>ilist</i>	An initializer list of type T.
----	--------------	--------------------------------

G.21.3.5 ~AutoArray()

```
template<class T >
BiometricEvaluation::Memory::AutoArray< T >::~ ~AutoArray ( )
Destructor
```

G.21.4 Member Function Documentation

G.21.4.1 at() [1/2]

```
template<class T >
BiometricEvaluation::Memory::AutoArray< T >:: reference BiometricEvaluation::Memory::AutoArray< T >::at (
    ptrdiff_t index )
Subscript into the AutoArray (p. 233) with checked access.
```

Parameters

in	<i>index</i>	Subscript into underlying storage.
----	--------------	------------------------------------

Returns

Reference to the element at the specified index.

Exceptions

<i>out_of_range</i>	Specified index is outside the bounds of this AutoArray (p. 233).
---------------------	--

G.21.4.2 at() [2/2]

```
template<class T >
BiometricEvaluation::Memory::AutoArray< T >:: const_reference BiometricEvaluation::Memory::AutoArray< T >::at (
```

```
    ptrdiff_t index ) const
```

Subscript into the **AutoArray** (p. 233) with checked access.

Parameters

<i>index</i>	Subscript into underlying storage.
--------------	------------------------------------

Returns

Const reference to the element at the specified index.

Exceptions

<i>out_of_range</i>	Specified index is outside the bounds of this AutoArray (p. 233).
---------------------	--

G.21.4.3 begin() [1/2]

```
template<class T >
BiometricEvaluation::Memory::AutoArray< T >:: iterator BiometricEvaluation::Memory::AutoArray< T >::begin ( )
```

Obtain an iterator to the beginning of the **AutoArray** (p. 233).

Returns

Iterator positioned at the first element of the **AutoArray** (p. 233).

G.21.4.4 begin() [2/2]

```
template<class T >
BiometricEvaluation::Memory::AutoArray< T >:: const_iterator BiometricEvaluation::Memory::AutoArray< T >::begin ( ) const
```

Obtain an iterator to the beginning of the **AutoArray** (p. 233).

Returns

Const iterator positioned at the first element of the **AutoArray** (p. 233).

G.21.4.5 cbegin()

```
template<class T >
BiometricEvaluation::Memory::AutoArray< T >:: const_iterator BiometricEvaluation::Memory::
::AutoArray< T >::cbegin ( ) const
```

Obtain an iterator to the beginning of the **AutoArray** (p. 233).

Returns

Const iterator positioned at the first element of the **AutoArray** (p. 233).

G.21.4.6 cend()

```
template<class T >
BiometricEvaluation::Memory::AutoArray< T >:: const_iterator BiometricEvaluation::Memory::
::AutoArray< T >::cend ( ) const
```

Obtain an iterator to the end of the **AutoArray** (p. 233).

Returns

Iterator positioned at the one-past-last element of the **AutoArray** (p. 233).

G.21.4.7 copy() [1/2]

```
template<class T>
void BiometricEvaluation::Memory::AutoArray< T >::copy (
    const T * buffer )
```

Deep-copy the contents of a buffer into this **AutoArray** (p. 233).

Parameters

in	<i>buffer</i>	An allocated buffer whose contents will be deep-copied into this object. Only size() (p. 242) bytes will be copied.
----	---------------	--

Warning

If buffer is smaller in size than the current size of the **AutoArray** (p. 233), you MUST call **copy(const T*, size_type)** (p. 239). This method must only be used when buffer is larger than or equal to the size of the **AutoArray** (p. 233).

G.21.4.8 copy() [2/2]

```
template<class T>
void BiometricEvaluation::Memory::AutoArray< T >::copy (
    const T * buffer,
    size_type size )
```

Deep-copy the contents of a buffer into this **AutoArray** (p. 233).

Parameters

in	<i>buffer</i>	An allocated buffer whose contents will be deep-copied into this object.
in	<i>size</i>	The number of bytes from buffer that will be deep-copied.

Warning

size must be less than or equal to the size of buffer.

G.21.4.9 end() [1/2]

```
template<class T >
BiometricEvaluation::Memory::AutoArray< T >:: iterator BiometricEvaluation::Memory::AutoArray< T >::end ( )
```

Obtain an iterator to the end of the **AutoArray** (p. 233).

Returns

Iterator positioned at the one-past-last element of the **AutoArray** (p. 233).

G.21.4.10 end() [2/2]

```
template<class T >
BiometricEvaluation::Memory::AutoArray< T >:: const_iterator BiometricEvaluation::Memory::AutoArray< T >::end ( ) const
```

Obtain an iterator to the end of the **AutoArray** (p. 233).

Returns

Iterator positioned at the one-past-last element of the **AutoArray** (p. 233).

G.21.4.11 operator const T*()

```
template<class T >
BiometricEvaluation::Memory::AutoArray< T >::operator const T* ( ) const
```

Convert **AutoArray** (p. 233) to const T array.

Returns

Const pointer to the beginning of the underlying array storage.

G.21.4.12 operator T*()

```
template<class T >
BiometricEvaluation::Memory::AutoArray< T >::operator T* ( )
```

Convert **AutoArray** (p. 233) to T array.

Returns

Pointer to the beginning of the underlying array storage.

G.21.4.13 operator=() [1/2]

```
template<class T >
BiometricEvaluation::Memory::AutoArray< T > & BiometricEvaluation::Memory::AutoArray< T >↔
::operator= (
    const AutoArray< T > & other )
```

Copy assignment operator overload performing a deep copy.

Parameters

in	<i>other</i>	AutoArray (p. 233) to be copied.
----	--------------	---

Returns

Reference to a new **AutoArray** (p. 233) object, the lvalue **AutoArray** (p. 233).

Exceptions

Error::MemoryError (p. 436)	Could not allocate new memory.
------------------------------------	--------------------------------

G.21.4.14 operator=() [2/2]

```
template<class T >
BiometricEvaluation::Memory::AutoArray< T > & BiometricEvaluation::Memory::AutoArray< T >↔
::operator= (
    AutoArray< T > && other ) [noexcept]
```

Move assignment operator.

Parameters

in	<i>other</i>	rvalue reference to another AutoArray (p. 233), whose contents will be moved and cleared from itself.
----	--------------	--

Returns

Reference to the lvalue **AutoArray** (p. 233).

G.21.4.15 operator[]() [1/2]

```
template<class T >
BiometricEvaluation::Memory::AutoArray< T >:: reference BiometricEvaluation::Memory::Auto↔
Array< T >::operator[] (
    ptrdiff_t index )
```

Subscripting operator overload with unchecked access.

Parameters

in	<i>index</i>	Subscript into underlying storage.
----	--------------	------------------------------------

Returns

Reference to the element at the specified index.

G.21.4.16 `operator[]()` [2/2]

```
template<class T >
BiometricEvaluation::Memory::AutoArray< T >:: const_reference BiometricEvaluation::Memory←
::AutoArray< T >::operator[] (
    ptrdiff_t index ) const
```

Const subscripting operator overload with unchecked access.

Parameters

in	<i>index</i>	Subscript into underlying storage.
----	--------------	------------------------------------

Returns

Const reference to the element at the specified index.

G.21.4.17 `resize()`

```
template<class T >
void BiometricEvaluation::Memory::AutoArray< T >::resize (
    size_type new_size,
    bool free = false )
```

Change the number of accessible elements.

Parameters

in	<i>new_size</i>	The number of elements the AutoArray (p. 233) should have allocated.
in	<i>free</i>	Whether or not excess memory should be freed if the new size is smaller than the current size.

Exceptions

Error::MemoryError (p. 436)	Problem allocating memory.
------------------------------------	----------------------------

G.21.4.18 size()

```
template<class T >
BiometricEvaluation::Memory::AutoArray< T >:: size_type BiometricEvaluation::Memory::AutoArray< T >::size ( ) const
```

Obtain the number of accessible elements.

Returns

Number of accessible elements.

Note

If **resize()** (p. 242) has been called, the value returned from **size()** (p. 242) may be smaller than the actual allocated size of the underlying storage.

G.21.4.19 to_vector()

```
template<class T >
std::vector< T > BiometricEvaluation::Memory::AutoArray< T >::to_vector ( ) const
```

Obtain a copy of elements in this **AutoArray** (p. 233) as a vector.

Warning

A key difference between vectors and AutoArrays is that all elements of a vector must be initialized. Calling this method on an **AutoArray** (p. 233) where not all elements have been initialized will likely cause undefined behavior.

Returns

A vector containing the contents of this **AutoArray** (p. 233).

G.22 BiometricEvaluation::Memory::AutoArrayIterator< CONST, T > Class Template Reference

RandomAccessIterator for any **AutoArray** (p. 233).

```
#include <be_memory_autoarrayiterator.h>
```

Public Types

- using **iterator_category** = std::random_access_iterator_tag
- using **value_type** = typename std::conditional< CONST, const T, T >::type
- using **difference_type** = std::ptrdiff_t
- using **pointer** = typename std::conditional< CONST, const T *, T * >::type
- using **reference** = typename std::conditional< CONST, const T &, T & >::type
- using **container** = typename std::conditional< CONST, const **AutoArray**< T > *, **AutoArray**< T > * >::type

Convenience definition for a reference to the iterated type with appropriate constness.

Public Member Functions

- **AutoArrayIterator** (**container** autoArray=nullptr, **difference_type** offset=0)
Default constructor.
- **AutoArrayIterator** (const **AutoArrayIterator** &rhs)=default
- **AutoArrayIterator** (**AutoArrayIterator** &&rhs)=default
- **~AutoArrayIterator** ()=default
- **AutoArrayIterator** & **operator=** (**pointer** rhs)
- **AutoArrayIterator** & **operator=** (const **AutoArrayIterator** &rhs)=default
- **AutoArrayIterator** & **operator+=** (const **difference_type** &rhs)
- **AutoArrayIterator** & **operator-=** (const **difference_type** &rhs)
- **reference operator*** () const
- **pointer operator->** () const
- **reference operator[]** (const **difference_type** &rhs) const
- **AutoArrayIterator** & **operator++** ()
- **AutoArrayIterator** & **operator--** ()
- **AutoArrayIterator** **operator++** (int postfix)
- **AutoArrayIterator** **operator--** (int postfix)
- **AutoArrayIterator** **operator+** (const **AutoArrayIterator** &rhs) const
- **difference_type operator-** (const **AutoArrayIterator**< CONST, T > &rhs) const
- **AutoArrayIterator** **operator+** (const **difference_type** &rhs) const
- **AutoArrayIterator** **operator-** (const **difference_type** &rhs) const
- bool **operator==** (const **AutoArrayIterator** &rhs) const
- bool **operator!=** (const **AutoArrayIterator** &rhs) const
- bool **operator>** (const **AutoArrayIterator** &rhs) const
- bool **operator<** (const **AutoArrayIterator** &rhs) const
- bool **operator>=** (const **AutoArrayIterator** &rhs) const
- bool **operator<=** (const **AutoArrayIterator** &rhs) const

Friends

- **AutoArrayIterator** **operator+** (const **difference_type** &lhs, const **AutoArrayIterator** &rhs)
- **AutoArrayIterator** **operator-** (const **difference_type** &lhs, const **AutoArrayIterator** &rhs)

G.22.1 Detailed Description

template<bool CONST, class T>

class BiometricEvaluation::Memory::AutoArrayIterator< CONST, T >

RandomAccessIterator for any **AutoArray** (p. 233).

Note

This class encapsulates a const and non-const iterator in one. The first parameter to the template is a boolean whether or not to use the const version of the iterator. The second is the contained type of the **AutoArray** (p. 233).

G.22.2 Member Typedef Documentation

G.22.2.1 difference_type

```
template<bool CONST, class T>
using BiometricEvaluation::Memory::AutoArrayIterator< CONST, T >:: difference_type = std::
::ptrdiff_t
    Type used to measure distance between iterators
```

G.22.2.2 iterator_category

```
template<bool CONST, class T>
using BiometricEvaluation::Memory::AutoArrayIterator< CONST, T >:: iterator_category = std::
::random_access_iterator_tag
    Type of iterator
```

G.22.2.3 pointer

```
template<bool CONST, class T>
using BiometricEvaluation::Memory::AutoArrayIterator< CONST, T >:: pointer = typename std::
::conditional<CONST, const T*, T*>::type
    Pointer to the type iterated over
```

G.22.2.4 reference

```
template<bool CONST, class T>
using BiometricEvaluation::Memory::AutoArrayIterator< CONST, T >:: reference = typename std::
::conditional<CONST, const T&, T&>::type
    Reference to the type iterated over
```

G.22.2.5 value_type

```
template<bool CONST, class T>
using BiometricEvaluation::Memory::AutoArrayIterator< CONST, T >:: value_type = typename std::
::conditional<CONST, const T, T>::type
    Type when dereferencing iterators
```

G.22.3 Constructor & Destructor Documentation

G.22.3.1 AutoArrayIterator() [1/3]

```
template<bool CONST, class T>
BiometricEvaluation::Memory::AutoArrayIterator< CONST, T >:: AutoArrayIterator (
    container autoArray = nullptr,
    difference_type offset = 0 ) [inline]
    Default constructor.
```

Parameters

autoArray	Pointer to the AutoArray (p. 233) to iterate
offset	The offset into the AutoArray (p. 233) where this iterator should start.

G.22.3.2 AutoArrayIterator() [2/3]

```
template<bool CONST, class T>
BiometricEvaluation::Memory::AutoArrayIterator< CONST, T >:: AutoArrayIterator (
    const AutoArrayIterator< CONST, T > & rhs ) [default]
    Default copy constructor
```

G.22.3.3 AutoArrayIterator() [3/3]

```
template<bool CONST, class T>
BiometricEvaluation::Memory::AutoArrayIterator< CONST, T >:: AutoArrayIterator (
    AutoArrayIterator< CONST, T > && rhs ) [default]
    Default move constructor
```

G.22.3.4 ~AutoArrayIterator()

```
template<bool CONST, class T>
BiometricEvaluation::Memory::AutoArrayIterator< CONST, T >::~~ AutoArrayIterator ( ) [default]
    Default destructor
```

G.22.4 Member Function Documentation**G.22.4.1 operator"!="()**

```
template<bool CONST, class T>
bool BiometricEvaluation::Memory::AutoArrayIterator< CONST, T >::operator!= (
    const AutoArrayIterator< CONST, T > & rhs ) const [inline]
```

Returns

Whether or not the offsets are different.

G.22.4.2 operator*()

```
template<bool CONST, class T>
reference BiometricEvaluation::Memory::AutoArrayIterator< CONST, T >::operator* ( ) const [inline]
```

Returns

Object at the current offset.

G.22.4.3 operator+() [1/2]

```
template<bool CONST, class T>
AutoArrayIterator BiometricEvaluation::Memory::AutoArrayIterator< CONST, T >::operator+ (
    const AutoArrayIterator< CONST, T > & rhs ) const [inline]
```

Returns

This object with offset incremented by rhs' offset.

G.22.4.4 operator+() [2/2]

```
template<bool CONST, class T>
AutoArrayIterator BiometricEvaluation::Memory::AutoArrayIterator< CONST, T >::operator+ (
    const difference_type & rhs ) const [inline]
```

Returns

This object with offset incremented rhs.

G.22.4.5 operator++() [1/2]

```
template<bool CONST, class T>
AutoArrayIterator& BiometricEvaluation::Memory::AutoArrayIterator< CONST, T >::operator++ (
) [inline]
```

Returns

This object with incremented offset.

G.22.4.6 operator++() [2/2]

```
template<bool CONST, class T>
AutoArrayIterator BiometricEvaluation::Memory::AutoArrayIterator< CONST, T >::operator++ (
    int postfix ) [inline]
```

Returns

This object before incrementing offset.

G.22.4.7 operator+=()

```
template<bool CONST, class T>
AutoArrayIterator& BiometricEvaluation::Memory::AutoArrayIterator< CONST, T >::operator+= (
    const difference_type & rhs ) [inline]
```

Returns

This object with rhs added to offset.

G.22.4.8 operator-() [1/2]

```
template<bool CONST, class T>
difference_type BiometricEvaluation::Memory::AutoArrayIterator< CONST, T >::operator- (
    const AutoArrayIterator< CONST, T > & rhs ) const [inline]
```

Returns

Offset decremented by rhs' offset.

G.22.4.9 operator-() [2/2]

```
template<bool CONST, class T>
AutoArrayIterator BiometricEvaluation::Memory::AutoArrayIterator< CONST, T >::operator- (
    const difference_type & rhs ) const [inline]
```

Returns

This object with offset decremented rhs.

G.22.4.10 operator--() [1/2]

```
template<bool CONST, class T>
AutoArrayIterator& BiometricEvaluation::Memory::AutoArrayIterator< CONST, T >::operator-- (
) [inline]
```

Returns

This object with decremented offset.

G.22.4.11 operator--() [2/2]

```
template<bool CONST, class T>
AutoArrayIterator BiometricEvaluation::Memory::AutoArrayIterator< CONST, T >::operator-- (
    int postfix ) [inline]
```

Returns

This object before decrementing offset.

G.22.4.12 operator-=()

```
template<bool CONST, class T>
AutoArrayIterator& BiometricEvaluation::Memory::AutoArrayIterator< CONST, T >::operator-= (
    const difference_type & rhs ) [inline]
```

Returns

This object with rhs removed from offset.

G.22.4.13 operator->()

```
template<bool CONST, class T>
pointer BiometricEvaluation::Memory::AutoArrayIterator< CONST, T >::operator-> ( ) const [inline]
```

Returns

Address of object at the current offset.

G.22.4.14 operator<()

```
template<bool CONST, class T>
bool BiometricEvaluation::Memory::AutoArrayIterator< CONST, T >::operator< (
    const AutoArrayIterator< CONST, T > & rhs ) const [inline]
```

Returns

true if this offset is < rhs'.

G.22.4.15 operator<=()

```
template<bool CONST, class T>
bool BiometricEvaluation::Memory::AutoArrayIterator< CONST, T >::operator<= (
    const AutoArrayIterator< CONST, T > & rhs ) const [inline]
```

Returns

true if this offset is <= rhs'.

G.22.4.16 operator=() [1/2]

```
template<bool CONST, class T>
AutoArrayIterator& BiometricEvaluation::Memory::AutoArrayIterator< CONST, T >::operator= (
    pointer rhs ) [inline]
```

Returns

This object with offset set to rhs.

G.22.4.17 operator=() [2/2]

```
template<bool CONST, class T>
AutoArrayIterator& BiometricEvaluation::Memory::AutoArrayIterator< CONST, T >::operator= (
    const AutoArrayIterator< CONST, T > & rhs ) [inline], [default]
```

Default assignment operator.

G.22.4.18 operator==()

```
template<bool CONST, class T>
bool BiometricEvaluation::Memory::AutoArrayIterator< CONST, T >::operator== (
    const AutoArrayIterator< CONST, T > & rhs ) const [inline]
```

Returns

Whether or not the offsets are the same.

G.22.4.19 operator>()

```
template<bool CONST, class T>
bool BiometricEvaluation::Memory::AutoArrayIterator< CONST, T >::operator> (
    const AutoArrayIterator< CONST, T > & rhs ) const [inline]
```

Returns

true if this offset is > rhs'.

G.22.4.20 operator>=()

```
template<bool CONST, class T>
bool BiometricEvaluation::Memory::AutoArrayIterator< CONST, T >::operator>= (
    const AutoArrayIterator< CONST, T > & rhs ) const [inline]
```

Returns

true if this offset is >= rhs'.

G.22.4.21 operator[]()

```
template<bool CONST, class T>
reference BiometricEvaluation::Memory::AutoArrayIterator< CONST, T >::operator[] (
    const difference_type & rhs ) const [inline]
```

Returns

Object at rhs.

G.22.5 Friends And Related Function Documentation**G.22.5.1 operator+**

```
template<bool CONST, class T>
AutoArrayIterator operator+ (
    const difference_type & lhs,
    const AutoArrayIterator< CONST, T > & rhs ) [friend]
```

Returns

New iterator combining offsets.

G.22.5.2 operator-

```
template<bool CONST, class T>
AutoArrayIterator operator- (
    const difference_type & lhs,
    const AutoArrayIterator< CONST, T > & rhs ) [friend]
```

Returns

New iterator differing offsets, iterating rhs' **AutoArray** (p. 233).

G.23 BiometricEvaluation::Memory::AutoBuffer< T > Class Template Reference

Public Types

- using **value_type** = T
Manage a memory buffer.
- using **reference** = T &
- using **const_reference** = const T &

Public Member Functions

- **operator T*** ()
- T * **operator->** ()
- **AutoBuffer** & **operator=** (const **AutoBuffer** &other)
- **AutoBuffer** (T *data)
- **AutoBuffer** (int(*ctor)(T **), void(*dtor)(T *), int(*copyCtor)(T **, T *)=nullptr)
- **AutoBuffer** (const **AutoBuffer** ©)

G.23.1 Member Typedef Documentation

G.23.1.1 value_type

```
template<class T>
using BiometricEvaluation::Memory::AutoBuffer< T >:: value_type = T
    Manage a memory buffer.
```

It's easier to think of **AutoBuffer** (p. 251) as a wrapper for a pointer rather than the object it truly is. Therefore, you can interact with the **AutoBuffer** (p. 251) object exactly how you would a traditional pointer, without worrying about memory management.

Say you wanted to use an ANSI_NIST* but didn't want to be responsible for allocating or freeing the memory. Create an **AutoBuffer** (p. 251) object like:

```
AutoBuffer<ANSI_NIST> obj = AutoBuffer(allocator_fn,
    deallocator_fn[, copy_constructor]);
```

Notice the **AutoBuffer** (p. 251) is for ANSI_NIST and not ANSI_NIST*, since **AutoBuffer** (p. 251) will handle the pointer for you. You can pass the **AutoBuffer**<ANSI_NIST> (p. 251) object to any function that takes an ANSI_NIST*. For example, it's perfectly valid to pass our 'obj' object above to:

```
write_fmttext(FILE *, ANSI_NIST *)
```

If you want to access a member from 'obj', you can use the dereference operator just like you would on a regular ANSI_NIST*:

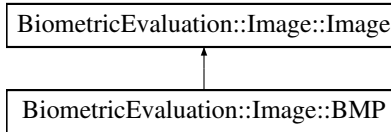
```
int size = obj->num_bytes;
```

G.24 BiometricEvaluation::Image::BMP Class Reference

A BMP-encoded image.

```
#include <be_image_bmp.h>
```

Inheritance diagram for BiometricEvaluation::Image::BMP:



Classes

- struct **ColorTableEntry**

Public Types

- using **ColorTableEntry** = struct **ColorTableEntry**
- using **ColorTable** = std::vector< **ColorTableEntry** >

Public Member Functions

- **BMP** (const uint8_t *data, const uint64_t size)
- **BMP** (const **Memory::uint8Array** &data)
- **Memory::AutoArray**< uint8_t > **getRawData** () const
Accessor for the raw image data. The data returned should not be compressed or encoded.
- **Memory::AutoArray**< uint8_t > **getRawGrayscaleData** (uint8_t depth) const
Accessor for decompressed data in grayscale.

Static Public Member Functions

- static bool **isBMP** (const uint8_t *data, uint64_t size)

Additional Inherited Members

G.24.1 Detailed Description

A BMP-encoded image.

Note

Only supports uncompressed BMPs with the 40-byte BITMAPINFOHEADER header information with no compression or RLE8 compression.

G.24.2 Member Function Documentation

G.24.2.1 getRawData()

Memory::AutoArray<uint8_t> BiometricEvaluation::Image::BMP::getRawData () const [virtual]
Accessor for the raw image data. The data returned should not be compressed or encoded.

Returns

AutoArray holding raw image data.

Exceptions

Error::DataError (p. 295)	Error (p. 108) decompressing image data.
----------------------------------	---

Implements **BiometricEvaluation::Image::Image** (p. 361).

G.24.2.2 getRawGrayscaleData()

Memory::AutoArray<uint8_t> BiometricEvaluation::Image::BMP::getRawGrayscaleData (
uint8_t depth) const [virtual]
Accessor for decompressed data in grayscale.

Parameters

<i>depth</i>	The desired bit depth of the resulting raw image. This value may either be 16, 8, or 1.
--------------	---

Returns

AutoArray holding raw grayscale image data.

Exceptions

Error::DataError (p. 295)	Error (p. 108) decompressing image data.
Error::NotImplemented (p. 456)	Unsupported conversion based on source color depth.
Error::ParameterError (p. 474)	Invalid value for depth.

Note

This method does not save a cached copy of the decompressed image because the bit depth of the image can be changed between calls.

When depth is 1, this method returns an image that uses 8 bits to represent a single pixel. The depth parameter is used to adjust the number of gray levels. When depth is 1, there are only 2 gray levels (black and white), despite using 8 bits to represent each pixel.

Alpha channels are completely ignored when converting to grayscale.

Implements **BiometricEvaluation::Image::Image** (p. 362).

G.24.2.3 isBMP()

static bool BiometricEvaluation::Image::BMP::isBMP (

```
const uint8_t * data,
uint64_t size ) [static]
```

Whether or not data is a **BMP** (p. 252) image.

Parameters

in	<i>data</i>	The buffer to check.
in	<i>size</i>	The size of data.

Returns

true if data appears to be a **BMP** (p. 252) image, false otherwise.

G.25 BiometricEvaluation::DataInterchange::AN2KRecord::CharacterSet Struct Reference

Public Member Functions

- **CharacterSet** (uint16_t **identifier**=0, std::string **commonName**="", std::string **version**="")
Create a new *CharacterSet* (p. 254) struct.

Public Attributes

- uint16_t **identifier**
- std::string **commonName**
- std::string **version**

G.25.1 Constructor & Destructor Documentation

G.25.1.1 CharacterSet()

```
BiometricEvaluation::DataInterchange::AN2KRecord::CharacterSet::CharacterSet (
    uint16_t identifier = 0,
    std::string commonName = "",
    std::string version = "" ) [inline]
```

Create a new **CharacterSet** (p. 254) struct.

Parameters

<i>identifier</i>	Numeric identifier of the character set.
<i>commonName</i>	Common name of the character set.
<i>version</i>	Optional version number of the character set.

G.25.2 Member Data Documentation

G.25.2.1 commonName

`std::string BiometricEvaluation::DataInterchange::AN2KRecord::CharacterSet::commonName`
Common name of the character set

G.25.2.2 identifier

`uint16_t BiometricEvaluation::DataInterchange::AN2KRecord::CharacterSet::identifier`
Identifier (000-999)

G.25.2.3 version

`std::string BiometricEvaluation::DataInterchange::AN2KRecord::CharacterSet::version`
Optional version of the character set

G.26 BiometricEvaluation::Image::BMP::ColorTableEntry Struct Reference

```
#include <be_image_bmp.h>
```

Public Attributes

- `uint8_t red`
- `uint8_t green`
- `uint8_t blue`
- `uint8_t reserved`

G.26.1 Detailed Description

One element of the colormap table.

G.26.2 Member Data Documentation

G.26.2.1 blue

`uint8_t BiometricEvaluation::Image::BMP::ColorTableEntry::blue`
Blue value

G.26.2.2 green

`uint8_t BiometricEvaluation::Image::BMP::ColorTableEntry::green`
Green value

G.26.2.3 red

`uint8_t BiometricEvaluation::Image::BMP::ColorTableEntry::red`
Red value

G.26.2.4 reserved

```
uint8_t BiometricEvaluation::Image::BMP::ColorTableEntry::reserved
    Reserved value
```

G.27 BiometricEvaluation::Process::CommandCenter< T, typename >↵ ::Command Class Reference

```
#include <be_process_commandcenter.h>
```

Public Attributes

- uint32_t **clientID**
- T **command**
- std::vector< std::string > **arguments**

G.27.1 Detailed Description

```
template<typename T, typename = typename std::enable_if<std::is_enum<T>::value>>
class BiometricEvaluation::Process::CommandCenter< T, typename >::Command
```

Parsed command received from the network.

G.27.2 Member Data Documentation

G.27.2.1 arguments

```
template<typename T, typename = typename std::enable_if<std::is_enum<T>::value>>
std::vector<std::string> BiometricEvaluation::Process::CommandCenter< T, typename >::Command↵
::arguments
    Arguments passed to command (optional).
```

G.27.2.2 clientID

```
template<typename T, typename = typename std::enable_if<std::is_enum<T>::value>>
uint32_t BiometricEvaluation::Process::CommandCenter< T, typename >::Command::clientID
    ID of the sender.
```

G.27.2.3 command

```
template<typename T, typename = typename std::enable_if<std::is_enum<T>::value>>
T BiometricEvaluation::Process::CommandCenter< T, typename >::Command::command
    Enumeration value of the command.
```

G.28 BiometricEvaluation::Process::CommandCenter< T, typename > Class Template Reference

```
#include <be_process_commandcenter.h>
```

Classes

- class **Command**

Public Member Functions

- **CommandCenter** (uint16_t port= **MessageCenter::DEFAULT_PORT**)
Constructor.
- **~CommandCenter** ()=default
- bool **hasPendingCommands** ()
Determine if there are commands waiting.
- bool **getNextCommand** (**Command** &command, int numSeconds=-1, std::string invalidCommand←Response="")
Get the next command.
- void **sendResponse** (uint32_t clientID, const std::string &response, const std::string prefix=">> ", const std::string suffix="\")
Send a string response to a client.
- void **disconnectClient** (uint32_t clientID)
Break the connection with a client.

G.28.1 Detailed Description

```
template<typename T, typename = typename std::enable_if<std::is_enum<T>::value>>
class BiometricEvaluation::Process::CommandCenter< T, typename >
```

Receive enumerations as commands over the network.

G.28.2 Constructor & Destructor Documentation

G.28.2.1 CommandCenter()

```
template<typename T, typename = typename std::enable_if<std::is_enum<T>::value>>
BiometricEvaluation::Process::CommandCenter< T, typename >:: CommandCenter (
    uint16_t port = MessageCenter::DEFAULT_PORT ) [inline]
    Constructor.
```

Parameters

<i>port</i>	Port to listen on for commands.
-------------	---------------------------------

G.28.2.2 ~CommandCenter()

```
template<typename T, typename = typename std::enable_if<std::is_enum<T>::value>>
BiometricEvaluation::Process::CommandCenter< T, typename >::~ ~CommandCenter ( ) [default]
    Destructor (default).
```

G.28.3 Member Function Documentation

G.28.3.1 disconnectClient()

```
template<typename T, typename = typename std::enable_if<std::is_enum<T>::value>>
void BiometricEvaluation::Process::CommandCenter< T, typename >::disconnectClient (
    uint32_t clientID ) [inline]
```

Break the connection with a client.

Parameters

<i>clientID</i>	ID of the client to disconnect.
-----------------	---------------------------------

G.28.3.2 getNextCommand()

```
template<typename T, typename = typename std::enable_if<std::is_enum<T>::value>>
bool BiometricEvaluation::Process::CommandCenter< T, typename >::getNextCommand (
    Command & command,
    int numSeconds = -1,
    std::string invalidCommandResponse = "" ) [inline]
```

Get the next command.

Parameters

<i>command</i>	Reference to a Command (p. 256) that will be populated when this method returns true.
<i>numSeconds</i>	Number of seconds to wait for a command, or -1 to block indefinitely.
<i>invalidCommandResponse</i>	Optional string to send, such as usage, that will be sent when an unrecognized command is received.

Returns

true if command has been populated, false otherwise.

G.28.3.3 hasPendingCommands()

```
template<typename T, typename = typename std::enable_if<std::is_enum<T>::value>>
bool BiometricEvaluation::Process::CommandCenter< T, typename >::hasPendingCommands ( ) [inline]
```

Determine if there are commands waiting.

Returns

true if there are commands waiting, false otherwise.

Note

Returns immediately.

See also

BiometricEvaluation::Process::CommandCenter (p. 256):: **getNextCommand()** (p. 258)

G.28.3.4 sendResponse()

```
template<typename T, typename = typename std::enable_if<std::is_enum<T>::value>>
void BiometricEvaluation::Process::CommandCenter< T, typename >::sendResponse (
    uint32_t clientID,
    const std::string & response,
    const std::string prefix = ">> ",
    const std::string suffix = "\n" ) [inline]
```

Send a string response to a client.

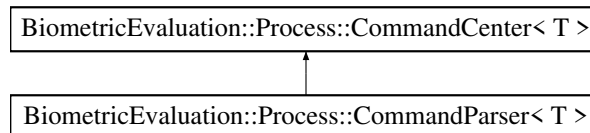
Parameters

<i>clientID</i>	ID of client to communicate with.
<i>response</i>	Printable string to send to client.
<i>prefix</i>	String to prefix to responses.
<i>suffix</i>	String to append to responses.

G.29 BiometricEvaluation::Process::CommandParser< T > Class Template Reference

```
#include <be_process_commandcenter.h>
```

Inheritance diagram for BiometricEvaluation::Process::CommandParser< T >:



Public Member Functions

- virtual void **parse** (const typename **CommandCenter**< T >::Command &command)=0
Parse command.
- bool **getNextCommand** (typename **CommandCenter**< T >::Command &command, int numSeconds=-1)
Get the next command.
- void **setUsage** (const std::string &usage)
String sent when an invalid command is received.
- std::string **getUsage** () const
- **CommandParser** (uint16_t port= **MessageCenter::DEFAULT_PORT**)
Constructor.
- virtual ~**CommandParser** ()=default

G.29.1 Detailed Description

template<typename T>
class BiometricEvaluation::Process::CommandParser< T >

Abstraction to parse messages received via **CommandCenter** (p. 256).

G.29.2 Constructor & Destructor Documentation

G.29.2.1 CommandParser()

```
template<typename T >
BiometricEvaluation::Process::CommandParser< T >:: CommandParser (
    uint16_t port = MessageCenter::DEFAULT_PORT ) [inline]
```

Constructor.

Parameters

<i>port</i>	Port to listen on for commands.
-------------	---------------------------------

G.29.2.2 ~CommandParser()

```
template<typename T >
virtual BiometricEvaluation::Process::CommandParser< T >::~~ CommandParser ( ) [virtual], [default]
```

Virtual destructor (default).

G.29.3 Member Function Documentation

G.29.3.1 getNextCommand()

```
template<typename T >
bool BiometricEvaluation::Process::CommandParser< T >::getNextCommand (
    typename CommandCenter< T >::Command & command,
    int numSeconds = -1 ) [inline]
```

Get the next command.

Parameters

<i>command</i>	Reference to a Command that will be populated when this method returns true.
<i>numSeconds</i>	Number of seconds to wait for a command, or -1 to block indefinitely.

Returns

true if command has been populated, false otherwise.

G.29.3.2 `getUsage()`

```
template<typename T >
std::string BiometricEvaluation::Process::CommandParser< T >::getUsage ( ) const [inline]
```

Returns

Usage string.

G.29.3.3 `parse()`

```
template<typename T >
virtual void BiometricEvaluation::Process::CommandParser< T >::parse (
    const typename CommandCenter< T >::Command & command ) [pure virtual]
```

Parse command.

Implement this method as a switch statement of your command enumeration.

G.29.3.4 `setUsage()`

```
template<typename T >
void BiometricEvaluation::Process::CommandParser< T >::setUsage (
    const std::string & usage ) [inline]
```

String sent when an invalid command is received.

Parameters

<i>usage</i>	String to send when an invalid command is received.
--------------	---

Note

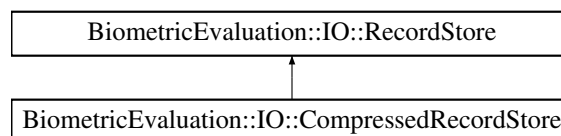
If not set, no additional usage is sent.

G.30 BiometricEvaluation::IO::CompressedRecordStore Class Reference

Sibling-implemented **RecordStore** (p. 504) with Compression.

```
#include <be_io_compressedrecstore.h>
```

Inheritance diagram for BiometricEvaluation::IO::CompressedRecordStore:



Public Member Functions

- **CompressedRecordStore** (const std::string &pathname, const std::string &description, const **RecordStore::Kind** &recordStoreType, const std::string &compressorType)

- **CompressedRecordStore** (const std::string &pathname, const std::string &description, const **RecordStore::Kind** &recordStoreType, const **Compressor::Kind** &compressorType)
- **CompressedRecordStore** (const std::string &pathname, **IO::Mode** mode= **IO::Mode::ReadOnly**)
- uint64_t **getSpaceUsed** () const override
Obtain real storage utilization.
- void **sync** () const override
- unsigned int **getCount** () const override
- std::string **getPathname** () const override
- std::string **getDescription** () const override
- void **changeDescription** (const std::string &description) override
- void **insert** (const std::string &key, const void *const data, const uint64_t size) override
- void **remove** (const std::string &key) override
- **Memory::uint8Array** **read** (const std::string &key) const override
Read a complete record from a store.
- uint64_t **length** (const std::string &key) const override
- void **flush** (const std::string &key) const override
- **RecordStore::Record** **sequence** (int cursor= **BE_RECSTORE_SEQ_NEXT**) override
*Sequence through a **RecordStore** (p. 504), returning the key/data pairs.*
- std::string **sequenceKey** (int cursor= **BE_RECSTORE_SEQ_NEXT**) override
*Sequence through a **RecordStore** (p. 504), returning the key.*
- void **setCursorAtKey** (const std::string &key) override
- void **move** (const std::string &pathname) override
*Move the **RecordStore** (p. 504).*
- **CompressedRecordStore** (const **CompressedRecordStore** &rhs)=delete
Copy constructor (disabled).
- **CompressedRecordStore** & **operator=** (const **CompressedRecordStore** &rhs)=delete
Assignment operator (disabled).

Additional Inherited Members

G.30.1 Detailed Description

Sibling-implemented **RecordStore** (p. 504) with Compression.

G.30.2 Constructor & Destructor Documentation

G.30.2.1 CompressedRecordStore() [1/4]

```
BiometricEvaluation::IO::CompressedRecordStore::CompressedRecordStore (
    const std::string & pathname,
    const std::string & description,
    const RecordStore::Kind & recordStoreType,
    const std::string & compressorType )
```

Create a new **CompressedRecordStore** (p. 261), read/write mode.

Parameters

in	<i>pathname</i>	The directory where the store is to be created.
----	-----------------	---

Parameters

in	<i>description</i>	The store's description.
in	<i>recordStoreType</i>	The type of RecordStore (p. 504) subclass the internal RecordStores should be.
in	<i>compressorType</i>	The type of compression that should be used within the internal RecordStores.

Exceptions

Error::ObjectExists (p. 458)	The store already exists.
Error::StrategyError (p. 567)	An error occurred when accessing the underlying file system.

G.30.2.2 CompressedRecordStore() [2/4]

```
BiometricEvaluation::IO::CompressedRecordStore::CompressedRecordStore (
    const std::string & pathname,
    const std::string & description,
    const RecordStore::Kind & recordStoreType,
    const Compressor::Kind & compressorType )
```

Create a new **CompressedRecordStore** (p. 261), read/write mode.

Parameters

in	<i>pathname</i>	The directory where the store is to be created.
in	<i>description</i>	The store's description.
in	<i>recordStoreType</i>	The type of RecordStore (p. 504) subclass the internal RecordStores should be.
in	<i>compressorType</i>	The type of compression that should be used within the internal RecordStores.

Exceptions

Error::ObjectExists (p. 458)	The store already exists.
Error::StrategyError (p. 567)	An error occurred when accessing the underlying file system.

G.30.2.3 CompressedRecordStore() [3/4]

```
BiometricEvaluation::IO::CompressedRecordStore::CompressedRecordStore (
    const std::string & pathname,
    IO::Mode mode = IO::Mode::ReadOnly )
```

Open an existing **CompressedRecordStore** (p. 261).

Parameters

in	<i>pathname</i>	The path name of the store.
in	<i>mode</i>	Open mode, read-only or read-write.

Exceptions

Error::ObjectDoesNotExist (p. 457)	The store does not exist.
Error::StrategyError (p. 567)	An error occurred when accessing the underlying file system.

G.30.2.4 CompressedRecordStore() [4/4]

```
BiometricEvaluation::IO::CompressedRecordStore::CompressedRecordStore (
    const CompressedRecordStore & rhs ) [delete]
```

Copy constructor (disabled).

Disabled because this object could represent a file on disk.

Parameters

<i>rhs</i>	CompressedRecordStore (p. 261) object to copy.
------------	--

G.30.3 Member Function Documentation**G.30.3.1 changeDescription()**

```
void BiometricEvaluation::IO::CompressedRecordStore::changeDescription (
    const std::string & description ) [override], [virtual]
```

Change the description of the **RecordStore** (p. [504](#)).

Parameters

<i>in</i>	<i>description</i>	The new description.
-----------	--------------------	----------------------

Exceptions

Error::StrategyError (p. 567)	An error occurred when using the underlying storage system.
---	---

Implements **BiometricEvaluation::IO::RecordStore** (p. [506](#)).

G.30.3.2 flush()

```
void BiometricEvaluation::IO::CompressedRecordStore::flush (
    const std::string & key ) const [override], [virtual]
```

Commit the record's data to storage.

Parameters

<i>in</i>	<i>key</i>	The key of the record to be flushed.
-----------	------------	--------------------------------------

Exceptions

Error::ObjectDoesNotExist (p. 457)	A record for the key does not exist.
Error::StrategyError (p. 567)	An error occurred when using the underlying storage system.

Implements **BiometricEvaluation::IO::RecordStore** (p. 508).

G.30.3.3 getCount()

```
unsigned int BiometricEvaluation::IO::CompressedRecordStore::getCount ( ) const [override], [virtual]
```

Obtain the number of items in the **RecordStore** (p. 504).

Returns

The number of items in the **RecordStore** (p. 504).

Implements **BiometricEvaluation::IO::RecordStore** (p. 508).

G.30.3.4 getDescription()

```
std::string BiometricEvaluation::IO::CompressedRecordStore::getDescription ( ) const [override], [virtual]
```

Obtain a textual description of the **RecordStore** (p. 504).

Returns

The **RecordStore** (p. 504)'s description.

Implements **BiometricEvaluation::IO::RecordStore** (p. 508).

G.30.3.5 getPathname()

```
std::string BiometricEvaluation::IO::CompressedRecordStore::getPathname ( ) const [override], [virtual]
```

Return the path name of the **RecordStore** (p. 504).

Returns

Where in the file system the **RecordStore** (p. 504) is located.

Implements **BiometricEvaluation::IO::RecordStore** (p. 509).

G.30.3.6 getSpaceUsed()

```
uint64_t BiometricEvaluation::IO::CompressedRecordStore::getSpaceUsed ( ) const [override], [virtual]
```

Obtain real storage utilization.

The amount of disk space used, for example. This is the actual space allocated by the underlying storage mechanism, in bytes.

Returns

The amount of backing storage used by the **RecordStore** (p. 504).

Exceptions

Error::StrategyError (p. 567)	An error occurred when using the underlying storage system.
---	---

Implements **BiometricEvaluation::IO::RecordStore** (p. [509](#)).

G.30.3.7 insert()

```
void BiometricEvaluation::IO::CompressedRecordStore::insert (
    const std::string & key,
    const void *const data,
    const uint64_t size ) [override], [virtual]
```

Insert a record into the store.

Parameters

in	<i>key</i>	The key of the record to be inserted.
in	<i>data</i>	The data for the record.
in	<i>size</i>	The size of the record, in bytes.

Exceptions

Error::ObjectExists (p. 458)	A record with the given key is already present.
Error::StrategyError (p. 567)	The RecordStore (p. 504) is opened read-only, or an error occurred when using the underlying storage system.

Implements **BiometricEvaluation::IO::RecordStore** (p. [510](#)).

G.30.3.8 length()

```
uint64_t BiometricEvaluation::IO::CompressedRecordStore::length (
    const std::string & key ) const [override], [virtual]
```

Return the length of a record.

Parameters

in	<i>key</i>	The key of the record.
----	------------	------------------------

Returns

The record length.

Exceptions

Error::ObjectDoesNotExist (p. 457)	A record for the key does not exist.
Error::StrategyError (p. 567)	An error occurred when using the underlying storage system.

Implements **BiometricEvaluation::IO::RecordStore** (p. 510).

G.30.3.9 move()

```
void BiometricEvaluation::IO::CompressedRecordStore::move (
    const std::string & pathname ) [override], [virtual]
```

Move the **RecordStore** (p. 504).

The **RecordStore** (p. 504) can be moved to a new path in the file system.

Parameters

in	<i>pathname</i>	The new path of the RecordStore (p. 504).
----	-----------------	--

Exceptions

Error::StrategyError (p. 567)	An error occurred when using the underlying storage system.
--------------------------------------	---

Implements **BiometricEvaluation::IO::RecordStore** (p. 511).

G.30.3.10 operator=()

```
CompressedRecordStore& BiometricEvaluation::IO::CompressedRecordStore::operator= (
    const CompressedRecordStore & rhs ) [delete]
```

Assignment operator (disabled).

Disabled because this object could represent a file on disk.

Parameters

<i>rhs</i>	CompressedRecordStore (p. 261) object to assign.
------------	---

Returns

CompressedRecordStore (p. 261) object, now containing the contents of rhs.

G.30.3.11 read()

```
Memory::uint8Array BiometricEvaluation::IO::CompressedRecordStore::read (
    const std::string & key ) const [override], [virtual]
```

Read a complete record from a store.

The AutoArray will be resized to match the size of the data.

Parameters

in	<i>key</i>	The key of the record to be read.
----	------------	-----------------------------------

Returns

The record associated with the key.

Exceptions

Error::ObjectDoesNotExist (p. 457)	A record for the key does not exist.
Error::StrategyError (p. 567)	An error occurred when using the underlying storage system.

Implements **BiometricEvaluation::IO::RecordStore** (p. 512).

G.30.3.12 remove()

```
void BiometricEvaluation::IO::CompressedRecordStore::remove (
    const std::string & key ) [override], [virtual]
```

Remove a record from the store.

Parameters

in	key	The key of the record to be removed.
----	-----	--------------------------------------

Exceptions

Error::ObjectDoesNotExist (p. 457)	A record for the key does not exist.
Error::StrategyError (p. 567)	An error occurred when using the underlying storage system.

Implements **BiometricEvaluation::IO::RecordStore** (p. 513).

G.30.3.13 sequence()

```
RecordStore::Record BiometricEvaluation::IO::CompressedRecordStore::sequence (
    int cursor = BE_RECSTORE_SEQ_NEXT ) [override], [virtual]
```

Sequence through a **RecordStore** (p. 504), returning the key/data pairs.

Sequencing means to start at some point in the store and return the record, then repeatedly calling the function to return the next record. The starting point is typically the first record, and is set to that when the **RecordStore** (p. 504) object is created. The starting point can be reset by calling this method with the cursor parameter set to **BE_RECSTORE_SEQ_START**.

Parameters

in	cursor	The location within the sequence of the key/data pair to return.
----	--------	--

Returns

The record that is currently in sequence.

Exceptions

Error::ObjectDoesNotExist (p. 457)	End of sequencing.
Error::StrategyError (p. 567)	An error occurred when using the underlying storage system.

Implements **BiometricEvaluation::IO::RecordStore** (p. 515).

G.30.3.14 sequenceKey()

```
std::string BiometricEvaluation::IO::CompressedRecordStore::sequenceKey (
    int cursor = BE_RECSTORE_SEQ_NEXT ) [override], [virtual]
```

Sequence through a **RecordStore** (p. 504), returning the key.

Sequencing means to start at some point in the store and return the key, then repeatedly calling the function to return the next key. The starting point is typically the first record, and is set to that when the **RecordStore** (p. 504) object is created. The starting point can be reset by calling this method with the cursor parameter set to **BE_RECSTORE_SEQ_START**.

Parameters

in	<i>cursor</i>	The location within the sequence of the key/data pair to return.
----	---------------	--

Returns

The key of the currently sequenced record.

Exceptions

Error::ObjectDoesNotExist (p. 457)	End of sequencing.
Error::StrategyError (p. 567)	An error occurred when using the underlying storage system.

Implements **BiometricEvaluation::IO::RecordStore** (p. 515).

G.30.3.15 setCursorAtKey()

```
void BiometricEvaluation::IO::CompressedRecordStore::setCursorAtKey (
    const std::string & key ) [override], [virtual]
```

Set the sequence cursor to an arbitrary position within the **RecordStore** (p. 504), starting at key. Key will be the first record returned from the next call to **sequence()** (p. 268).

Parameters

in	<i>key</i>	The key of the record which will be returned by the first subsequent call to sequence() (p. 268).
----	------------	--

Exceptions

Error::ObjectDoesNotExist (p. 457)	A record for the key does not exist.
---	--------------------------------------

Exceptions

Error::StrategyError (p. 567)	An error occurred when using the underlying storage system.
--------------------------------------	---

Implements **BiometricEvaluation::IO::RecordStore** (p. 516).

G.30.3.16 sync()

```
void BiometricEvaluation::IO::CompressedRecordStore::sync ( ) const [override], [virtual]
```

Synchronize the entire record store to persistent storage.

Exceptions

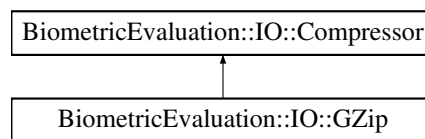
Error::StrategyError (p. 567)	An error occurred when using the underlying storage system.
--------------------------------------	---

Implements **BiometricEvaluation::IO::RecordStore** (p. 516).

G.31 BiometricEvaluation::IO::Compressor Class Reference

```
#include <be.io_compressor.h>
```

Inheritance diagram for BiometricEvaluation::IO::Compressor:



Public Types

- enum **Kind** { **GZIP** }

Public Member Functions

- **Compressor** ()
*Create a new **Compressor** (p. 270) object.*
- virtual **Memory::uint8Array** **compress** (const uint8_t *const uncompressedData, uint64_t uncompressedDataSize) const =0
Compress a buffer.
- virtual **Memory::uint8Array** **compress** (const **Memory::uint8Array** &uncompressedData) const =0
Compress a buffer.
- virtual void **compress** (const uint8_t *const uncompressedData, uint64_t uncompressedDataSize, const std::string &outputFile) const =0
Compress a buffer.
- virtual void **compress** (const **Memory::uint8Array** &uncompressedData, const std::string &outputFile) const =0
Compress a buffer.

- virtual **Memory::uint8Array** **compress** (const std::string &inputFile) const =0
Compress a file.
- virtual void **compress** (const std::string &inputFile, const std::string &outputFile) const =0
Compress a file.
- virtual **Memory::uint8Array** **decompress** (const uint8_t *const compressedData, uint64_t compressedDataSize) const =0
Decompress a compressed buffer.
- virtual **Memory::uint8Array** **decompress** (const **Memory::uint8Array** &compressedData) const =0
Decompress a compressed buffer.
- virtual **Memory::uint8Array** **decompress** (const std::string &inputFile) const =0
Decompress a compressed buffer into a file.
- virtual void **decompress** (const **Memory::uint8Array** &compressedData, const std::string &outputFile) const =0
Decompress a file.
- virtual void **decompress** (const uint8_t *const compressedData, const uint64_t compressedDataSize, const std::string &outputFile) const =0
Decompress a file.
- virtual void **decompress** (const std::string &inputFile, const std::string &outputFile) const =0
Decompress a file.
- void **setOption** (const std::string &optionName, const std::string &optionValue)
Assign a compressor option.
- void **setOption** (const std::string &optionName, int64_t optionValue)
Assign a compressor option.
- std::string **getOption** (const std::string &optionName) const
Obtain a compressor option as an integer.
- int64_t **getOptionAsInteger** (const std::string &optionName) const
Obtain a compressor option as an integer.
- void **removeOption** (const std::string &optionName)
Remove a compressor option.
- virtual **~Compressor** ()
- **Compressor** (const **Compressor** &other)=delete
Copy constructor (disabled).
- **Compressor** & **operator=** (const **Compressor** &other)=delete
Assignment overload (disabled).

Static Public Member Functions

- static std::shared_ptr< **Compressor** > **createCompressor** (**Compressor::Kind** compressorKind=**Kind::GZIP**)

G.31.1 Detailed Description

Implementations for compressing and decompressing data

G.31.2 Member Enumeration Documentation

G.31.2.1 Kind

enum **BiometricEvaluation::IO::Compressor::Kind** [strong]
 Kinds of Compressors (for factory)

G.31.3 Constructor & Destructor Documentation

G.31.3.1 Compressor() [1/2]

BiometricEvaluation::IO::Compressor::Compressor ()
 Create a new **Compressor** (p. 270) object.
 Default compression options will be used.

G.31.3.2 ~Compressor()

virtual BiometricEvaluation::IO::Compressor::~~Compressor () [virtual]
 Destructor

G.31.3.3 Compressor() [2/2]

BiometricEvaluation::IO::Compressor::Compressor (
 const Compressor & other) [delete]
 Copy constructor (disabled).
 Disabled because **Properties** (p. 486) member cannot be copied.

Parameters

<i>other</i>	Compressor (p. 270) to copy.
--------------	-------------------------------------

G.31.4 Member Function Documentation

G.31.4.1 compress() [1/6]

virtual Memory::uint8Array **BiometricEvaluation::IO::Compressor::compress (**
 const uint8_t *const uncompressedData,
 uint64_t uncompressedDataSize) const [pure virtual]
 Compress a buffer.

Parameters

<i>uncompressedData</i>	Uncompressed data buffer to compress.
<i>uncompressedDataSize</i>	Size of uncompressedData.

Returns

Compressed buffer.

Exceptions

<i>Error::StrategyError</i> (p. 567)	Error (p. 108) in compression unit.
--------------------------------------	--

Implemented in **BiometricEvaluation::IO::GZip** (p. 349).

G.31.4.2 compress() [2/6]

```
virtual Memory::uint8Array BiometricEvaluation::IO::Compressor::compress (
    const Memory::uint8Array & uncompressedData ) const [pure virtual]
```

Compress a buffer.

Parameters

<i>uncompressedData</i>	Uncompressed data buffer to compress.
-------------------------	---------------------------------------

Returns

Compressed buffer.

Exceptions

<i>Error::StrategyError</i> (p. 567)	Error (p. 108) in decompression unit.
--------------------------------------	--

Implemented in **BiometricEvaluation::IO::GZip** (p. 349).

G.31.4.3 compress() [3/6]

```
virtual void BiometricEvaluation::IO::Compressor::compress (
    const uint8_t *const uncompressedData,
    uint64_t uncompressedDataSize,
    const std::string & outputFile ) const [pure virtual]
```

Compress a buffer.

Parameters

<i>uncompressedData</i>	Uncompressed data buffer to compress.
<i>uncompressedDataSize</i>	Size of uncompressedData.
<i>outputFile</i>	Location to save compressed file.

Exceptions

<i>Error::ObjectExists</i> (p. 458)	Output file already exists.
<i>Error::StrategyError</i> (p. 567)	Error (p. 108) in compression unit.

Implemented in **BiometricEvaluation::IO::GZip** (p. 349).

G.31.4.4 compress() [4/6]

```
virtual void BiometricEvaluation::IO::Compressor::compress (
    const Memory::uint8Array & uncompressedData,
    const std::string & outputFile ) const [pure virtual]
```

Compress a buffer.

Parameters

<i>uncompressedData</i>	Uncompressed data buffer to compress.
<i>outputFile</i>	Location to save compressed file.

Exceptions

Error::ObjectExists (p. 458)	Output file already exists.
Error::StrategyError (p. 567)	Error (p. 108) in decompression unit.

Implemented in **BiometricEvaluation::IO::GZip** (p. 350).

G.31.4.5 compress() [5/6]

```
virtual Memory::uint8Array BiometricEvaluation::IO::Compressor::compress (
    const std::string & inputFile ) const [pure virtual]
```

Compress a file.

Parameters

<i>inputFile</i>	Path to file to compress.
------------------	---------------------------

Returns

Compressed buffer.

Exceptions

Error::ObjectDoesNotExist (p. 457)	Input file does not exist.
Error::StrategyError (p. 567)	Error (p. 108) in decompression unit.

Implemented in **BiometricEvaluation::IO::GZip** (p. 350).

G.31.4.6 compress() [6/6]

```
virtual void BiometricEvaluation::IO::Compressor::compress (
```

```
const std::string & inputFile,  
const std::string & outputFile ) const [pure virtual]
```

Compress a file.

Parameters

<i>inputFile</i>	Path to file to compress.
<i>outputFile</i>	Path to location where compressed version will be saved.

Exceptions

Error::ObjectDoesNotExist (p. 457)	Input file does not exist.
Error::ObjectExists (p. 458)	Output file already exists.
Error::StrategyError (p. 567)	Error (p. 108) in decompression unit.

Implemented in **BiometricEvaluation::IO::GZip** (p. 351).

G.31.4.7 createCompressor()

```
static std::shared_ptr< Compressor> BiometricEvaluation::IO::Compressor::createCompressor (  
    Compressor::Kind compressorKind = Kind::GZIP ) [static]  
    Compressor (p. 270) factory.
```

Parameters

<i>compressorKind</i>	A known kind of compressor.
-----------------------	-----------------------------

Returns

A new compressor with default options.

Exceptions

Error::ObjectDoesNotExist (p. 457)	Invalid compressor type.
---	--------------------------

G.31.4.8 decompress() [1/6]

```
virtual Memory::uint8Array BiometricEvaluation::IO::Compressor::decompress (  
    const uint8_t *const compressedData,  
    uint64_t compressedDataSize ) const [pure virtual]  
    Decompress a compressed buffer.
```

Parameters

<i>compressedData</i>	Compressed data buffer to decompress.
-----------------------	---------------------------------------

Parameters

<i>compressedDataSize</i>	Size of compressedData.
---------------------------	-------------------------

Returns

Decompressed data.

Exceptions

Error::StrategyError (p. 567)	Error (p. 108) in compression unit.
---	---

Implemented in **BiometricEvaluation::IO::GZip** (p. [351](#)).

G.31.4.9 decompress() [2/6]

```
virtual Memory::uint8Array BiometricEvaluation::IO::Compressor::decompress (
    const Memory::uint8Array & compressedData ) const [pure virtual]
```

Decompress a compressed buffer.

Parameters

<i>compressedData</i>	Compressed data buffer to decompress.
-----------------------	---------------------------------------

Returns

Decompressed data.

Exceptions

Error::StrategyError (p. 567)	Error (p. 108) in decompression unit.
---	---

Implemented in **BiometricEvaluation::IO::GZip** (p. [352](#)).

G.31.4.10 decompress() [3/6]

```
virtual Memory::uint8Array BiometricEvaluation::IO::Compressor::decompress (
    const std::string & inputFile ) const [pure virtual]
```

Decompress a compressed buffer into a file.

Parameters

<i>inputFile</i>	Location to save compressed file.
------------------	-----------------------------------

Returns

Decompressed data.

Exceptions

Error::StrategyError (p. 567)	Error (p. 108) in decompression unit.
Error::ObjectDoesNotExist s	Output file already exists.

Implemented in **BiometricEvaluation::IO::GZip** (p. 352).

G.31.4.11 decompress() [4/6]

```
virtual void BiometricEvaluation::IO::Compressor::decompress (  
    const Memory::uint8Array & compressedData,  
    const std::string & outputFile ) const [pure virtual]
```

Decompress a file.

Parameters

<i>compressedData</i>	Compressed data buffer to decompress.
<i>outputFile</i>	Path to location where decompressed version will be saved.

Exceptions

Error::ObjectExists (p. 458)	Output file already exists.
Error::StrategyError (p. 567)	Error (p. 108) in compression unit.

Implemented in **BiometricEvaluation::IO::GZip** (p. 354).

G.31.4.12 decompress() [5/6]

```
virtual void BiometricEvaluation::IO::Compressor::decompress (  
    const uint8_t *const compressedData,  
    const uint64_t compressedDataSize,  
    const std::string & outputFile ) const [pure virtual]
```

Decompress a file.

Parameters

<i>compressedData</i>	Compressed data buffer to decompress.
<i>compressedDataSize</i>	Size of <i>compressedData</i> .
<i>outputFile</i>	Path to location where decompressed version will be saved.

Exceptions

Error::ObjectExists (p. 458)	Output file already exists.
Error::StrategyError (p. 567)	Error (p. 108) in compression unit.

Implemented in **BiometricEvaluation::IO::GZip** (p. 353).

G.31.4.13 decompress() [6/6]

```
virtual void BiometricEvaluation::IO::Compressor::decompress (
    const std::string & inputFile,
    const std::string & outputFile ) const [pure virtual]
```

Decompress a file.

Parameters

<i>inputFile</i>	Path to file to decompress.
<i>outputFile</i>	Path to location where decompressed version will be saved.

Exceptions

Error::ObjectDoesNotExist (p. 457)	Input file does not exist.
Error::ObjectExists (p. 458)	Output file already exists.
Error::StrategyError (p. 567)	Error (p. 108) in compression unit.

Implemented in **BiometricEvaluation::IO::GZip** (p. 353).

G.31.4.14 getOption()

```
std::string BiometricEvaluation::IO::Compressor::getOption (
    const std::string & optionName ) const
```

Obtain a compressor option as an integer.

Parameters

<i>optionName</i>	Name of the option to obtain.
-------------------	-------------------------------

Returns

Value of compressor option.

G.31.4.15 getOptionAsInteger()

```
int64_t BiometricEvaluation::IO::Compressor::getOptionAsInteger (
    const std::string & optionName ) const
```


Obtain a compressor option as an integer.

Parameters

<i>optionName</i>	Name of the option to obtain.
-------------------	-------------------------------

Returns

Value of compressor option.

Exceptions

Error::ObjectDoesNotExist (p. 457)	The option was never set.
---	---------------------------

G.31.4.16 operator=()

```
Compressor& BiometricEvaluation::IO::Compressor::operator= (
    const Compressor & other ) [delete]
```

Assignment overload (disabled).

Disabled because **Properties** (p. 486) member cannot be assigned.

Parameters

<i>other</i>	Compressor (p. 270) to assign.
--------------	---------------------------------------

Returns

lhs **Compressor** (p. 270).

G.31.4.17 removeOption()

```
void BiometricEvaluation::IO::Compressor::removeOption (
    const std::string & optionName )
```

Remove a compressor option.

Parameters

<i>optionName</i>	Name of the option to remove.
-------------------	-------------------------------

G.31.4.18 setOption() [1/2]

```
void BiometricEvaluation::IO::Compressor::setOption (
    const std::string & optionName,
    const std::string & optionValue )
```

Assign a compressor option.

Will overwrite existing values without warning.

Parameters

<i>optionName</i>	Name of the option to add.
<i>optionValue</i>	Value of the option.

Exceptions

Error::StrategyError (p. 567)	Error (p. 108) setting option.
---	--

G.31.4.19 setOption() [2/2]

```
void BiometricEvaluation::IO::Compressor::setOption (
    const std::string & optionName,
    int64_t optionValue )
```

Assign a compressor option.

Will overwrite existing values without warning.

Parameters

<i>optionName</i>	Name of the option to add.
<i>optionValue</i>	Value of the option.

Exceptions

Error::StrategyError (p. 567)	Error (p. 108) setting option.
---	--

G.32 BiometricEvaluation::Video::Container Class Reference

Representation of a video container.

```
#include <be_video_container.h>
```

Public Member Functions

- **Container** (const **Memory::uint8Array** &buffer)
*Construct a **Container** (p. [281](#)) from a memory buffer.*
- **Container** (const std::shared_ptr< **Memory::uint8Array** > &buffer)
*Construct a **Container** (p. [281](#)) from a memory buffer wrapped in a shared pointer.*
- **Container** (const std::string &filename)
*Construct a **Container** (p. [281](#)) from file.*
- uint32_t **getAudioCount** ()
Obtain the number of audio streams.
- uint32_t **getVideoCount** ()
Obtain the number of video streams.

- `std::unique_ptr< Video::Stream > getVideoStream (uint32_t videoNum)`

*Obtain a video stream from the container. **Video** (p. 163) streams are indexed independently from other streams in the container.*

G.32.1 Detailed Description

Representation of a video container.

The **Container** (p. 281) class represents a single container stream that can be used to access the video and audio components of the stream.

G.32.2 Constructor & Destructor Documentation

G.32.2.1 Container() [1/3]

```
BiometricEvaluation::Video::Container::Container (
    const Memory::uint8Array & buffer )
```

Construct a **Container** (p. 281) from a memory buffer.

Using this constructor can result in buffer memory usage twice that of other constructors.

Exceptions

Error::MemoryError (p. 436)	Error (p. 108) allocating memory for internal buffering.
Error::StrategyError (p. 567)	Other error when reading the container stream.

G.32.2.2 Container() [2/3]

```
BiometricEvaluation::Video::Container::Container (
    const std::shared_ptr< Memory::uint8Array > & buffer )
```

Construct a **Container** (p. 281) from a memory buffer wrapped in a shared pointer.

Applications must not modify the data underlying the AutoArray.

Exceptions

Error::MemoryError (p. 436)	Error (p. 108) allocating memory for internal buffering.
Error::StrategyError (p. 567)	Other error when reading the container stream.

G.32.2.3 Container() [3/3]

```
BiometricEvaluation::Video::Container::Container (
    const std::string & filename )
```

Construct a **Container** (p. 281) from file.

Exceptions

Error::ObjectDoesNotExist (p. 457)	File does not exist.
---	----------------------

Exceptions

<i>Error::MemoryError</i> (p. 436)	Error (p. 108) allocating memory for internal buffering.
<i>Error::StrategyError</i> (p. 567)	Other error when reading the container stream.

G.32.3 Member Function Documentation

G.32.3.1 getVideoStream()

```
std::unique_ptr< Video::Stream> BiometricEvaluation::Video::Container::getVideoStream (
    uint32_t videoNum )
```

Obtain a video stream from the container. **Video** (p. 163) streams are indexed independently from other streams in the container.

Parameters

<i>videoNum</i>	The number of the video stream within the container.
-----------------	--

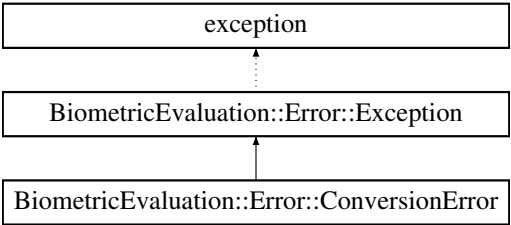
Exceptions

<i>Error::ParameterError</i> (p. 474)	The requested video stream is not available.
---------------------------------------	--

G.33 BiometricEvaluation::Error::ConversionError Class Reference

Error (p. 108) when converting one object into another, a property value from string to int, for example.

```
#include <be_error_exception.h>
Inheritance diagram for BiometricEvaluation::Error::ConversionError:
```



Public Member Functions

- **ConversionError** ()
- **ConversionError** (const std::string &info)

G.33.1 Detailed Description

Error (p. 108) when converting one object into another, a property value from string to int, for example.

G.33.2 Constructor & Destructor Documentation

G.33.2.1 ConversionError() [1/2]

```
BiometricEvaluation::Error::ConversionError::ConversionError ( )
```

Construct a **ConversionError** (p. 283) object with the default information string.

G.33.2.2 ConversionError() [2/2]

```
BiometricEvaluation::Error::ConversionError::ConversionError (
    const std::string & info )
```

Construct a **ConversionError** (p. 283) object with an information string appended to the default information string.

G.34 BiometricEvaluation::Image::Coordinate Struct Reference

A structure to contain a two-dimensional coordinate without a specified origin.

```
#include <be_image.h>
```

Public Member Functions

- **Coordinate** (const uint32_t x=0, const uint32_t y=0, const float xDistance=0, const float yDistance=0)
Create a *Coordinate* (p. 284) struct.

Public Attributes

- uint32_t x
- uint32_t y
- float xDistance
- float yDistance

G.34.1 Detailed Description

A structure to contain a two-dimensional coordinate without a specified origin.

G.34.2 Constructor & Destructor Documentation

G.34.2.1 Coordinate()

```
BiometricEvaluation::Image::Coordinate::Coordinate (
    const uint32_t x = 0,
    const uint32_t y = 0,
    const float xDistance = 0,
    const float yDistance = 0 )
```

Create a **Coordinate** (p. 284) struct.

Parameters

in	x	X-coordinate
----	---	--------------

Parameters

in	<i>y</i>	Y-coordinate
in	<i>xDistance</i>	X-coordinate distance from origin
in	<i>yDistance</i>	Y-coordinate distance from origin

G.34.3 Member Data Documentation

G.34.3.1 *x*

```
uint32_t BiometricEvaluation::Image::Coordinate::x
    X-coordinate
```

G.34.3.2 *xDistance*

```
float BiometricEvaluation::Image::Coordinate::xDistance
    X-coordinate distance from origin
```

G.34.3.3 *y*

```
uint32_t BiometricEvaluation::Image::Coordinate::y
    Y-coordinate
```

G.34.3.4 *yDistance*

```
float BiometricEvaluation::Image::Coordinate::yDistance
    Y-coordinate distance from origin
```

G.35 BiometricEvaluation::Feature::AN2K11EFS::CorePoint Struct Reference

Public Attributes

- **Image::Coordinate** location
- bool **has_cdi**
- int **cdi**
- bool **has_rpu**
- int **rpu**
- bool **has_duy**
- int **duy**

G.36 BiometricEvaluation::Feature::CorePoint Struct Reference

Representation of the core.

```
#include <be_feature_minutiae.h>
```

Public Member Functions

- **CorePoint** (**Image::Coordinate** coordinate, bool has_angle=false, int angle=0)

Create a *CorePoint* (p. 285) struct.

Public Attributes

- **Image::Coordinate** coordinate
- bool **has_angle**
- int **angle**

G.36.1 Detailed Description

Representation of the core.

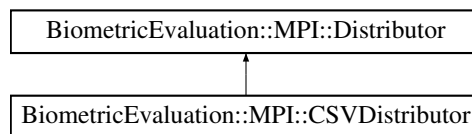
A core has a coordinate and an optional angle. The units for the X/Y coordinate and the angle are specific to the record format represented by an object of this class.

G.37 BiometricEvaluation::MPI::CSVDistributor Class Reference

An implementation of the MPI::Distributor abstraction that distribute lines of a text file via work packages.

```
#include <be_mpi_csvdistributor.h>
```

Inheritance diagram for BiometricEvaluation::MPI::CSVDistributor:



Public Member Functions

- **CSVDistributor** (const std::string &propertiesFileName, const std::string &delimiter="")

Construct a *CSVDistributor* (p. 286) using named properties.

Static Public Attributes

- static const std::string **CHECKPOINTLINECOUNT**
- static const std::string **CHECKPOINTRANDOMSEED**

Protected Member Functions

- void **createWorkPackage** (**MPI::WorkPackage** &workPackage)

Create a work package for distribution.

- void **checkpointSave** (const std::string &reason)

Create a checkpoint state.

- void **checkpointRestore** ()

Restore from a checkpoint state.

G.37.1 Detailed Description

An implementation of the MPI::Distributor abstraction that distribute lines of a text file via work packages.

This class supports checkpointing when an early exit is requested, allowing all workers to complete their current work package. If the input data lines were randomized, the random number generator seed is saved as part of the checkpoint.

On checkpoint restart, if the input data lines are randomized, the seed in the checkpoint must match the current seed; else an exception is thrown. If the checkpoint contains a seed, and the input is not currently randomized, and exception is thrown. See [MPI::CSVResources](#) (p. 291).

G.37.2 Constructor & Destructor Documentation

G.37.2.1 CSVDistributor()

```
BiometricEvaluation::MPI::CSVDistributor::CSVDistributor (
    const std::string & propertiesFileName,
    const std::string & delimiter = "" )
```

Construct a **CSVDistributor** (p. 286) using named properties.

Parameters

in	<i>propertiesFileName</i>	The file containing the properties.
in	<i>delimiter</i>	Delimiter used to tokenize lines read from CSV.

G.37.3 Member Function Documentation

G.37.3.1 checkpointRestore()

```
void BiometricEvaluation::MPI::CSVDistributor::checkpointRestore ( ) [protected], [virtual]
```

Restore from a checkpoint state.

Implementations of this class use a checkpoint state to move the data sequence cursor to a point past data that has been previously distributed. The **MPI** (p. 145) **Framework** (p. 117) calls this method prior to the start of distributing work packages.

Implements **BiometricEvaluation::MPI::Distributor** (p. 306).

G.37.3.2 checkpointSave()

```
void BiometricEvaluation::MPI::CSVDistributor::checkpointSave (
    const std::string & reason ) [protected], [virtual]
```

Create a checkpoint state.

Implementations of this class create a checkpoint state that captures enough information to allow the implementation to move the data sequence cursor to a point past data that has been previously distributed. The **MPI** (p. 145) **Framework** (p. 117) calls this method when a premature shutdown is requested.

Parameters

<i>reason</i>	A string giving the reason for the checkpoint to be saved.
---------------	--

Implements **BiometricEvaluation::MPI::Distributor** (p. 307).

G.37.3.3 createWorkPackage()

```
void BiometricEvaluation::MPI::CSVDistributor::createWorkPackage (
    MPI::WorkPackage & workPackage ) [protected], [virtual]
```

Create a work package for distribution.

Implementations of this class create a work package to encapsulate the specific data type that is to be distributed.

Implements **BiometricEvaluation::MPI::Distributor** (p. 307).

G.37.4 Member Data Documentation

G.37.4.1 CHECKPOINTLINECOUNT

```
const std::string BiometricEvaluation::MPI::CSVDistributor::CHECKPOINTLINECOUNT [static]
```

The number of lines that were distributed, "Line Count".

G.37.4.2 CHECKPOINTRANDOMSEED

```
const std::string BiometricEvaluation::MPI::CSVDistributor::CHECKPOINTRANDOMSEED [static]
```

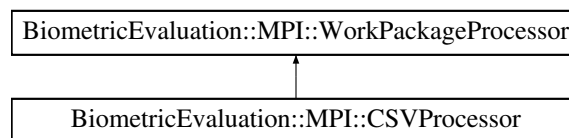
The seed used to randomize the input CSV file lines, "Random Seed".

G.38 BiometricEvaluation::MPI::CSVProcessor Class Reference

An implementation of a work package processor that will extract lines (and optionally tokenize) a line from a CSV text file.

```
#include <be_mpi_csvprocessor.h>
```

Inheritance diagram for BiometricEvaluation::MPI::CSVProcessor:



Public Member Functions

- **CSVProcessor** (const std::string &propertiesFileName)
Construct a work package processor with the given properties.
- virtual void **processLine** (const uint64_t lineNum, const std::string &line)=0
Method implemented by child classes to perform an action using each record from the Record Store.
- virtual std::shared_ptr< **WorkPackageProcessor** > **newProcessor** (std::shared_ptr< **IO::Logsheet** > &logsheet)=0
Obtain an object that will process work packages. This method is part of the factory personality.
- virtual void **performInitialization** (std::shared_ptr< **IO::Logsheet** > &logsheet)=0
Initialization function to be called before work is distributed to the work package processor.

- void **processWorkPackage** (**MPI::WorkPackage** &workPackage)
Process (p. 150) the data contents of the work package. This method is part of the worker personality.

Protected Member Functions

- std::shared_ptr< **MPI::CSVResources** > **getResources** ()

G.38.1 Detailed Description

An implementation of a work package processor that will extract lines (and optionally tokenize) a line from a CSV text file.

Subclasses of this abstract class must implement the method to process the lines.

G.38.2 Constructor & Destructor Documentation

G.38.2.1 CSVProcessor()

```
BiometricEvaluation::MPI::CSVProcessor::CSVProcessor (
    const std::string & propertiesFileName )
```

Construct a work package processor with the given properties.

A **CSVProcessor** (p. 288) uses a text file to retrieve the data to be processed.

Note

Subclasses of this class should not manually read lines from the CSV.

The size of a single value item is limited to 2^{64} octets. If the size of the value item is larger, behavior is undefined.

Parameters

in	<i>propertiesFileName</i>	The name of the file containing the properties for this object.
----	---------------------------	---

Exceptions

Error::Exception (p. 310)	An error occurred, usually due to missing or incorrect properties.
----------------------------------	--

G.38.3 Member Function Documentation

G.38.3.1 newProcessor()

```
virtual std::shared_ptr< WorkPackageProcessor> BiometricEvaluation::MPI::CSVProcessor::new←
Processor (
```

```
    std::shared_ptr< IO::Logsheet > & logsheet ) [pure virtual]
```

Obtain an object that will process work packages. This method is part of the factory personality.

Parameters

<i>logsheet</i>	A shared pointer to the IO::Logsheet (p. 420) that may be used to save messages generated by the object.
-----------------	---

Returns

A shared pointer to the work package processor.

Note

This method should always create a non-null **WorkPackageProcessor** (p. 608). If an error occurs during construction, throw a **Error::Exception** (p. 310) with a message to be caught and logged.

Implements **BiometricEvaluation::MPI::WorkPackageProcessor** (p. 609).

G.38.3.2 performInitialization()

```
virtual void BiometricEvaluation::MPI::CSVProcessor::performInitialization (
    std::shared_ptr< IO::Logsheet > & logsheet ) [pure virtual]
```

Initialization function to be called before work is distributed to the work package processor.

Implementations of this class can use this function to do any processing necessary before work is given to the processor, pre-forking.

This method is part of the factory personality. All state that is to be common across all package processor objects can be initialized in this method.

Parameters

<i>logsheet</i>	A shared pointer to the IO::Logsheet (p. 420) that may be used to save messages generated by the object.
-----------------	---

Exceptions

Error::Exception (p. 310)	An implementation specific error occurred. The exception string will be logged by the Framework (p. 117).
----------------------------------	--

Implements **BiometricEvaluation::MPI::WorkPackageProcessor** (p. 609).

G.38.3.3 processLine()

```
virtual void BiometricEvaluation::MPI::CSVProcessor::processLine (
    const uint64_t lineNum,
    const std::string & line ) [pure virtual]
```

Method implemented by child classes to perform an action using each record from the Record Store.

The source RecordStore must be accessible to the the implementation as the value for each key is not included.

Parameters

in	<i>lineNum</i>	The line number from the input file (1-based).
in	<i>line</i>	The key associated with the record that is to be processed.

Exceptions

Error::Exception (p. 310)	An error occurred processing the record: Missing record, input/output error, or memory allocation.
----------------------------------	--

G.38.3.4 processWorkPackage()

```
void BiometricEvaluation::MPI::CSVProcessor::processWorkPackage (
    MPI::WorkPackage & workPackage ) [virtual]
```

Process (p. 150) the data contents of the work package. This method is part of the worker personality.

Parameters

in	<i>workPackage</i>	The work package.
----	--------------------	-------------------

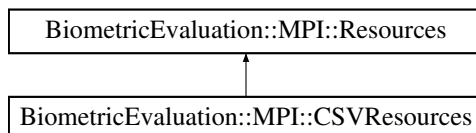
Exceptions

Error::Exception (p. 310)	An fatal error occurred when processing the work package; the processing responsible for this object should shut down.
----------------------------------	--

Implements **BiometricEvaluation::MPI::WorkPackageProcessor** (p. 610).

G.39 BiometricEvaluation::MPI::CSVResources Class Reference

Inheritance diagram for BiometricEvaluation::MPI::CSVResources:



Public Member Functions

- **CSVResources** (const std::string &propertiesFileName)
- uint32_t **getChunkSize** () const
- bool **useBuffer** () const

Obtain whether or not the entire CSV was read into memory at construction.
- bool **randomizeLines** () const

If using buffer, whether or not to randomize how lines from the buffer are iterated.

- uint64_t **getNumRemainingLines** () const

*Obtain the number of lines that have not yet been read from **readLine**() (p. 293) by a **Distributor** (p. 305).*

- std::string **getDelimiter** () const
- std::pair< uint64_t, std::string > **readLine** ()

Obtain the next line from a buffer of file stream.

- uint64_t **getNumLines** () const

Obtain number of lines of input.

- std::mt19937_64::result_type **getRandomSeed** () const

Obtain the seed used to shuffle lines.

Static Public Member Functions

- static std::vector< std::string > **getRequiredProperties** ()
- static std::vector< std::string > **getOptionalProperties** ()

Static Public Attributes

- static const std::string **INPUTCSVPROPERTY**
- static const std::string **CHUNKSIZEPROPERTY**
- static const std::string **USEBUFFERPROPERTY**
- static const std::string **RANDOMIZEPROPERTY**
- static const std::string **RANDOMSEEDPROPERTY**
- static const std::string **DELIMITERPROPERTY**
- static const std::string **TRIMPROPERTY**

G.39.1 Member Function Documentation

G.39.1.1 getDelimiter()

```
std::string BiometricEvaluation::MPI::CSVResources::getDelimiter ( ) const
```

Returns

Delimiter used to tokenize sent lines.

G.39.1.2 getNumLines()

```
uint64_t BiometricEvaluation::MPI::CSVResources::getNumLines ( ) const
```

Obtain number of lines of input.

Returns

Number of lines of input to send.

Exceptions

Error::StrategyError (p. 567)	Neither CSV file open nor CSV buffer populated.
--------------------------------------	---

G.39.1.3 getNumRemainingLines()

```
uint64_t BiometricEvaluation::MPI::CSVResources::getNumRemainingLines ( ) const
```

Obtain the number of lines that have not yet been read from **readLine()** (p. 293) by a **Distributor** (p. 305).

Returns

Number of lines that have not been distributed.

G.39.1.4 getRandomSeed()

```
std::mt19937_64::result_type BiometricEvaluation::MPI::CSVResources::getRandomSeed ( ) const
```

Obtain the seed used to shuffle lines.

Returns

Seed used to shuffle lines.

Exceptions

Error::StrategyError (p. 567)	Lines not randomized.
--------------------------------------	-----------------------

G.39.1.5 randomizeLines()

```
bool BiometricEvaluation::MPI::CSVResources::randomizeLines ( ) const
```

If using buffer, whether or not to randomize how lines from the buffer are iterated.

Returns

true if RANDOMIZEPROPERTY and USEBUFFERPROPERTY are true, false otherwise.

G.39.1.6 readLine()

```
std::pair<uint64_t, std::string> BiometricEvaluation::MPI::CSVResources::readLine ( )
```

Obtain the next line from a buffer of file stream.

Note

If `_randomizeLines` is true, sequential calls to this method will not necessarily return sequential lines.

Returns

The next line from buffer or file stream and the line number in the file where the line is from.

Exceptions

Error::StrategyError (p. 567)	Error (p. 108) with the file stream.
Error::ObjectDoesNotExist (p. 457)	File stream or buffer is exhausted.

G.39.1.7 useBuffer()

```
bool BiometricEvaluation::MPI::CSVResources::useBuffer ( ) const
```

Obtain whether or not the entire CSV was read into memory at construction.

Returns

true if the entire INPUTCSVPROPERTY was read into memory at construction, false if an ifstream is kept open.

G.39.2 Member Data Documentation**G.39.2.1 CHUNKSIZEPROPERTY**

```
const std::string BiometricEvaluation::MPI::CSVResources::CHUNKSIZEPROPERTY [static]
```

Number of lines sent in succession

G.39.2.2 DELIMITERPROPERTY

```
const std::string BiometricEvaluation::MPI::CSVResources::DELIMITERPROPERTY [static]
```

Delimiter to tokenize sent lines

G.39.2.3 INPUTCSVPROPERTY

```
const std::string BiometricEvaluation::MPI::CSVResources::INPUTCSVPROPERTY [static]
```

Text (p. 154) file to read

G.39.2.4 RANDOMIZEPROPERTY

```
const std::string BiometricEvaluation::MPI::CSVResources::RANDOMIZEPROPERTY [static]
```

Randomly iterate buffer

G.39.2.5 RANDOMSEEDPROPERTY

```
const std::string BiometricEvaluation::MPI::CSVResources::RANDOMSEEDPROPERTY [static]
```

Seed for randomization

G.39.2.6 TRIMPROPERTY

```
const std::string BiometricEvaluation::MPI::CSVResources::TRIMPROPERTY [static]
```

Trim whitespace from lines read

G.39.2.7 USEBUFFERPROPERTY

```
const std::string BiometricEvaluation::MPI::CSVResources::USEBUFFERPROPERTY [static]
```

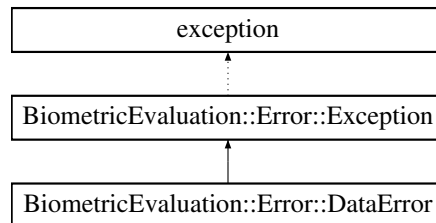
Read file into buffer first, or read from file

G.40 BiometricEvaluation::Error::DataError Class Reference

Error (p. 108) when reading data from an external source.

```
#include <be_error_exception.h>
```

Inheritance diagram for BiometricEvaluation::Error::DataError:



Public Member Functions

- **DataError** ()
- **DataError** (const std::string &info)

G.40.1 Detailed Description

Error (p. 108) when reading data from an external source.

Typically occurs when reading data from a standard record, ANST/NIST 2000, for example, and a required field is missing, or a field has invalid data.

G.40.2 Constructor & Destructor Documentation

G.40.2.1 DataError() [1/2]

```
BiometricEvaluation::Error::DataError::DataError ( )
```

Construct a **DataError** (p. 295) object with the default information string.

G.40.2.2 DataError() [2/2]

```
BiometricEvaluation::Error::DataError::DataError (
    const std::string & info )
```

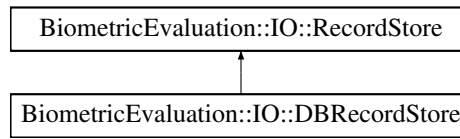
Construct a **DataError** (p. 295) object with an information string appended to the default information string.

G.41 BiometricEvaluation::IO::DBRecordStore Class Reference

A class that implements **IO::RecordStore** (p. 504) using a Berkeley DB database as the underlying record storage system.

```
#include <be_io_dbrecstore.h>
```

Inheritance diagram for BiometricEvaluation::IO::DBRecordStore:



Public Member Functions

- **DBRecordStore** (const std::string &pathname, const std::string &description)
- **DBRecordStore** (const std::string &pathname, **IO::Mode** mode= **IO::Mode::ReadOnly**)
- **Memory::uint8Array read** (const std::string &key) const override
Read a complete record from a store.
- void **insert** (const std::string &key, const void *const data, const uint64_t size) override
- void **remove** (const std::string &key) override
- uint64_t **length** (const std::string &key) const override
- void **flush** (const std::string &key) const override
- **RecordStore::Record sequence** (int cursor= **BE_RECSTORE_SEQ_NEXT**) override
*Sequence through a **RecordStore** (p. 504), returning the key/data pairs.*
- std::string **sequenceKey** (int cursor= **BE_RECSTORE_SEQ_NEXT**) override
*Sequence through a **RecordStore** (p. 504), returning the key.*
- void **setCursorAtKey** (const std::string &key) override
- void **move** (const std::string &pathname) override
*Move the **RecordStore** (p. 504).*
- uint64_t **getSpaceUsed** () const override
Obtain real storage utilization.
- void **sync** () const override
- unsigned int **getCount** () const override
- std::string **getPathname** () const override
- std::string **getDescription** () const override
- void **changeDescription** (const std::string &description) override
- **DBRecordStore** (const **DBRecordStore** &)=delete
- **DBRecordStore & operator=** (const **DBRecordStore** &)=delete

Additional Inherited Members

G.41.1 Detailed Description

A class that implements **IO::RecordStore** (p. 504) using a Berkeley DB database as the underlying record storage system.

G.41.2 Constructor & Destructor Documentation

G.41.2.1 DBRecordStore() [1/2]

```
BiometricEvaluation::IO::DBRecordStore::DBRecordStore (
    const std::string & pathname,
    const std::string & description )
```

Create a new **DBRecordStore** (p. 295), read/write mode.

Parameters

in	<i>pathname</i>	The directory where the store will be created.
in	<i>description</i>	The store's description.

Exceptions

Error::ObjectExists (p. 458)	The store already exists.
Error::StrategyError (p. 567)	An error occurred when accessing the underlying file system.

G.41.2.2 DBRecordStore() [2/2]

```
BiometricEvaluation::IO::DBRecordStore::DBRecordStore (
    const std::string & pathname,
    IO::Mode mode = IO::Mode::ReadOnly )
    Open an existing DBRecordStore (p. 295).
```

Parameters

in	<i>name</i>	The path name of the store.
in	<i>mode</i>	Open mode, read-only or read-write.

Exceptions

Error::ObjectDoesNotExist (p. 457)	The store does not exist.
Error::StrategyError (p. 567)	An error occurred when accessing the underlying file system.

G.41.3 Member Function Documentation

G.41.3.1 changeDescription()

```
void BiometricEvaluation::IO::DBRecordStore::changeDescription (
    const std::string & description ) [override], [virtual]
    Change the description of the RecordStore (p. 504).
```

Parameters

in	<i>description</i>	The new description.
----	--------------------	----------------------

Exceptions

Error::StrategyError (p. 567)	An error occurred when using the underlying storage system.
---	---

Implements **BiometricEvaluation::IO::RecordStore** (p. 506).

G.41.3.2 flush()

```
void BiometricEvaluation::IO::DBRecordStore::flush (
    const std::string & key ) const [override], [virtual]
```

Commit the record's data to storage.

Parameters

in	key	The key of the record to be flushed.
----	-----	--------------------------------------

Exceptions

Error::ObjectDoesNotExist (p. 457)	A record for the key does not exist.
Error::StrategyError (p. 567)	An error occurred when using the underlying storage system.

Implements **BiometricEvaluation::IO::RecordStore** (p. 508).

G.41.3.3 getCount()

```
unsigned int BiometricEvaluation::IO::DBRecordStore::getCount ( ) const [override], [virtual]
```

Obtain the number of items in the **RecordStore** (p. 504).

Returns

The number of items in the **RecordStore** (p. 504).

Implements **BiometricEvaluation::IO::RecordStore** (p. 508).

G.41.3.4 getDescription()

```
std::string BiometricEvaluation::IO::DBRecordStore::getDescription ( ) const [override], [virtual]
```

Obtain a textual description of the **RecordStore** (p. 504).

Returns

The **RecordStore** (p. 504)'s description.

Implements **BiometricEvaluation::IO::RecordStore** (p. 508).

G.41.3.5 getPathname()

```
std::string BiometricEvaluation::IO::DBRecordStore::getPathname ( ) const [override], [virtual]
```

Return the path name of the **RecordStore** (p. 504).

Returns

Where in the file system the **RecordStore** (p. 504) is located.

Implements **BiometricEvaluation::IO::RecordStore** (p. 509).

G.41.3.6 getSpaceUsed()

```
uint64_t BiometricEvaluation::IO::DBRecordStore::getSpaceUsed ( ) const [override], [virtual]
```

Obtain real storage utilization.

The amount of disk space used, for example. This is the actual space allocated by the underlying storage mechanism, in bytes.

Returns

The amount of backing storage used by the **RecordStore** (p. 504).

Exceptions

Error::StrategyError (p. 567)	An error occurred when using the underlying storage system.
--------------------------------------	---

Implements **BiometricEvaluation::IO::RecordStore** (p. 509).

G.41.3.7 insert()

```
void BiometricEvaluation::IO::DBRecordStore::insert (
    const std::string & key,
    const void *const data,
    const uint64_t size ) [override], [virtual]
```

Insert a record into the store.

Parameters

in	<i>key</i>	The key of the record to be inserted.
in	<i>data</i>	The data for the record.
in	<i>size</i>	The size of the record, in bytes.

Exceptions

Error::ObjectExists (p. 458)	A record with the given key is already present.
Error::StrategyError (p. 567)	The RecordStore (p. 504) is opened read-only, or an error occurred when using the underlying storage system.

Implements **BiometricEvaluation::IO::RecordStore** (p. 510).

G.41.3.8 length()

```
uint64_t BiometricEvaluation::IO::DBRecordStore::length (
    const std::string & key ) const [override], [virtual]
```

Return the length of a record.

Parameters

in	<i>key</i>	The key of the record.
----	------------	------------------------

Returns

The record length.

Exceptions

Error::ObjectDoesNotExist (p. 457)	A record for the key does not exist.
Error::StrategyError (p. 567)	An error occurred when using the underlying storage system.

Implements **BiometricEvaluation::IO::RecordStore** (p. 510).

G.41.3.9 move()

```
void BiometricEvaluation::IO::DBRecordStore::move (
    const std::string & pathname ) [override], [virtual]
```

Move the **RecordStore** (p. 504).

The **RecordStore** (p. 504) can be moved to a new path in the file system.

Parameters

in	<i>pathname</i>	The new path of the RecordStore (p. 504).
----	-----------------	--

Exceptions

Error::StrategyError (p. 567)	An error occurred when using the underlying storage system.
--------------------------------------	---

Implements **BiometricEvaluation::IO::RecordStore** (p. 511).

G.41.3.10 read()

```
Memory::uint8Array BiometricEvaluation::IO::DBRecordStore::read (
    const std::string & key ) const [override], [virtual]
```

Read a complete record from a store.

The AutoArray will be resized to match the size of the data.

Parameters

in	<i>key</i>	The key of the record to be read.
----	------------	-----------------------------------

Returns

The record associated with the key.

Exceptions

Error::ObjectDoesNotExist (p. 457)	A record for the key does not exist.
Error::StrategyError (p. 567)	An error occurred when using the underlying storage system.

Implements **BiometricEvaluation::IO::RecordStore** (p. 512).

G.41.3.11 remove()

```
void BiometricEvaluation::IO::DBRecordStore::remove (
    const std::string & key ) [override], [virtual]
```

Remove a record from the store.

Parameters

in	key	The key of the record to be removed.
----	-----	--------------------------------------

Exceptions

Error::ObjectDoesNotExist (p. 457)	A record for the key does not exist.
Error::StrategyError (p. 567)	An error occurred when using the underlying storage system.

Implements **BiometricEvaluation::IO::RecordStore** (p. 513).

G.41.3.12 sequence()

```
RecordStore::Record BiometricEvaluation::IO::DBRecordStore::sequence (
    int cursor = BE_RECSTORE_SEQ_NEXT ) [override], [virtual]
```

Sequence through a **RecordStore** (p. 504), returning the key/data pairs.

Sequencing means to start at some point in the store and return the record, then repeatedly calling the function to return the next record. The starting point is typically the first record, and is set to that when the **RecordStore** (p. 504) object is created. The starting point can be reset by calling this method with the cursor parameter set to **BE_RECSTORE_SEQ_START**.

Parameters

in	cursor	The location within the sequence of the key/data pair to return.
----	--------	--

Returns

The record that is currently in sequence.

Exceptions

Error::ObjectDoesNotExist (p. 457)	End of sequencing.
Error::StrategyError (p. 567)	An error occurred when using the underlying storage system.

Implements **BiometricEvaluation::IO::RecordStore** (p. 515).

G.41.3.13 sequenceKey()

```
std::string BiometricEvaluation::IO::DBRecordStore::sequenceKey (
    int cursor = BE_RECSTORE_SEQ_NEXT ) [override], [virtual]
```

Sequence through a **RecordStore** (p. 504), returning the key.

Sequencing means to start at some point in the store and return the key, then repeatedly calling the function to return the next key. The starting point is typically the first record, and is set to that when the **RecordStore** (p. 504) object is created. The starting point can be reset by calling this method with the cursor parameter set to **BE_RECSTORE_SEQ_START**.

Parameters

in	<i>cursor</i>	The location within the sequence of the key/data pair to return.
----	---------------	--

Returns

The key of the currently sequenced record.

Exceptions

Error::ObjectDoesNotExist (p. 457)	End of sequencing.
Error::StrategyError (p. 567)	An error occurred when using the underlying storage system.

Implements **BiometricEvaluation::IO::RecordStore** (p. 515).

G.41.3.14 setCursorAtKey()

```
void BiometricEvaluation::IO::DBRecordStore::setCursorAtKey (
    const std::string & key ) [override], [virtual]
```

Set the sequence cursor to an arbitrary position within the **RecordStore** (p. 504), starting at key. Key will be the first record returned from the next call to **sequence()** (p. 301).

Parameters

in	<i>key</i>	The key of the record which will be returned by the first subsequent call to sequence() (p. 301).
----	------------	--

Exceptions

Error::ObjectDoesNotExist (p. 457)	A record for the key does not exist.
Error::StrategyError (p. 567)	An error occurred when using the underlying storage system.

Implements **BiometricEvaluation::IO::RecordStore** (p. 516).

G.41.3.15 sync()

```
void BiometricEvaluation::IO::DBRecordStore::sync ( ) const [override], [virtual]
```


Synchronize the entire record store to persistent storage.

Exceptions

Error::StrategyError (p. 567)	An error occurred when using the underlying storage system.
---	---

Implements **BiometricEvaluation::IO::RecordStore** (p. [516](#)).

G.42 BiometricEvaluation::Feature::AN2K11EFS::DeltaPoint Struct Reference

Representation of an extended feature set delta.

```
#include <be_feature_an2k11efs.h>
```

Public Attributes

- **Image::Coordinate** location
- bool **has_dup**
- int **dup**
- bool **has_dlf**
- int **dlf**
- bool **has_drt**
- int **drt**
- bool **has_dtp**
- DeltaType **dtp**
- bool **has_rpu**
- int **rpu**
- bool **has_duu**
- int **duu**
- bool **has_dul**
- int **dul**
- bool **has_dur**
- int **dur**

G.42.1 Detailed Description

Representation of an extended feature set delta.

G.43 BiometricEvaluation::Feature::DeltaPoint Struct Reference

Representation of the delta.

```
#include <be_feature_minutiae.h>
```

Public Member Functions

- **DeltaPoint** (**Image::Coordinate** coordinate, bool has_angle=false, int angle1=0, int angle2=0, int angle3=0)

Create a *DeltaPoint* (p. [304](#)) struct.

Public Attributes

- **Image::Coordinate** coordinate
- bool **has_angle**
- int **angle1**
- int **angle2**
- int **angle3**

G.43.1 Detailed Description

Representation of the delta.

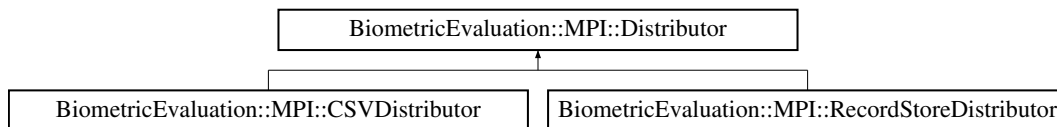
A delta has a coordinate and an optional angle. The units for the X/Y coordinate and the angle are specific to the record format represented by an object of this class.

G.44 BiometricEvaluation::MPI::Distributor Class Reference

A class to represent an **MPI** (p. 145) task that distributes work to other tasks.

```
#include <be_mpi_distributor.h>
```

Inheritance diagram for BiometricEvaluation::MPI::Distributor:



Public Member Functions

- **Distributor** (const std::string &propertiesFileName)
Constructor with properties file name.
- void **start** ()
*Start of **MPI** (p. 145) processing for the distributor.*

Static Public Attributes

- static const std::string **CHECKPOINTFILENAME**
- static const std::string **CHECKPOINTREASON**
- static const std::string **CHECKPOINTPID**

Protected Member Functions

- virtual void **createWorkPackage** (**MPI::WorkPackage** &workPackage)=0
Create a work package for distribution.
- virtual void **checkpointSave** (const std::string &reason)=0
Create a checkpoint state.
- virtual void **checkpointRestore** ()=0
Restore from a checkpoint state.
- std::shared_ptr< **IO::Logsheet** > **getLogsheet** () const
Get access to the Logsheet object.

- `std::shared_ptr< IO::PropertiesFile > getCheckpointData () const`
Get access to the checkpoint data object.

G.44.1 Detailed Description

A class to represent an **MPI** (p. 145) task that distributes work to other tasks.

A **Distributor** (p. 305) object is based on a set of properties contained in a file. This class must be subclassed and an implementation of the **createWorkPackage()** (p. 307) method provided.

The distributor sends an **MPI** (p. 145) message to each receiver object indicating whether it should start and ready for accepting work packages, or proceed immediately to the shutdown state. Failure to start the **Distributor** (p. 305) object will result in the entire **MPI** (p. 145) job shutting down before any work is done.

If the Logsheet URL property is set, log messages will be written to that sheet. Otherwise, log messages will be written to a Null Logsheet.

See also

IO::Properties (p. 486)
MPI::Receiver (p. 498)
MPI::WorkPackage (p. 606)

G.44.2 Constructor & Destructor Documentation

G.44.2.1 Distributor()

```
BiometricEvaluation::MPI::Distributor::Distributor (
    const std::string & propertiesFileName )
```

Constructor with properties file name.

Parameters

in	<i>propertiesFileName</i>	The name of the file containing the properties for the new object.
----	---------------------------	--

Exceptions

Error::Exception (p. 310)	An error occurred, possibly due to missing or invalid properties.
----------------------------------	---

G.44.3 Member Function Documentation

G.44.3.1 checkpointRestore()

```
virtual void BiometricEvaluation::MPI::Distributor::checkpointRestore ( ) [protected], [pure virtual]
```

Restore from a checkpoint state.

Implementations of this class use a checkpoint state to move the data sequence cursor to a point past data that has been previously distributed. The **MPI** (p. 145) **Framework** (p. 117) calls this method prior to the start of distributing work packages.

Implemented in **BiometricEvaluation::MPI::RecordStoreDistributor** (p. 519), and **BiometricEvaluation::MPI::CSVDistributor** (p. 287).

G.44.3.2 checkpointSave()

```
virtual void BiometricEvaluation::MPI::Distributor::checkpointSave (
    const std::string & reason ) [protected], [pure virtual]
```

Create a checkpoint state.

Implementations of this class create a checkpoint state that captures enough information to allow the implementation to move the data sequence cursor to a point past data that has been previously distributed. The **MPI** (p. 145) **Framework** (p. 117) calls this method when a premature shutdown is requested.

Parameters

<i>reason</i>	A string giving the reason for the checkpoint to be saved.
---------------	--

Implemented in **BiometricEvaluation::MPI::RecordStoreDistributor** (p. 519), and **BiometricEvaluation::MPI::CSVDistributor** (p. 287).

G.44.3.3 createWorkPackage()

```
virtual void BiometricEvaluation::MPI::Distributor::createWorkPackage (
    MPI::WorkPackage & workPackage ) [protected], [pure virtual]
```

Create a work package for distribution.

Implementations of this class create a work package to encapsulate the specific data type that is to be distributed.

Implemented in **BiometricEvaluation::MPI::RecordStoreDistributor** (p. 519), and **BiometricEvaluation::MPI::CSVDistributor** (p. 288).

G.44.3.4 getCheckpointData()

```
std::shared_ptr< IO::PropertiesFile> BiometricEvaluation::MPI::Distributor::getCheckpointData
( ) const [protected]
```

Get access to the checkpoint data object.

Returns

A shared pointer for the checkpoint data object.

G.44.3.5 getLogsheet()

```
std::shared_ptr< IO::Logsheet> BiometricEvaluation::MPI::Distributor::getLogsheet ( ) const [protected]
```

Get access to the Logsheet object.

Returns

A shared pointer for the Logsheet object.

G.44.3.6 start()

```
void BiometricEvaluation::MPI::Distributor::start ( )
```

Start of **MPI** (p. 145) processing for the distributor.

Once started, the distributor will send a message to each receiver task telling it to start and wait for status back from each receiver.

G.44.4 Member Data Documentation

G.44.4.1 CHECKPOINTFILENAME

```
const std::string BiometricEvaluation::MPI::Distributor::CHECKPOINTFILENAME [static]
```

The name of the checkpoint properties file, "Distributor.chk".

G.44.4.2 CHECKPOINTPID

```
const std::string BiometricEvaluation::MPI::Distributor::CHECKPOINTPID [static]
```

The process ID of the checkpointing **Distributor** (p. 305) process, "PID".

G.44.4.3 CHECKPOINTREASON

```
const std::string BiometricEvaluation::MPI::Distributor::CHECKPOINTREASON [static]
```

The reason string given for the checkpoint to be taken, "Reason".

G.45 BiometricEvaluation::DataInterchange::AN2KRecord::Domain↵ Name Struct Reference

Representation of a domain name for the user-defined Type-2 logical record implementation.

```
#include <be_data_interchange_an2k.h>
```

Public Member Functions

- **DomainName** (std::string **identifier**="", std::string **version**="")

Create a *DomainName* (p. 308) struct.

Public Attributes

- std::string **identifier**
- std::string **version**

G.45.1 Detailed Description

Representation of a domain name for the user-defined Type-2 logical record implementation.

G.45.2 Constructor & Destructor Documentation

G.45.2.1 DomainName()

BiometricEvaluation::DataInterchange::AN2KRecord::DomainName::DomainName (
 std::string identifier = "",
 std::string version = "") [inline]
Create a DomainName (p. 308) struct.

Parameters

identifier	Unique identifier for agency, entity, or implementation.
version	Optional unique version number of the implementation of the identifier.

G.45.3 Member Data Documentation

G.45.3.1 identifier

std::string BiometricEvaluation::DataInterchange::AN2KRecord::DomainName::identifier
Unique identifier for agency, entity, or implementation.

G.45.3.2 version

std::string BiometricEvaluation::DataInterchange::AN2KRecord::DomainName::version
Optional version of the implementation

G.46 BiometricEvaluation::Feature::AN2K7Minutiae::PatternClassification::Entry Struct Reference

Public Member Functions

- Entry (bool standard, std::string code)

Public Attributes

- bool standard
- std::string code

G.46.1 Constructor & Destructor Documentation

G.46.1.1 Entry()

BiometricEvaluation::Feature::AN2K7Minutiae::PatternClassification::Entry::Entry (
 bool standard,
 std::string code)
Create an Entry (p. 309) struct.

Parameters

<i>standard</i>	Whether or not code is a standard AN2K pattern classification code.
<i>code</i>	AN2K or user-defined pattern classification code.

G.46.2 Member Data Documentation

G.46.2.1 code

`std::string BiometricEvaluation::Feature::AN2K7Minutiae::PatternClassification::Entry::code`
AN2K or user-defined pattern classification code.

G.46.2.2 standard

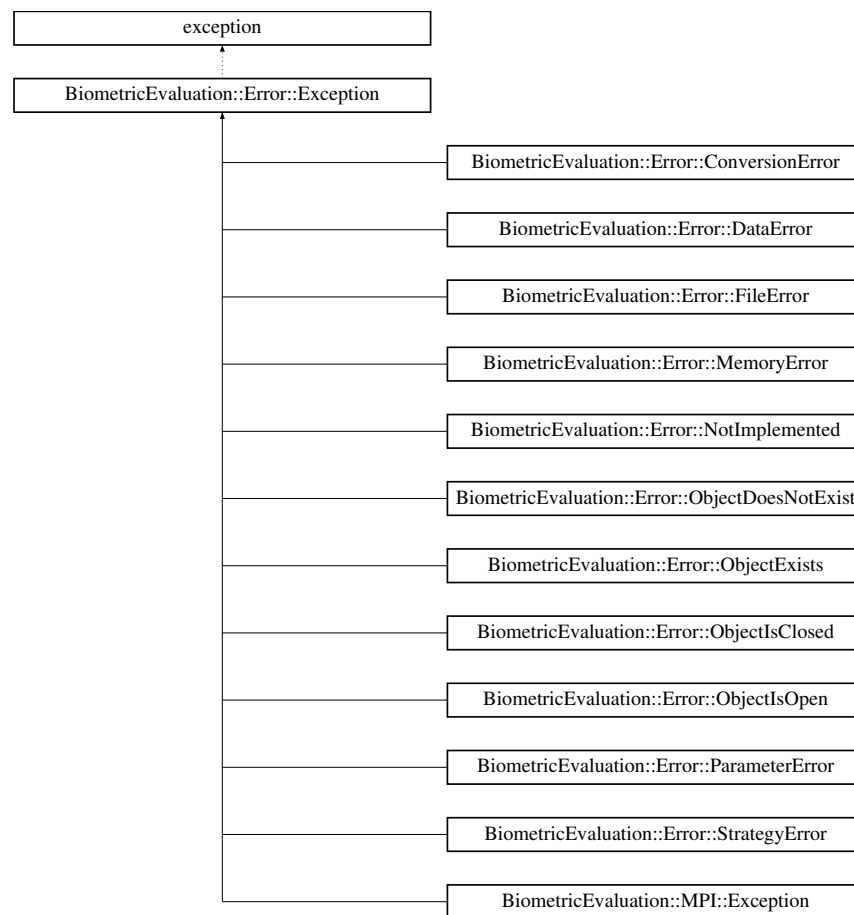
`bool BiometricEvaluation::Feature::AN2K7Minutiae::PatternClassification::Entry::standard`
Whether code is a standard AN2K pattern classification code.

G.47 BiometricEvaluation::Error::Exception Class Reference

The parent class of all **BiometricEvaluation** (p. 107) exceptions.

```
#include <be_error_exception.h>
```

Inheritance diagram for BiometricEvaluation::Error::Exception:



Public Member Functions

- **Exception** ()
- **Exception** (std::string info)
- const char * **what** () const noexcept
- const std::string **whatString** () const noexcept

G.47.1 Detailed Description

The parent class of all **BiometricEvaluation** (p. 107) exceptions.

The classes derived from this class will have a default information string set indicating the type of exception. Any additional information string is appended to that string.

G.47.2 Constructor & Destructor Documentation

G.47.2.1 Exception() [1/2]

`BiometricEvaluation::Error::Exception::Exception ()`

Construct an **Exception** (p. 310) object without an information string.

G.47.2.2 Exception() [2/2]

```
BiometricEvaluation::Error::Exception::Exception (
    std::string info )
```

Construct an **Exception** (p. 310) object with an information string.

Parameters

in	info	The information string associated with the exception.
----	------	---

G.47.3 Member Function Documentation

G.47.3.1 what()

```
const char* BiometricEvaluation::Error::Exception::what ( ) const [noexcept]
```

Obtain the information string associated with the exception.

Returns

The information string as a char array.

G.47.3.2 whatString()

```
const std::string BiometricEvaluation::Error::Exception::whatString ( ) const [noexcept]
```

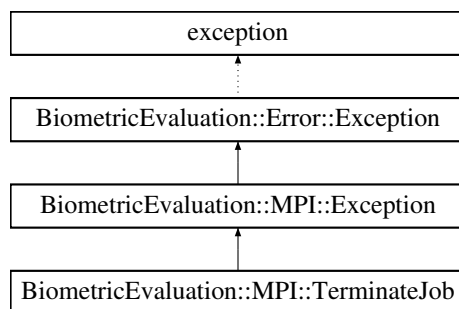
Obtain the information string associated with the exception.

Returns

The information string.

G.48 BiometricEvaluation::MPI::Exception Class Reference

Inheritance diagram for BiometricEvaluation::MPI::Exception:



Public Member Functions

- **Exception** ()
- **Exception** (std::string info)
Constructor.
- virtual **~Exception** () noexcept=default

G.48.1 Constructor & Destructor Documentation

G.48.1.1 Exception() [1/2]

BiometricEvaluation::MPI::Exception::Exception ()
Construct with default information string.

G.48.1.2 Exception() [2/2]

BiometricEvaluation::MPI::Exception::Exception (
std::string info)
Constructor.

Parameters

<i>info</i>	Custom information string. Will be appended to the default information string.
-------------	--

G.48.1.3 ~Exception()

virtual BiometricEvaluation::MPI::Exception::~~Exception () [virtual], [default], [noexcept]
Destructor.
Reimplemented from **BiometricEvaluation::Error::Exception** (p. 310).

G.49 BiometricEvaluation::Feature::AN2K11EFS::ExtendedFeatureSet Class Reference

A class to represent the Extended **Feature** (p. 111) Set optionally present in an ANSI/NIST Type-9 record.

```
#include <be_feature_an2k11efs.h>
```

Public Member Functions

- **ExtendedFeatureSet** (const std::string &filename, int recordNumber)
Construct an AN2K11 EFS object from file data.
- **ExtendedFeatureSet** (**Memory::uint8Array** &buf, int recordNumber)
Construct an AN2K11 EFS object from data contained in a memory buffer.
- **ImageInfo** **getImageInfo** ()
*Obtain the structure containing information about the image and Extended **Feature** (p. 111) Set.*
- BiometricEvaluation::Feature::AN2K11EFS::MinutiaPointSet **getMPS** ()
Obtain the minutiae point set.

- **BiometricEvaluation::Feature::AN2K11EFS::MinutiaeRidgeCountInfo** **getMRCI** ()
Obtain all the information relating to minutiae ridge count information.
- **BiometricEvaluation::Feature::AN2K11EFS::CorePointSet** **getCPS** ()
Obtain the core point set.
- **BiometricEvaluation::Feature::AN2K11EFS::DeltaPointSet** **getDPS** ()
Obtain the delta point set.
- **BiometricEvaluation::Feature::AN2K11EFS::NoFeaturesPresent** **getNFP** ()

G.49.1 Detailed Description

A class to represent the Extended **Feature** (p. 111) Set optionally present in an ANSI/NIST Type-9 record.

Each minutiae point, ridge count item, core, and delta is represented in the native ANSI/NIST format. Conforms with ANSI/NIST-ITL-2011: Update 2015 standard.

G.49.2 Constructor & Destructor Documentation

G.49.2.1 ExtendedFeatureSet() [1/2]

```
BiometricEvaluation::Feature::AN2K11EFS::ExtendedFeatureSet::ExtendedFeatureSet (
    const std::string & filename,
    int recordNumber )
```

Construct an AN2K11 EFS object from file data.

The file contains a complete ANSI/NIST record, and an object of this class represents a single Type-9 extended feature set structure.

Parameters

in	<i>filename</i>	The name of the file containing the complete ANSI/NIST record.
in	<i>recordNumber</i>	Which fingerprint minutiae record to read from the complete AN2K record.

Exceptions

Error::ObjectDoesNotExist (p. 457)	The named file does not exist.
Error::StrategyError (p. 567)	An error occurred when opening or reading from the file.
Error::DataError (p. 295)	An error occurred reading the AN2K record, or there is no fingerprint minutiae record for the requested number.

G.49.2.2 ExtendedFeatureSet() [2/2]

```
BiometricEvaluation::Feature::AN2K11EFS::ExtendedFeatureSet::ExtendedFeatureSet (
    Memory::uint8Array & buf,
    int recordNumber )
```

Construct an AN2K11 EFS object from data contained in a memory buffer.

The buffer contains a complete ANSI/NIST record, and an object of this class represents a single Type-9 extended feature set structure.

Parameters

in	<i>buf</i>	The memory buffer containing the complete ANSI/NIST record.
in	<i>recordNumber</i>	Which fingerprint minutiae record to read from the complete AN2K record.

Exceptions

Error::DataError (p. 295)	An error occurred reading the AN2K record, or there is no fingerprint minutiae record for the requested number.
----------------------------------	---

G.49.3 Member Function Documentation

G.49.3.1 getCPS()

BiometricEvaluation::Feature::AN2K11EFS::CorePointSet BiometricEvaluation::Feature::AN2K11EFS::ExtendedFeatureSet::getCPS ()

Obtain the core point set.

The set may be empty as this Type-9 field is optional.

Returns

The set of core points.

G.49.3.2 getDPS()

BiometricEvaluation::Feature::AN2K11EFS::DeltaPointSet BiometricEvaluation::Feature::AN2K11EFS::ExtendedFeatureSet::getDPS ()

Obtain the delta point set.

The set may be empty as this Type-9 field is optional.

Returns

The set of delta points.

G.49.3.3 getImageInfo()

ImageInfo BiometricEvaluation::Feature::AN2K11EFS::ExtendedFeatureSet::getImageInfo ()

Obtain the structure containing information about the image and Extended **Feature** (p. 111) Set.

Returns

The information about the image.

G.49.3.4 getMPS()

`BiometricEvaluation::Feature::AN2K11EFS::MinutiaPointSet` `BiometricEvaluation::Feature::AN2K11EFS::ExtendedFeatureSet::getMPS ()`

Obtain the minutiae point set.

The set may be empty as this Type-9 field is optional.

Returns

The set of minutia points.

G.49.3.5 getMRCI()

`BiometricEvaluation::Feature::AN2K11EFS::MinutiaeRidgeCountInfo` `BiometricEvaluation::Feature::AN2K11EFS::ExtendedFeatureSet::getMRCI ()`

Obtain all the information relating to minutiae ridge count information.

Some of the information may not be present for the optional fields in the AN2k11 extended feature set.

Returns

The minutiae ridge count information structure.

G.49.3.6 getNFP()

`BiometricEvaluation::Feature::AN2K11EFS::NoFeaturesPresent` `BiometricEvaluation::Feature::AN2K11EFS::ExtendedFeatureSet::getNFP ()`

Obtain the No Features Present indicators.

Returns

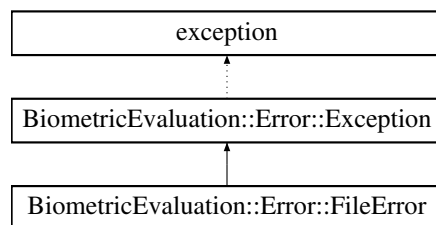
The flags for No Features Present.

G.50 BiometricEvaluation::Error::FileError Class Reference

File error when opening, reading, writing, etc.

```
#include <be_error_exception.h>
```

Inheritance diagram for `BiometricEvaluation::Error::FileError`:



Public Member Functions

- **FileError** ()
- **FileError** (const std::string &info)

G.50.1 Detailed Description

File error when opening, reading, writing, etc.

G.50.2 Constructor & Destructor Documentation

G.50.2.1 FileError() [1/2]

```
BiometricEvaluation::Error::FileError::FileError ( )
```

Construct a **FileError** (p. 316) object with the default information string.

G.50.2.2 FileError() [2/2]

```
BiometricEvaluation::Error::FileError::FileError (
    const std::string & info )
```

Construct a **FileError** (p. 316) object with an information string appended to the default information string.

G.51 BiometricEvaluation::IO::FileLogCabinet Class Reference

```
#include <be_io_filelogcabinet.h>
```

Public Member Functions

- **FileLogCabinet** (const std::string &pathname, const std::string &description)
- **FileLogCabinet** (const std::string &pathname)
- std::shared_ptr< **FileLogsSheet** > **newLogsSheet** (const std::string &name, const std::string &description)
- std::string **getPathname** ()
- std::string **getDescription** ()
- unsigned int **getCount** ()

G.51.1 Detailed Description

A class to represent a collection of log sheets.

G.51.2 Constructor & Destructor Documentation

G.51.2.1 FileLogCabinet() [1/2]

```
BiometricEvaluation::IO::FileLogCabinet::FileLogCabinet (
    const std::string & pathname,
    const std::string & description )
```

Create a new **FileLogCabinet** (p. 317) in the file system.

Parameters

in	<i>pathname</i>	The pathname where the FileLogCabinet (p. 317) is to be created.
in	<i>description</i>	The text used to describe the cabinet.

Exceptions

Error::ObjectExists (p. 458)	The cabinet was previously created.
Error::StrategyError (p. 567)	An error occurred when using the underlying file system.

G.51.2.2 FileLogCabinet() [2/2]

```
BiometricEvaluation::IO::FileLogCabinet::FileLogCabinet (
    const std::string & pathname )
```

Open an existing **FileLogCabinet** (p. [317](#)).

Parameters

in	<i>pathname</i>	The pathname where the FileLogCabinet (p. 317) is located.
----	-----------------	--

Exceptions

Error::ObjectDoesNotExist (p. 457)	The cabinet does not exist in the file system.
Error::StrategyError (p. 567)	An error occurred when using the underlying file system.

G.51.3 Member Function Documentation**G.51.3.1 getCount()**

```
unsigned int BiometricEvaluation::IO::FileLogCabinet::getCount ( )
```

Obtain the number of items in the **FileLogCabinet** (p. [317](#)).

@ returns The number of logsheets manages by the cabinet.

G.51.3.2 getDescription()

```
std::string BiometricEvaluation::IO::FileLogCabinet::getDescription ( )
```

Obtain the description of the **FileLogCabinet** (p. [317](#)).

@ returns The description of the **FileLogCabinet** (p. [317](#)).

G.51.3.3 getPathname()

```
std::string BiometricEvaluation::IO::FileLogCabinet::getPathname ( )
```

Obtain the pathname of the **FileLogCabinet** (p. [317](#)).

@ returns The pathname of the **FileLogCabinet** (p. [317](#)).

G.51.3.4 newLogsheet()

```
std::shared_ptr< FileLogsheet> BiometricEvaluation::IO::FileLogCabinet::newLogsheet (
    const std::string & name,
    const std::string & description )
```


Create a new **FileLogsheet** (p. 319) within the cabinet.

Parameters

in	<i>name</i>	The name of the FileLogsheet (p. 319) to be created. This can not be a path name.
in	<i>description</i>	The text used to describe the sheet. This text is written into the log file prior to any entries.

Returns

An object pointer to the new log sheet.

Exceptions

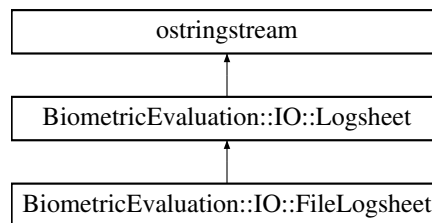
Error::ObjectExists (p. 458)	The sheet was previously created.
Error::StrategyError (p. 567)	An error occurred when using the underlying file system.

G.52 BiometricEvaluation::IO::FileLogsheet Class Reference

A class to represent a single logging mechanism with a file as the backing store.

```
#include <be_io_filelogsheet.h>
```

Inheritance diagram for BiometricEvaluation::IO::FileLogsheet:



Public Member Functions

- **FileLogsheet** (const std::string &url, const std::string &description)
Create a new log sheet.
- **FileLogsheet** (const std::string &url)
Open an existing log sheet for appending.
- **~FileLogsheet** ()
- std::string **sequence** (bool allEntries=false, bool **trim**=true, int32_t cursor= **BE_FILELOGSHEET_SEQ_NEXT**)
*Sequence through a **FileLogsheet** (p. 319), returning one entry per invocation.*
- void **write** (const std::string &entry)
Write a string as an entry to the backing store.
- void **writeComment** (const std::string &entry)
Write a string as a comment to the backing store.

- void **writeDebug** (const std::string &entry)
Write a string as a debug entry to the backing store.
- void **sync** ()
Synchronize any buffered data to the underlying backing store.

Static Public Member Functions

- static void **mergeLogsheets** (std::vector< std::shared_ptr< **FileLogsheet** >> &logsheets)
*Merge multiple FileLogsheets into a single **FileLogsheet** (p. 319).*
- static std::string **trim** (const std::string &entry)
*Trim delimiters from **FileLogsheet** (p. 319) entries.*

Static Public Attributes

- static const int32_t **BE_FILELOGSHEET_SEQ_START** = 1
- static const int32_t **BE_FILELOGSHEET_SEQ_NEXT** = 2

Protected Member Functions

- **FileLogsheet** (const **FileLogsheet** &)
- **FileLogsheet** & **operator=** (const **FileLogsheet** &)
- void **updateCursor** ()
Update the cursor position of the sequence file.

Protected Attributes

- std::unique_ptr< std::fstream > **_theLogFile**
- std::shared_ptr< std::fstream > **_sequenceFile**
- streamoff **_cursor**

Additional Inherited Members

G.52.1 Detailed Description

A class to represent a single logging mechanism with a file as the backing store.

A **FileLogsheet** (p. 319) object can be constructed and passed back to the client by the LogCabinet object. All sheets created in this manner are placed in a common area maintained by the cabinet.

G.52.2 Constructor & Destructor Documentation

G.52.2.1 FileLogsheet() [1/3]

```
BiometricEvaluation::IO::FileLogsheet::FileLogsheet (
    const std::string & url,
    const std::string & description )
```

Create a new log sheet.

the log sheet is named by the uniform resource locator, usually starting with 'file://'. However, relative and absolute path names are also accepted for backward compatibility.

Parameters

in	<i>url</i>	The Uniform Resource Locator of the FileLogsheet (p. 319) to be created.
in	<i>description</i>	The text used to describe the sheet. This text is written into the log file prior to any entries.

Exceptions

Error::ParameterError (p. 474)	The URL is malformed.
Error::ObjectExists (p. 458)	The sheet was previously created.
Error::StrategyError (p. 567)	An error occurred when using the underlying file system, or name or parentDir is malformed.

G.52.2.2 FileLogsheet() [2/3]

```
BiometricEvaluation::IO::FileLogsheet::FileLogsheet (
    const std::string & url )
```

Open an existing log sheet for appending.

On open, the current entry counter is set to the last entry number plus one.

Note

Opening a large **FileLogsheet** (p. 319) may be a costly operation.

Parameters

in	<i>url</i>	The Uniform Resource Locator of the FileLogsheet (p. 319) to be opened.
----	------------	--

Exceptions

Error::ParameterError (p. 474)	The URL is malformed.
Error::ObjectDoesNotExist (p. 457)	The sheet does not exist.
Error::StrategyError (p. 567)	An error occurred when using the underlying file system, or name or parentDir is malformed.

G.52.2.3 ~FileLogsheet()

```
BiometricEvaluation::IO::FileLogsheet::~~FileLogsheet ( )
```

Destructor

G.52.2.4 FileLogsheet() [3/3]

```
BiometricEvaluation::IO::FileLogsheet::FileLogsheet (
    const FileLogsheet & ) [protected]
```

Prevent copying of **FileLogsheet** (p. 319) objects

G.52.3 Member Function Documentation

G.52.3.1 mergeLogsheets()

```
static void BiometricEvaluation::IO::FileLogsheet::mergeLogsheets (
    std::vector< std::shared_ptr< FileLogsheet >> & logsheets ) [static]
```

Merge multiple FileLogsheets into a single **FileLogsheet** (p. 319).

Logsheet (p. 420) 2 - n will be appended to **Logsheet** (p. 420) 1.

Parameters

<i>logSheets</i>	Logsheet (p. 420) to merge.
------------------	------------------------------------

Exceptions

<i>Error::FileError</i> (p. 316)	Error (p. 108) during log sequence.
<i>Error::StrategyError</i> (p. 567)	Error (p. 108) during log sequence.

G.52.3.2 operator=()

```
FileLogsheet& BiometricEvaluation::IO::FileLogsheet::operator= (
    const FileLogsheet & ) [protected]
```

Prevent copying of **FileLogsheet** (p. 319) objects

G.52.3.3 sequence()

```
std::string BiometricEvaluation::IO::FileLogsheet::sequence (
    bool allEntries = false,
    bool trim = true,
    int32_t cursor = BE_FILELOGSHEET_SEQ_NEXT )
```

Sequence through a **FileLogsheet** (p. 319), returning one entry per invocation.

Parameters

<i>allEntries</i>	Include debug and comment entries when sequencing
<i>trim</i>	Whether or not to include entry delimiters.
<i>cursor</i>	The location within the sequence to return.

Returns

The contents of the sequenced entry, as was originally given to **write()** (p. 323).

Exceptions

<i>Error::FileError</i> (p. 316), <i>Error</i> (p. 108)	occured while performing file IO (p. 126).
<i>Error::ObjectDoesNotExist</i> (p. 457)	The FileLogsheet (p. 319) cannot be found on disk.
<i>Error::StrategyError</i> (p. 567)	Invalid cursor position or the contents of the FileLogsheet (p. 319) is malformed.

G.52.3.4 sync()

```
void BiometricEvaluation::IO::FileLogsheet::sync ( ) [virtual]
```

Synchronize any buffered data to the underlying backing store.

This syncing is dependent on the behavior of the underlying storage mechanism.

Exceptions

<i>Error::StrategyError</i> (p. 567)	An error occurred when using the underlying backing store.
--------------------------------------	--

Reimplemented from **BiometricEvaluation::IO::Logsheet** (p. 427).

G.52.3.5 trim()

```
static std::string BiometricEvaluation::IO::FileLogsheet::trim (
    const std::string & entry ) [static]
```

Trim delimiters from **FileLogsheet** (p. 319) entries.

Works for comments and numbered entries.

Parameters

in	<i>entry</i>	The entry to trim.
----	--------------	--------------------

Returns

Delimiter-less entry.

G.52.3.6 updateCursor()

```
void BiometricEvaluation::IO::FileLogsheet::updateCursor ( ) [protected]
```

Update the cursor position of the sequence file.

Exceptions

<i>Error::FileError</i> (p. 316)	Error (p. 108) getting file position from sequence file.
----------------------------------	---

G.52.3.7 write()

```
void BiometricEvaluation::IO::FileLogsheet::write (
    const std::string & entry ) [virtual]
```

Write a string as an entry to the backing store.

This does not affect the current log entry buffer, but does increment the entry number.

Parameters

in	<i>entry</i>	The text of the log entry.
----	--------------	----------------------------

Exceptions

Error::StrategyError (p. 567)	An error occurred when using the underlying backing store.
---	--

Reimplemented from **BiometricEvaluation::IO::Logsheet** (p. [427](#)).

G.52.3.8 writeComment()

```
void BiometricEvaluation::IO::FileLogsheet::writeComment (
    const std::string & entry ) [virtual]
```

Write a string as a comment to the backing store.

This does not affect the current log entry buffer, and does not increment the entry number. A comment line is prefixed with CommentDelimiter followed by a space by this method.

Parameters

in	<i>entry</i>	The text of the comment.
----	--------------	--------------------------

Exceptions

Error::StrategyError (p. 567)	An error occurred when using the underlying backing store.
---	--

Reimplemented from **BiometricEvaluation::IO::Logsheet** (p. [428](#)).

G.52.3.9 writeDebug()

```
void BiometricEvaluation::IO::FileLogsheet::writeDebug (
    const std::string & entry ) [virtual]
```

Write a string as a debug entry to the backing store.

This does not affect the current log entry buffer, and does not increment the entry number. A debug line is prefixed with DebugDelimiter followed by a space.

Parameters

in	<i>entry</i>	The text of the debug message.
----	--------------	--------------------------------

Exceptions

Error::StrategyError (p. 567)	An error occurred when logging.
---	---------------------------------

Reimplemented from **BiometricEvaluation::IO::Logsheet** (p. [428](#)).

G.52.4 Member Data Documentation

G.52.4.1 _cursor

`streamoff BiometricEvaluation::IO::FileLogsheet::_cursor` [protected]

Position of the sequencer, relative to SOF

G.52.4.2 _sequenceFile

`std::shared_ptr<std::fstream> BiometricEvaluation::IO::FileLogsheet::_sequenceFile` [protected]

Stream used for sequencing

G.52.4.3 _theLogFile

`std::unique_ptr<std::fstream> BiometricEvaluation::IO::FileLogsheet::_theLogFile` [protected]

Stream used for writing the log file

G.52.4.4 BE_FILELOGSHEET_SEQ_NEXT

`const int32_t BiometricEvaluation::IO::FileLogsheet::BE_FILELOGSHEET_SEQ_NEXT = 2` [static]

Sequence from current position

G.52.4.5 BE_FILELOGSHEET_SEQ_START

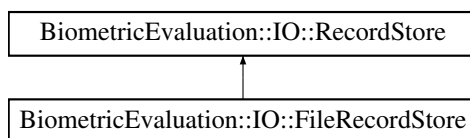
`const int32_t BiometricEvaluation::IO::FileLogsheet::BE_FILELOGSHEET_SEQ_START = 1` [static]

Sequence from beginning

G.53 BiometricEvaluation::IO::FileRecordStore Class Reference

`#include <be_io_filerecstore.h>`

Inheritance diagram for BiometricEvaluation::IO::FileRecordStore:



Public Member Functions

- **FileRecordStore** (const std::string &pathname, const std::string &description)
- **FileRecordStore** (const std::string &name, **IO::Mode** mode= **IO::Mode::ReadOnly**)
- void **insert** (const std::string &key, const void *const data, const uint64_t size) override
- void **remove** (const std::string &key) override
- **Memory::uint8Array read** (const std::string &key) const override
Read a complete record from a store.
- void **replace** (const std::string &key, const void *const data, const uint64_t size) override final
- uint64_t **length** (const std::string &key) const override
- void **flush** (const std::string &key) const override
- **RecordStore::Record sequence** (int cursor= **BE_RECSTORE_SEQ_NEXT**) override
*Sequence through a **RecordStore** (p. 504), returning the key/data pairs.*
- std::string **sequenceKey** (int cursor= **BE_RECSTORE_SEQ_NEXT**) override
*Sequence through a **RecordStore** (p. 504), returning the key.*
- void **setCursorAtKey** (const std::string &key) override
- void **move** (const std::string &pathname) override
*Move the **RecordStore** (p. 504).*
- uint64_t **getSpaceUsed** () const override
Obtain real storage utilization.
- void **sync** () const override
- unsigned int **getCount** () const override
- std::string **getPathname** () const override
- std::string **getDescription** () const override
- void **changeDescription** (const std::string &description) override
- **FileRecordStore** (const **FileRecordStore** &)=delete
- **FileRecordStore & operator=** (const **FileRecordStore** &)=delete

Additional Inherited Members

G.53.1 Detailed Description

Class to represent the record store data storage mechanism implemented as files for each record.

Note

For the methods that take a key parameter, **Error::StrategyError** (p. 567) will be thrown if the key string is not compliant. A **FileRecordStore** (p. 325) has the additional requirement that a key name may not contain path delimiter characters ('/' and '\'), or begin with whitespace.

G.53.2 Constructor & Destructor Documentation

G.53.2.1 FileRecordStore() [1/2]

```
BiometricEvaluation::IO::FileRecordStore::FileRecordStore (
    const std::string & pathname,
    const std::string & description )
```

Create a new **FileRecordStore** (p. 325), read/write mode.

Parameters

in	<i>pathname</i>	The directory where the store is to be created.
in	<i>description</i>	The store's description.

Exceptions

Error::ObjectExists (p. 458)	The store already exists.
Error::StrategyError (p. 567)	An error occurred when accessing the underlying file system.

G.53.2.2 FileRecordStore() [2/2]

```
BiometricEvaluation::IO::FileRecordStore::FileRecordStore (
    const std::string & name,
    IO::Mode mode = IO::Mode::ReadOnly )
    Open an existing FileRecordStore (p. 325).
```

Parameters

in	<i>name</i>	The path name of the store.
in	<i>mode</i>	Open mode, read-only or read-write.

Exceptions

Error::ObjectDoesNotExist (p. 457)	The store does not exist.
Error::StrategyError (p. 567)	An error occurred when accessing the underlying file system.

G.53.3 Member Function Documentation

G.53.3.1 changeDescription()

```
void BiometricEvaluation::IO::FileRecordStore::changeDescription (
    const std::string & description ) [override], [virtual]
    Change the description of the RecordStore (p. 504).
```

Parameters

in	<i>description</i>	The new description.
----	--------------------	----------------------

Exceptions

Error::StrategyError (p. 567)	An error occurred when using the underlying storage system.
---	---

Implements **BiometricEvaluation::IO::RecordStore** (p. 506).

G.53.3.2 flush()

```
void BiometricEvaluation::IO::FileRecordStore::flush (
    const std::string & key ) const [override], [virtual]
```

Commit the record's data to storage.

Parameters

in	key	The key of the record to be flushed.
----	-----	--------------------------------------

Exceptions

Error::ObjectDoesNotExist (p. 457)	A record for the key does not exist.
Error::StrategyError (p. 567)	An error occurred when using the underlying storage system.

Implements **BiometricEvaluation::IO::RecordStore** (p. 508).

G.53.3.3 getCount()

```
unsigned int BiometricEvaluation::IO::FileRecordStore::getCount ( ) const [override], [virtual]
```

Obtain the number of items in the **RecordStore** (p. 504).

Returns

The number of items in the **RecordStore** (p. 504).

Implements **BiometricEvaluation::IO::RecordStore** (p. 508).

G.53.3.4 getDescription()

```
std::string BiometricEvaluation::IO::FileRecordStore::getDescription ( ) const [override], [virtual]
```

Obtain a textual description of the **RecordStore** (p. 504).

Returns

The **RecordStore** (p. 504)'s description.

Implements **BiometricEvaluation::IO::RecordStore** (p. 508).

G.53.3.5 getPathname()

```
std::string BiometricEvaluation::IO::FileRecordStore::getPathname ( ) const [override], [virtual]
```

Return the path name of the **RecordStore** (p. 504).

Returns

Where in the file system the **RecordStore** (p. 504) is located.

Implements **BiometricEvaluation::IO::RecordStore** (p. 509).

G.53.3.6 getSpaceUsed()

```
uint64_t BiometricEvaluation::IO::FileRecordStore::getSpaceUsed ( ) const [override], [virtual]
```

Obtain real storage utilization.

The amount of disk space used, for example. This is the actual space allocated by the underlying storage mechanism, in bytes.

Returns

The amount of backing storage used by the **RecordStore** (p. 504).

Exceptions

Error::StrategyError (p. 567)	An error occurred when using the underlying storage system.
--------------------------------------	---

Implements **BiometricEvaluation::IO::RecordStore** (p. 509).

G.53.3.7 insert()

```
void BiometricEvaluation::IO::FileRecordStore::insert (
    const std::string & key,
    const void *const data,
    const uint64_t size ) [override], [virtual]
```

Insert a record into the store.

Parameters

in	<i>key</i>	The key of the record to be inserted.
in	<i>data</i>	The data for the record.
in	<i>size</i>	The size of the record, in bytes.

Exceptions

Error::ObjectExists (p. 458)	A record with the given key is already present.
Error::StrategyError (p. 567)	The RecordStore (p. 504) is opened read-only, or an error occurred when using the underlying storage system.

Implements **BiometricEvaluation::IO::RecordStore** (p. 510).

G.53.3.8 length()

```
uint64_t BiometricEvaluation::IO::FileRecordStore::length (
    const std::string & key ) const [override], [virtual]
```

Return the length of a record.

Parameters

in	<i>key</i>	The key of the record.
----	------------	------------------------

Returns

The record length.

Exceptions

Error::ObjectDoesNotExist (p. 457)	A record for the key does not exist.
Error::StrategyError (p. 567)	An error occurred when using the underlying storage system.

Implements **BiometricEvaluation::IO::RecordStore** (p. 510).

G.53.3.9 move()

```
void BiometricEvaluation::IO::FileRecordStore::move (
    const std::string & pathname ) [override], [virtual]
```

Move the **RecordStore** (p. 504).

The **RecordStore** (p. 504) can be moved to a new path in the file system.

Parameters

in	<i>pathname</i>	The new path of the RecordStore (p. 504).
----	-----------------	--

Exceptions

Error::StrategyError (p. 567)	An error occurred when using the underlying storage system.
--------------------------------------	---

Implements **BiometricEvaluation::IO::RecordStore** (p. 511).

G.53.3.10 read()

```
Memory::uint8Array BiometricEvaluation::IO::FileRecordStore::read (
    const std::string & key ) const [override], [virtual]
```

Read a complete record from a store.

The AutoArray will be resized to match the size of the data.

Parameters

in	<i>key</i>	The key of the record to be read.
----	------------	-----------------------------------

Returns

The record associated with the key.

Exceptions

Error::ObjectDoesNotExist (p. 457)	A record for the key does not exist.
Error::StrategyError (p. 567)	An error occurred when using the underlying storage system.

Implements **BiometricEvaluation::IO::RecordStore** (p. 512).

G.53.3.11 remove()

```
void BiometricEvaluation::IO::FileRecordStore::remove (
    const std::string & key ) [override], [virtual]
```

Remove a record from the store.

Parameters

in	key	The key of the record to be removed.
----	-----	--------------------------------------

Exceptions

Error::ObjectDoesNotExist (p. 457)	A record for the key does not exist.
Error::StrategyError (p. 567)	An error occurred when using the underlying storage system.

Implements **BiometricEvaluation::IO::RecordStore** (p. 513).

G.53.3.12 replace()

```
void BiometricEvaluation::IO::FileRecordStore::replace (
    const std::string & key,
    const void *const data,
    const uint64_t size ) [final], [override], [virtual]
```

Replace a complete record in a **RecordStore** (p. 504).

Parameters

in	key	The key of the record to be replaced.
in	data	The data for the record.
in	size	The size of the record, in bytes.

Exceptions

Error::ObjectDoesNotExist (p. 457)	A record for the key does not exist.
Error::StrategyError (p. 567)	The RecordStore (p. 504) is opened read-only, or an error occurred when using the underlying storage system.

Reimplemented from **BiometricEvaluation::IO::RecordStore** (p. 514).

G.53.3.13 sequence()

```
RecordStore::Record BiometricEvaluation::IO::FileRecordStore::sequence (
    int cursor = BE_RECSTORE_SEQ_NEXT ) [override], [virtual]
```

Sequence through a **RecordStore** (p. 504), returning the key/data pairs.

Sequencing means to start at some point in the store and return the record, then repeatedly calling the function to return the next record. The starting point is typically the first record, and is set to that when the **RecordStore** (p. 504) object is created. The starting point can be reset by calling this method with the cursor parameter set to BE_RECSTORE_SEQ_START.

Parameters

in	<i>cursor</i>	The location within the sequence of the key/data pair to return.
----	---------------	--

Returns

The record that is currently in sequence.

Exceptions

Error::ObjectDoesNotExist (p. 457)	End of sequencing.
Error::StrategyError (p. 567)	An error occurred when using the underlying storage system.

Implements **BiometricEvaluation::IO::RecordStore** (p. 515).

G.53.3.14 sequenceKey()

```
std::string BiometricEvaluation::IO::FileRecordStore::sequenceKey (
    int cursor = BE_RECSTORE_SEQ_NEXT ) [override], [virtual]
```

Sequence through a **RecordStore** (p. 504), returning the key.

Sequencing means to start at some point in the store and return the key, then repeatedly calling the function to return the next key. The starting point is typically the first record, and is set to that when the **RecordStore** (p. 504) object is created. The starting point can be reset by calling this method with the cursor parameter set to BE_RECSTORE_SEQ_START.

Parameters

in	<i>cursor</i>	The location within the sequence of the key/data pair to return.
----	---------------	--

Returns

The key of the currently sequenced record.

Exceptions

Error::ObjectDoesNotExist (p. 457)	End of sequencing.
Error::StrategyError (p. 567)	An error occurred when using the underlying storage system.

Implements **BiometricEvaluation::IO::RecordStore** (p. 515).

G.53.3.15 setCursorAtKey()

```
void BiometricEvaluation::IO::FileRecordStore::setCursorAtKey (
    const std::string & key ) [override], [virtual]
```

Set the sequence cursor to an arbitrary position within the **RecordStore** (p. 504), starting at key. Key will be the first record returned from the next call to **sequence()** (p. 331).

Parameters

in	key	The key of the record which will be returned by the first subsequent call to sequence() (p. 331).
----	-----	--

Exceptions

Error::ObjectDoesNotExist (p. 457)	A record for the key does not exist.
Error::StrategyError (p. 567)	An error occurred when using the underlying storage system.

Implements **BiometricEvaluation::IO::RecordStore** (p. 516).

G.53.3.16 sync()

```
void BiometricEvaluation::IO::FileRecordStore::sync ( ) const [override], [virtual]
    Synchronize the entire record store to persistent storage.
```

Exceptions

Error::StrategyError (p. 567)	An error occurred when using the underlying storage system.
--------------------------------------	---

Implements **BiometricEvaluation::IO::RecordStore** (p. 516).

G.54 BiometricEvaluation::Feature::AN2K7Minutiae::FingerprintReading↵
System Struct Reference

Representation of information about a fingerprint reader system.
#include <be_feature_an2k7minutiae.h>

Public Attributes

- std::string name
- EncodingMethod method
- std::string equipment

G.54.1 Detailed Description

Representation of information about a fingerprint reader system.

G.54.2 Member Data Documentation

G.54.2.1 equipment

`std::string BiometricEvaluation::Feature::AN2K7Minutiae::FingerprintReadingSystem::equipment`
Optional ID for equipment used in system

G.54.2.2 method

EncodingMethod `BiometricEvaluation::Feature::AN2K7Minutiae::FingerprintReadingSystem::method`
Method used to encoded minutiae

G.54.2.3 name

`std::string BiometricEvaluation::Feature::AN2K7Minutiae::FingerprintReadingSystem::name`
Name for system that encoded minutiae

G.55 BiometricEvaluation::Finger::AN2KViewCapture::FingerSegmentPosition Struct Reference

Locations of an individual finger segment in a slap.

```
#include <be_finger_an2kview_capture.h>
```

Public Member Functions

- **FingerSegmentPosition** (const **Finger::Position** **fingerPosition**, const **Image::CoordinateSet** **coordinates**)

Create an *FingerSegmentPosition* (p. 334) struct.

Public Attributes

- **Finger::Position** **fingerPosition**
- **Image::CoordinateSet** **coordinates**

G.55.1 Detailed Description

Locations of an individual finger segment in a slap.

G.55.2 Constructor & Destructor Documentation

G.55.2.1 FingerSegmentPosition()

```
BiometricEvaluation::Finger::AN2KViewCapture::FingerSegmentPosition::FingerSegmentPosition (
    const Finger::Position fingerPosition,
    const Image::CoordinateSet coordinates )
```

Create an **FingerSegmentPosition** (p. 334) struct.

Parameters

<i>fingerPosition</i>	Finger (p. 115) depicted in this segment.
<i>coordinates</i>	Collection of coordinates that compose the segment bonding polygon.

G.55.3 Member Data Documentation

G.55.3.1 coordinates

Image::CoordinateSet BiometricEvaluation::Finger::AN2KViewCapture::FingerSegmentPosition::coordinates
Points composing the segmented polygon

G.55.3.2 fingerPosition

Finger::Position BiometricEvaluation::Finger::AN2KViewCapture::FingerSegmentPosition::finger↔
Position

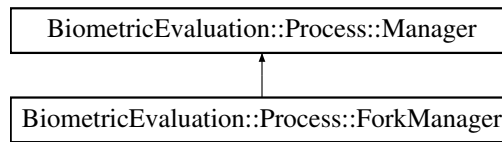
Finger (p. 115) depicted in this segment

G.56 BiometricEvaluation::Process::ForkManager Class Reference

Manager (p. 430) implementation that starts Workers by calling fork(2).

```
#include <be_process_forkmanager.h>
```

Inheritance diagram for BiometricEvaluation::Process::ForkManager:



Public Member Functions

- **ForkManager** ()
- std::shared_ptr< **WorkerController** > **addWorker** (std::shared_ptr< **Worker** > worker)
*Adds a **Worker** (p. 595) to be managed by this **Manager** (p. 430).*
- void **startWorkers** (bool wait=true, bool communicate=false)
*Begin **Worker** (p. 595)'s work.*
- void **startWorker** (std::shared_ptr< **WorkerController** > worker, bool wait=true, bool communicate=false)
Start a worker.
- void **stopWorker** (std::shared_ptr< **WorkerController** > workerController)
*Ask **Worker** (p. 595) to exit.*
- void **broadcastSignal** (int signo)
Send a POSIX signal to all workers.
- bool **responsibleFor** (const pid_t pid) const
*Obtain whether or not this **ForkManager** (p. 335) is responsible for a particular PID.*
- void **setNotWorking** (const pid_t pid)
Set Status.isWorking for PID to false.
- void **markAllFinished** ()
*Call setNotWorking() (p. 339) for all PIDs known to this **ForkManager** (p. 335).*
- bool **getIsWorkingStatus** (const pid_t pid) const

- Get Status.isWorking for PID.*
- void **waitForWorkerExit** ()
Block until all Workers have exited.
- **~ForkManager** ()
ForkManager (p. 335) destructor.
- void **setExitCallback** (void(*exitCallback)(std::shared_ptr< **ForkWorkerController** > worker, int stat_loc))
Call a function in your program when a child exits.
- void **setExitStatus** (const pid_t pid, const int32_t waitStatus)
*Set the exit status in the **WorkerController** (p. 601) for given process ID.*

Static Public Member Functions

- static void **defaultExitCallback** (std::shared_ptr< **ForkWorkerController** > worker, int status)
A default exit callback function.

Static Public Attributes

- static std::list< **ForkManager** * > **FORKMANAGERS**
List of all instantiated ForkManagers.

Additional Inherited Members

G.56.1 Detailed Description

Manager (p. 430) implementation that starts Workers by calling fork(2).

G.56.2 Constructor & Destructor Documentation

G.56.2.1 ForkManager()

BiometricEvaluation::Process::ForkManager::ForkManager ()
ForkManager (p. 335) constructor.

G.56.3 Member Function Documentation

G.56.3.1 addWorker()

std::shared_ptr< **WorkerController**> BiometricEvaluation::Process::ForkManager::addWorker (
std::shared_ptr< **Worker** > worker) [virtual]
Adds a **Worker** (p. 595) to be managed by this **Manager** (p. 430).

Parameters

<i>worker</i>	A Worker (p. 595) instance to run.
---------------	---

Returns

shared_ptr to worker.

Implements **BiometricEvaluation::Process::Manager** (p. 431).

G.56.3.2 broadcastSignal()

```
void BiometricEvaluation::Process::ForkManager::broadcastSignal (
    int signo )
```

Send a POSIX signal to all workers.

Parameters

in	<i>signo</i>	The signal to send.
----	--------------	---------------------

G.56.3.3 defaultExitCallback()

```
static void BiometricEvaluation::Process::ForkManager::defaultExitCallback (
    std::shared_ptr< ForkWorkerController > worker,
    int status ) [static]
```

A default exit callback function.
Writes to stdout in the form: PID #: Exited .

Parameters

<i>worker</i>	The ForkWorkerController (p. 341) object that exited.
<i>status</i>	The status of the Worker (p. 595) that exited (from wait(2)).

G.56.3.4 getIsWorkingStatus()

```
bool BiometricEvaluation::Process::ForkManager::getIsWorkingStatus (
    const pid_t pid ) const
```

Get Status.isWorking for PID.

Parameters

in	<i>pid</i>	PID whose inWorking flag should be queried
----	------------	--

Exceptions

Error::ObjectDoesNotExist (p. 457)	PID not under this manager's control.
---	---------------------------------------

G.56.3.5 responsibleFor()

```
bool BiometricEvaluation::Process::ForkManager::responsibleFor (
    const pid_t pid ) const
```

Obtain whether or not this **ForkManager** (p. 335) is responsible for a particular PID.

Parameters

in	<i>pid</i>	PID in question
----	------------	-----------------

Returns

true if this **ForkManager** (p. 335) spawned pid, false otherwise.

G.56.3.6 setExitCallback()

```
void BiometricEvaluation::Process::ForkManager::setExitCallback (
    void(*) (std::shared_ptr< ForkWorkerController > worker, int stat_loc) exitCallback
)
```

Call a function in your program when a child exits.

Parameters

<i>exitCallback</i>	Function pointer to a method that takes a <code>shared_ptr</code> to a ForkWorkerController (p. 341) and the integer status information.
---------------------	---

Note

The exit callback will not have any effect if the **Manager** (p. 430) is not set to wait for Workers.

G.56.3.7 setExitStatus()

```
void BiometricEvaluation::Process::ForkManager::setExitStatus (
    const pid_t pid,
    const int32_t waitStatus )
```

Set the exit status in the **WorkerController** (p. 601) for given process ID.

Parameters

in	<i>pid</i>	PID whose exit status should be set.
in	<i>status</i>	Status, as returned from <code>wait(2)</code> .

Exceptions

Error::ObjectDoesNotExist (p. 457)	PID not under this manager's control.
---	---------------------------------------

Note

Exit status is only set if process exited cleanly.

G.56.3.8 setNotWorking()

```
void BiometricEvaluation::Process::ForkManager::setNotWorking (
    const pid_t pid )
```

Set Status.isWorking for PID to false.

Parameters

in	<i>pid</i>	PID whose inWorking flag should be set to false
----	------------	---

Exceptions

Error::ObjectDoesNotExist (p. 457)	PID not under this manager's control.
---	---------------------------------------

G.56.3.9 startWorker()

```
void BiometricEvaluation::Process::ForkManager::startWorker (
    std::shared_ptr< WorkerController > worker,
    bool wait = true,
    bool communicate = false ) [virtual]
```

Start a worker.

Parameters

	<i>worker</i>	Pointer to a WorkerController (p. 601) that is being managed by this Manager (p. 430) instance.
	<i>wait</i>	Whether or not to wait for this Worker (p. 595) to exit before returning control to the caller.
in	<i>communicate</i>	Whether or not to enable communication among the Workers and Managers.

Exceptions

Error::ObjectExists (p. 458)	worker is already working.
Error::StrategyError (p. 567)	worker is not managed by this Manager (p. 430) instance.

Implements **BiometricEvaluation::Process::Manager** (p. 433).

G.56.3.10 startWorkers()

```
void BiometricEvaluation::Process::ForkManager::startWorkers (
```

```
bool wait = true,
bool communicate = false ) [virtual]
```

Begin **Worker** (p. 595)'s work.

Parameters

in	<i>wait</i>	Whether or not to wait for all Workers to return before returning.
in	<i>communicate</i>	Whether or not to enable communication among the Workers and Managers.

Exceptions

Error::ObjectExists (p. 458)	At least one Worker (p. 595) is already working.
Error::StrategyError (p. 567)	Problem forking.

Implements **BiometricEvaluation::Process::Manager** (p. 434).

G.56.3.11 stopWorker()

```
void BiometricEvaluation::Process::ForkManager::stopWorker (
    std::shared_ptr< WorkerController > workerController ) [virtual]
```

Ask **Worker** (p. 595) to exit.

Sends SIGUSR1 to the **Worker** (p. 595), which **ForkManager** (p. 335) will handle automatically.

Parameters

<i>workerController</i>	Pointer to the ForkWorkerController (p. 341) that should be stopped.
-------------------------	---

Exceptions

Error::ObjectDoesNotExist (p. 457)	worker is not working.
Error::StrategyError (p. 567)	Problem sending the signal.

Attention

Do not call **stopWorker()** (p. 340) when communication is enabled unless you will be finished with communication for all Workers at that point. This creates a race condition for reads()/writes() when the **Worker** (p. 595) exits.

Implements **BiometricEvaluation::Process::Manager** (p. 434).

G.56.3.12 waitForWorkerExit()

```
void BiometricEvaluation::Process::ForkManager::waitForWorkerExit ( ) [virtual]
```

Block until all Workers have exited.

Use this method if wait=false was set during a call to startWorker(s) but now wait=true is desired.

Implements **BiometricEvaluation::Process::Manager** (p. 435).

G.56.4 Member Data Documentation

G.56.4.1 FORKMANAGERS

```
std::list< ForkManager* > BiometricEvaluation::Process::ForkManager::FORKMANAGERS [static]
```

List of all instantiated ForkManagers.

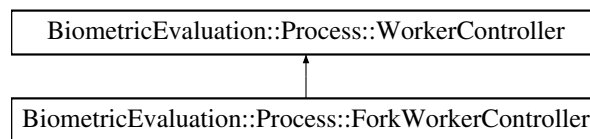
This is not a list of managed pointers to ForkManagers. If it was, the smart pointer's destructor would attempt to delete the object being pointed to at program termination, which is ultimately sometime after the destructor of the **ForkManager** (p. 335) itself was called.

G.57 BiometricEvaluation::Process::ForkWorkerController Class Reference

Wrapper of a **Worker** (p. 595) returned from a **Process::ForkManager** (p. 335).

```
#include <be_process_forkmanager.h>
```

Inheritance diagram for BiometricEvaluation::Process::ForkWorkerController:



Public Member Functions

- bool **isWorking** () const
*Obtain whether or not **Worker** (p. 595) is working.*
- bool **everWorked** () const
*Obtain whether or not this **Worker** (p. 595) has ever worked.*
- void **reset** ()
*Reuse the **Worker** (p. 595).*
- pid_t **getPID** () const
Obtain the PID of this process this instance represents.
- ~**ForkWorkerController** ()
***ForkWorkerController** (p. 341) destructor.*

Static Public Member Functions

- static void **_stop** (int signal)
*Tell **_staticWorker** to stop.*

Friends

- void **ForkManager::startWorkers** (bool wait, bool communicate)
*Begin **Worker** (p. 595)'s work.*
- void **ForkManager::startWorker** (std::shared_ptr< **WorkerController** > worker, bool wait, bool communicate)

- Restart a completed **Worker** (p. 595).
- void **ForkManager::stopWorker** (std::shared_ptr< **WorkerController** > workerController)
Ask **Worker** (p. 595) to exit.
- std::shared_ptr< **WorkerController** > **ForkManager::addWorker** (std::shared_ptr< **Worker** > worker)
Adds a **Worker** (p. 595) to be managed by this **Manager** (p. 430).
- void **ForkManager::setExitStatus** (const pid_t pid, const int32_t waitStatus)
Set the exit status in the **WorkerController** (p. 601) for given process ID.

Additional Inherited Members

G.57.1 Detailed Description

Wrapper of a **Worker** (p. 595) returned from a **Process::ForkManager** (p. 335).

G.57.2 Member Function Documentation

G.57.2.1 _stop()

```
static void BiometricEvaluation::Process::ForkWorkerController::_stop (
    int signal ) [static]
```

Tell _staticWorker to stop.

Called by the child process instance when SIGUSR1 is received.

Parameters

<i>signal</i>	The signal caught that prompted this function to be called (SIGUSR1).
---------------	---

G.57.2.2 everWorked()

```
bool BiometricEvaluation::Process::ForkWorkerController::everWorked ( ) const [virtual]
```

Obtain whether or not this **Worker** (p. 595) has ever worked.

Returns

true the **Worker** (p. 595) has ever or is currently working, false otherwise.

Note

reset() (p. 343) will change the result of this method.

Implements **BiometricEvaluation::Process::WorkerController** (p. 602).

G.57.2.3 getPID()

```
pid_t BiometricEvaluation::Process::ForkWorkerController::getPID ( ) const
```

Obtain the PID of this process this instance represents.

Returns

pid of the process this instance represents.

Note

Call `isRunning()` before doing anything with the PID returned from this function.

G.57.2.4 `isWorking()`

```
bool BiometricEvaluation::Process::ForkWorkerController::isWorking ( ) const [virtual]
```

Obtain whether or not **Worker** (p. 595) is working.

Returns

Whether or not the **Worker** (p. 595) is working.

Implements **BiometricEvaluation::Process::WorkerController** (p. 603).

G.57.2.5 `reset()`

```
void BiometricEvaluation::Process::ForkWorkerController::reset ( ) [virtual]
```

Reuse the **Worker** (p. 595).

Exceptions

<i>Error::ObjectExists</i> (p. 458)	The previously started Worker (p. 595) is still running.
-------------------------------------	---

Reimplemented from **BiometricEvaluation::Process::WorkerController** (p. 603).

G.57.3 Friends And Related Function Documentation

G.57.3.1 `ForkManager::addWorker`

```
std::shared_ptr< WorkerController> ForkManager::addWorker (
    std::shared_ptr< Worker > worker ) [friend]
```

Adds a **Worker** (p. 595) to be managed by this **Manager** (p. 430).

Parameters

<i>worker</i>	A Worker (p. 595) instance to run.
---------------	---

Returns

shared_ptr to worker.

G.57.3.2 ForkManager::setExitStatus

```
void ForkManager::setExitStatus (
    const pid_t pid,
    const int32_t waitStatus ) [friend]
```

Set the exit status in the **WorkerController** (p. 601) for given process ID.

Parameters

in	<i>pid</i>	PID whose exit status should be set.
in	<i>status</i>	Status, as returned from wait(2).

Exceptions

Error::ObjectDoesNotExist (p. 457)	PID not under this manager's control.
---	---------------------------------------

Note

Exit status is only set if process exited cleanly.

G.57.3.3 ForkManager::startWorker

```
void ForkManager::startWorker (
    std::shared_ptr< WorkerController > worker,
    bool wait,
    bool communicate ) [friend]
```

Restart a completed **Worker** (p. 595).

Parameters

	<i>worker</i>	Pointer to a WorkerController (p. 601) that is being managed by this Manager (p. 430) instance.
	<i>wait</i>	Whether or not to wait for this Worker (p. 595) to exit before returning control to the caller.
in	<i>communicate</i>	Whether or not to enable communication among the Workers and Managers.

Exceptions

Error::ObjectExists (p. 458)	worker is already working.
Error::StrategyError (p. 567)	worker is not managed by this Manager (p. 430) instance.

G.57.3.4 ForkManager::startWorkers

```
void ForkManager::startWorkers (
    bool wait,
    bool communicate ) [friend]
    Begin Worker (p. 595)'s work.
```

Parameters

in	wait	Whether or not to wait for all Workers to return before returning.
in	communicate	Whether or not to enable communication among the Workers and Managers.

Exceptions

Error::ObjectExists (p. 458)	One or more of the Workers is already working.
Error::StrategyError (p. 567)	Problem forking.

G.57.3.5 ForkManager::stopWorker

```
void ForkManager::stopWorker (
    std::shared_ptr< WorkerController > workerController ) [friend]
    Ask Worker (p. 595) to exit.
    Sends SIGUSR1 to the Worker (p. 595), which ForkManager (p. 335) will handle automatically.
```

Parameters

workerController	Pointer to the ForkWorkerController (p. 341) that should be stopped.
------------------	--

Exceptions

Error::ObjectDoesNotExist (p. 457)	worker is not working.
Error::StrategyError (p. 567)	Problem sending the signal.

G.58 BiometricEvaluation::Feature::AN2K11EFS::FPPPosition Struct Reference

Representation of finger-palm-plantar position.
#include <be_feature_an2k11efs.h>

Public Attributes

- Feature::FGP fgp

- bool **has_fsm**
- FingerprintSegment **fsm**
- bool **has_ocf**
- OffCenterFingerPosition **ocf**
- bool **has_sgp**
- BiometricEvaluation::Image::CoordinateSet **sgp**

G.58.1 Detailed Description

Representation of finger-palm-plantar position.

Contains one or more possible physical positions that correspond to the region of interest. Clients of this structure must check the fgp value to determine which of the position codes (Finger/Palm/Plantar) applies.

G.58.2 Member Data Documentation

G.58.2.1 fgp

Feature::FGP BiometricEvaluation::Feature::AN2K11EFS::FPPPosition::fgp

The friction ridge generalized position

G.58.2.2 fsm

FingerprintSegment BiometricEvaluation::Feature::AN2K11EFS::FPPPosition::fsm

The finger segment position

G.58.2.3 ocf

OffCenterFingerPosition BiometricEvaluation::Feature::AN2K11EFS::FPPPosition::ocf

The off-center fingerprint position

G.58.2.4 sgp

BiometricEvaluation::Image::CoordinateSet BiometricEvaluation::Feature::AN2K11EFS::FPPPosition←
::sgp

The segment polygon

G.59 BiometricEvaluation::Video::Frame Struct Reference

Public Attributes

- Image::Size **size**
- int64_t **timestamp**
- Memory::uint8Array **data**

G.60 BiometricEvaluation::Feature::FrictionRidgeGeneralizedPosition Struct Reference

Representation of the position (Finger/Palm/Plantar) used in this class and child classes.

```
#include <be_feature.h>
```

Public Attributes

- **PositionType** posType
- ```

union {
 Finger::Position fingerPos
 Palm::Position palmPos
 Planter::Position plantarPos
} position

```

### G.60.1 Detailed Description

Representation of the position (Finger/Palm/Planter) used in this class and child classes.

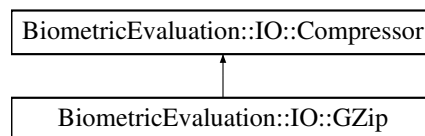
When the AN2K11 FGP field is read, it may represent a finger, palm, or plantar position. The union is tagged to indicate which position is present.

## G.61 BiometricEvaluation::IO::GZip Class Reference

**Compressor** (p. 270) for gzip compression from zlib.

```
#include <be_io_gzip.h>
```

Inheritance diagram for BiometricEvaluation::IO::GZip:



## Public Member Functions

- **Memory::uint8Array** **compress** (const uint8\_t \*const uncompressedData, uint64\_t uncompressedDataSize) const  
*Compress a buffer.*
- **Memory::uint8Array** **compress** (const **Memory::uint8Array** &uncompressedData) const  
*Compress a buffer.*
- void **compress** (const uint8\_t \*const uncompressedData, uint64\_t uncompressedDataSize, const std::string &outputFile) const  
*Compress a buffer.*
- void **compress** (const **Memory::uint8Array** &uncompressedData, const std::string &outputFile) const  
*Compress a buffer.*
- **Memory::uint8Array** **compress** (const std::string &inputFile) const  
*Compress a file.*
- void **compress** (const std::string &inputFile, const std::string &outputFile) const  
*Compress a file.*
- **Memory::uint8Array** **decompress** (const uint8\_t \*const compressedData, uint64\_t compressedDataSize) const  
*Decompress a compressed buffer.*

- **Memory::uint8Array decompress** (const **Memory::uint8Array** &compressedData) const  
*Decompress a compressed buffer.*
- **Memory::uint8Array decompress** (const std::string &input) const  
*Decompress a compressed buffer into a file.*
- void **decompress** (const std::string &inputFile, const std::string &outputFile) const  
*Decompress a file.*
- void **decompress** (const uint8\_t \*const compressedData, const uint64\_t compressedDataSize, const std::string &outputFile) const  
*Decompress a file.*
- void **decompress** (const **Memory::uint8Array** &compressedData, const std::string &outputFile) const  
*Decompress a file.*
- **GZip** (const **GZip** &other)=delete  
*Copy constructor (disabled).*
- **GZip & operator=** (const **GZip** &other)=delete  
*Assignment overload (disabled).*

## Static Public Attributes

- static const std::string **COMPRESSION\_LEVEL**
- static const std::string **COMPRESSION\_STRATEGY**
- static const std::string **COMPRESSION\_METHOD**
- static const std::string **INPUT\_DATA\_TYPE**
- static const std::string **WINDOW\_BITS**
- static const std::string **MEMORY\_LEVEL**
- static const std::string **CHUNK\_SIZE**

## Additional Inherited Members

### G.61.1 Detailed Description

**Compressor** (p. 270) for gzip compression from zlib.

### G.61.2 Constructor & Destructor Documentation

#### G.61.2.1 GZip()

```
BiometricEvaluation::IO::GZip::GZip (
 const GZip & other) [delete]
```

Copy constructor (disabled).

Disabled because **Properties** (p. 486) member of parent cannot be copied.

Parameters

|              |                               |
|--------------|-------------------------------|
| <i>other</i> | <b>GZip</b> (p. 347) to copy. |
|--------------|-------------------------------|

G.61.3 Member Function Documentation

G.61.3.1 compress() [1/6]

**Memory::uint8Array** BiometricEvaluation::IO::GZip::compress (   
const uint8\_t \*const *uncompressedData*,   
uint64\_t *uncompressedDataSize* ) const [virtual]   
Compress a buffer.

Parameters

|                             |                                       |
|-----------------------------|---------------------------------------|
| <i>uncompressedData</i>     | Uncompressed data buffer to compress. |
| <i>uncompressedDataSize</i> | Size of uncompressedData.             |

Returns

Compressed buffer.

Exceptions

|                                      |                                            |
|--------------------------------------|--------------------------------------------|
| <b>Error::StrategyError</b> (p. 567) | <b>Error</b> (p. 108) in compression unit. |
|--------------------------------------|--------------------------------------------|

Implements **BiometricEvaluation::IO::Compressor** (p. 272).

G.61.3.2 compress() [2/6]

**Memory::uint8Array** BiometricEvaluation::IO::GZip::compress (   
const **Memory::uint8Array** & *uncompressedData* ) const [virtual]   
Compress a buffer.

Parameters

|                         |                                       |
|-------------------------|---------------------------------------|
| <i>uncompressedData</i> | Uncompressed data buffer to compress. |
|-------------------------|---------------------------------------|

Returns

Compressed buffer.

Exceptions

|                                      |                                              |
|--------------------------------------|----------------------------------------------|
| <b>Error::StrategyError</b> (p. 567) | <b>Error</b> (p. 108) in decompression unit. |
|--------------------------------------|----------------------------------------------|

Implements **BiometricEvaluation::IO::Compressor** (p. 273).

**G.61.3.3 compress()** [3/6]

```
void BiometricEvaluation::IO::GZip::compress (
 const uint8_t *const uncompressedData,
 uint64_t uncompressedDataSize,
 const std::string & outputFile) const [virtual]
```

Compress a buffer.

Parameters

|                             |                                       |
|-----------------------------|---------------------------------------|
| <i>uncompressedData</i>     | Uncompressed data buffer to compress. |
| <i>uncompressedDataSize</i> | Size of <i>uncompressedData</i> .     |
| <i>outputFile</i>           | Location to save compressed file.     |

Exceptions

|                                                       |                                                             |
|-------------------------------------------------------|-------------------------------------------------------------|
| <b>Error::ObjectExists</b> (p. <a href="#">458</a> )  | Output file already exists.                                 |
| <b>Error::StrategyError</b> (p. <a href="#">567</a> ) | <b>Error</b> (p. <a href="#">108</a> ) in compression unit. |

Implements **BiometricEvaluation::IO::Compressor** (p. [273](#)).

**G.61.3.4 compress()** [4/6]

```
void BiometricEvaluation::IO::GZip::compress (
 const Memory::uint8Array & uncompressedData,
 const std::string & outputFile) const [virtual]
```

Compress a buffer.

Parameters

|                         |                                       |
|-------------------------|---------------------------------------|
| <i>uncompressedData</i> | Uncompressed data buffer to compress. |
| <i>outputFile</i>       | Location to save compressed file.     |

Exceptions

|                                                       |                                                               |
|-------------------------------------------------------|---------------------------------------------------------------|
| <b>Error::ObjectExists</b> (p. <a href="#">458</a> )  | Output file already exists.                                   |
| <b>Error::StrategyError</b> (p. <a href="#">567</a> ) | <b>Error</b> (p. <a href="#">108</a> ) in decompression unit. |

Implements **BiometricEvaluation::IO::Compressor** (p. [274](#)).

**G.61.3.5 compress()** [5/6]

```
Memory::uint8Array BiometricEvaluation::IO::GZip::compress (
 const std::string & inputFile) const [virtual]
```

Compress a file.



## Parameters

|                  |                           |
|------------------|---------------------------|
| <i>inputFile</i> | Path to file to compress. |
|------------------|---------------------------|

## Returns

Compressed buffer.

## Exceptions

|                                           |                                              |
|-------------------------------------------|----------------------------------------------|
| <b>Error::ObjectDoesNotExist</b> (p. 457) | Input file does not exist.                   |
| <b>Error::StrategyError</b> (p. 567)      | <b>Error</b> (p. 108) in decompression unit. |

Implements **BiometricEvaluation::IO::Compressor** (p. 274).

## G.61.3.6 compress() [6/6]

```
void BiometricEvaluation::IO::GZip::compress (
 const std::string & inputFile,
 const std::string & outputFile) const [virtual]
```

Compress a file.

## Parameters

|                   |                                                          |
|-------------------|----------------------------------------------------------|
| <i>inputFile</i>  | Path to file to compress.                                |
| <i>outputFile</i> | Path to location where compressed version will be saved. |

## Exceptions

|                                           |                                              |
|-------------------------------------------|----------------------------------------------|
| <b>Error::ObjectDoesNotExist</b> (p. 457) | Input file does not exist.                   |
| <b>Error::ObjectExists</b> (p. 458)       | Output file already exists.                  |
| <b>Error::StrategyError</b> (p. 567)      | <b>Error</b> (p. 108) in decompression unit. |

Implements **BiometricEvaluation::IO::Compressor** (p. 274).

## G.61.3.7 decompress() [1/6]

```
Memory::uint8Array BiometricEvaluation::IO::GZip::decompress (
 const uint8_t *const compressedData,
 uint64_t compressedDataSize) const [virtual]
```

Decompress a compressed buffer.

## Parameters

|                           |                                       |
|---------------------------|---------------------------------------|
| <i>compressedData</i>     | Compressed data buffer to decompress. |
| <i>compressedDataSize</i> | Size of compressedData.               |

## Returns

Decompressed data.

## Exceptions

|                                                       |                                                             |
|-------------------------------------------------------|-------------------------------------------------------------|
| <b>Error::StrategyError</b> (p. <a href="#">567</a> ) | <b>Error</b> (p. <a href="#">108</a> ) in compression unit. |
|-------------------------------------------------------|-------------------------------------------------------------|

Implements **BiometricEvaluation::IO::Compressor** (p. [275](#)).

## G.61.3.8 decompress() [2/6]

**Memory::uint8Array** BiometricEvaluation::IO::GZip::decompress (   
     const **Memory::uint8Array** & *compressedData* ) const [virtual]   
 Decompress a compressed buffer.

## Parameters

|                       |                                       |
|-----------------------|---------------------------------------|
| <i>compressedData</i> | Compressed data buffer to decompress. |
|-----------------------|---------------------------------------|

## Returns

Decompressed data.

## Exceptions

|                                                       |                                                               |
|-------------------------------------------------------|---------------------------------------------------------------|
| <b>Error::StrategyError</b> (p. <a href="#">567</a> ) | <b>Error</b> (p. <a href="#">108</a> ) in decompression unit. |
|-------------------------------------------------------|---------------------------------------------------------------|

Implements **BiometricEvaluation::IO::Compressor** (p. [276](#)).

## G.61.3.9 decompress() [3/6]

**Memory::uint8Array** BiometricEvaluation::IO::GZip::decompress (   
     const std::string & *inputFile* ) const [virtual]   
 Decompress a compressed buffer into a file.

## Parameters

|                  |                                   |
|------------------|-----------------------------------|
| <i>inputFile</i> | Location to save compressed file. |
|------------------|-----------------------------------|

## Returns

Decompressed data.

## Exceptions

|                                                       |                                                               |
|-------------------------------------------------------|---------------------------------------------------------------|
| <b>Error::StrategyError</b> (p. <a href="#">567</a> ) | <b>Error</b> (p. <a href="#">108</a> ) in decompression unit. |
|-------------------------------------------------------|---------------------------------------------------------------|

## Exceptions

|                                  |                             |
|----------------------------------|-----------------------------|
| <i>Error::ObjectDoesNotExist</i> | Output file already exists. |
|----------------------------------|-----------------------------|

Implements **BiometricEvaluation::IO::Compressor** (p. 276).

**G.61.3.10 decompress()** [4/6]

```
void BiometricEvaluation::IO::GZip::decompress (
 const std::string & inputFile,
 const std::string & outputFile) const [virtual]
 Decompress a file.
```

## Parameters

|                   |                                                            |
|-------------------|------------------------------------------------------------|
| <i>inputFile</i>  | Path to file to decompress.                                |
| <i>outputFile</i> | Path to location where decompressed version will be saved. |

## Exceptions

|                                           |                                            |
|-------------------------------------------|--------------------------------------------|
| <i>Error::ObjectDoesNotExist</i> (p. 457) | Input file does not exist.                 |
| <i>Error::ObjectExists</i> (p. 458)       | Output file already exists.                |
| <i>Error::StrategyError</i> (p. 567)      | <b>Error</b> (p. 108) in compression unit. |

Implements **BiometricEvaluation::IO::Compressor** (p. 278).

**G.61.3.11 decompress()** [5/6]

```
void BiometricEvaluation::IO::GZip::decompress (
 const uint8_t *const compressedData,
 const uint64_t compressedDataSize,
 const std::string & outputFile) const [virtual]
 Decompress a file.
```

## Parameters

|                           |                                                            |
|---------------------------|------------------------------------------------------------|
| <i>compressedData</i>     | Compressed data buffer to decompress.                      |
| <i>compressedDataSize</i> | Size of compressedData.                                    |
| <i>outputFile</i>         | Path to location where decompressed version will be saved. |

## Exceptions

|                                      |                                            |
|--------------------------------------|--------------------------------------------|
| <i>Error::ObjectExists</i> (p. 458)  | Output file already exists.                |
| <i>Error::StrategyError</i> (p. 567) | <b>Error</b> (p. 108) in compression unit. |

Implements **BiometricEvaluation::IO::Compressor** (p. 277).

### G.61.3.12 decompress() [6/6]

```
void BiometricEvaluation::IO::GZip::decompress (
 const Memory::uint8Array & compressedData,
 const std::string & outputFile) const [virtual]
```

Decompress a file.

Parameters

|                       |                                                            |
|-----------------------|------------------------------------------------------------|
| <i>compressedData</i> | Compressed data buffer to decompress.                      |
| <i>outputFile</i>     | Path to location where decompressed version will be saved. |

Exceptions

|                                      |                                            |
|--------------------------------------|--------------------------------------------|
| <b>Error::ObjectExists</b> (p. 458)  | Output file already exists.                |
| <b>Error::StrategyError</b> (p. 567) | <b>Error</b> (p. 108) in compression unit. |

Implements **BiometricEvaluation::IO::Compressor** (p. 277).

### G.61.3.13 operator=()

```
GZip& BiometricEvaluation::IO::GZip::operator= (
 const GZip & other) [delete]
```

Assignment overload (disabled).  
Disabled because **Properties** (p. 486) member of parent cannot be assigned.

Parameters

|              |                                 |
|--------------|---------------------------------|
| <i>other</i> | <b>GZip</b> (p. 347) to assign. |
|--------------|---------------------------------|

Returns

lhs **GZip** (p. 347).

## G.61.4 Member Data Documentation

### G.61.4.1 CHUNK\_SIZE

```
const std::string BiometricEvaluation::IO::GZip::CHUNK_SIZE [static]
```

How many bytes to work at a time

### G.61.4.2 COMPRESSION\_LEVEL

```
const std::string BiometricEvaluation::IO::GZip::COMPRESSION_LEVEL [static]
```

How thorough the compression should be

#### G.61.4.3 COMPRESSION\_METHOD

```
const std::string BiometricEvaluation::IO::GZip::COMPRESSION_METHOD [static]
```

Which underlying method in the compressor

#### G.61.4.4 COMPRESSION\_STRATEGY

```
const std::string BiometricEvaluation::IO::GZip::COMPRESSION_STRATEGY [static]
```

Which underlying algorithm to use

#### G.61.4.5 INPUT\_DATA\_TYPE

```
const std::string BiometricEvaluation::IO::GZip::INPUT_DATA_TYPE [static]
```

The type of data being compressed

#### G.61.4.6 MEMORY\_LEVEL

```
const std::string BiometricEvaluation::IO::GZip::MEMORY_LEVEL [static]
```

How much memory for internal compression state

#### G.61.4.7 WINDOW\_BITS

```
const std::string BiometricEvaluation::IO::GZip::WINDOW_BITS [static]
```

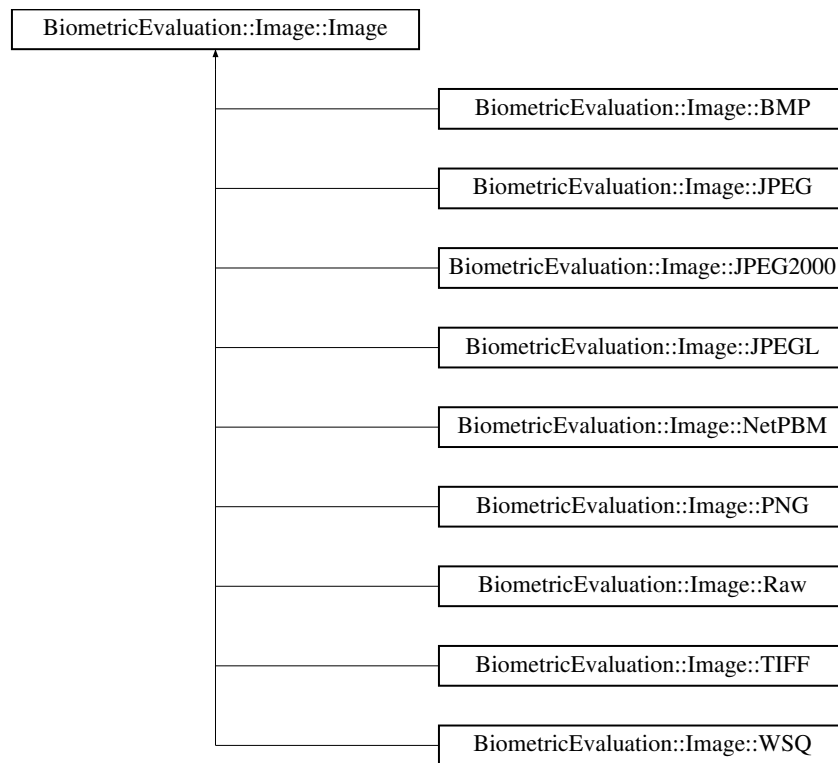
Window size

## G.62 BiometricEvaluation::Image::Image Class Reference

Represent attributes common to all images.

```
#include <be_image_image.h>
```

Inheritance diagram for BiometricEvaluation::Image::Image:



## Public Member Functions

- **Image** (const uint8\_t \*data, const uint64\_t size, const **Size** dimensions, const uint32\_t colorDepth, const uint16\_t bitDepth, const **Resolution** resolution, const **CompressionAlgorithm** compression, const bool hasAlphaChannel)

*Parent constructor for all **Image** (p. 355) classes.*

- **Image** (const uint8\_t \*data, const uint64\_t size, const **CompressionAlgorithm** compression)

*Parent constructor for all **Image** (p. 355) classes.*

- **CompressionAlgorithm** **getCompressionAlgorithm** () const

*Accessor for the CompressionAlgorithm of the image.*

- **Resolution** **getResolution** () const

*Accessor for the resolution of the image.*

- **Memory::uint8Array** **getData** () const

*Accessor for the image data. The data returned is likely encoded in a specialized format.*

- virtual **Memory::uint8Array** **getRawData** () const =0

*Accessor for the raw image data. The data returned should not be compressed or encoded.*

- virtual **Memory::uint8Array** **getRawData** (const bool removeAlphaChannelIfPresent) const

*Accessor for the raw image data. The data returned should not be compressed or encoded.*

- virtual **Memory::uint8Array** **getRawGrayscaleData** (uint8\_t depth) const =0

*Accessor for decompressed data in grayscale.*

- **Size** **getDimensions** () const

*Accessor for the dimensions of the image in pixels.*

- uint32\_t **getColorDepth** () const

- *Accessor for the color depth of the image in bits.*
- uint16\_t **getBitDepth** () const
- *Accessor for the number of bits per color component.*
- bool **hasAlphaChannel** () const
- *Accessor for the presence of an alpha channel.*

## Static Public Member Functions

- static uint64\_t **valueInColorspace** (uint64\_t color, uint64\_t maxColorValue, uint8\_t depth)  
*Calculate an equivalent color value for a color in an alternate colorspace.*
- static std::shared\_ptr< **Image** > **openImage** (const uint8\_t \*data, const uint64\_t size)  
*Determine the image type of a buffer of image data and create an **Image** (p. 355) object.*
- static std::shared\_ptr< **Image** > **openImage** (const **Memory::uint8Array** &data)  
*Determine the image type of a buffer of image data and create an **Image** (p. 355) object.*
- static std::shared\_ptr< **Image** > **openImage** (const std::string &path)  
*Determine the image type of an image file and create an **Image** (p. 355) object.*
- static **CompressionAlgorithm** **getCompressionAlgorithm** (const uint8\_t \*data, const uint64\_t size)  
*Determine the compression algorithm of a buffer of image data.*
- static **CompressionAlgorithm** **getCompressionAlgorithm** (const **Memory::uint8Array** &data)  
*Determine the compression algorithm of a buffer of image data.*
- static **CompressionAlgorithm** **getCompressionAlgorithm** (const std::string &path)  
*Determine the compression algorithm of a file.*
- static **BiometricEvaluation::Image::Raw** **getRawImage** (const std::shared\_ptr< **BiometricEvaluation::Image::Image** > &image)  
*Obtain **Image::Raw** (p. 497) version of an **Image::Image** (p. 355).*

## Protected Member Functions

- void **setResolution** (const **Resolution** resolution)  
*Mutator for the resolution of the image .*
- void **setDimensions** (const **Size** dimensions)  
*Mutator for the dimensions of the image in pixels.*
- void **setColorDepth** (const uint32\_t colorDepth)  
*Mutator for the color depth of the image in bits.*
- void **setBitDepth** (const uint16\_t bitDepth)  
*Mutator for the number of bits per component for color components in the image, in bits.*
- const uint8\_t \* **getDataPointer** () const
- uint64\_t **getDataSize** () const
- void **setHasAlphaChannel** (const bool **hasAlphaChannel**)  
*Mutator for the presence of an alpha channel.*

### G.62.1 Detailed Description

Represent attributes common to all images.

Images are represented by their size, depth, and resolution on the X and Y axes. The image data can be of any format, raw, **JPEG** (p. 406), etc. Implementations of this abstraction provide the **getRawData** method to convert image data to 'raw' format.

**Image** (p. 355) resolution is in pixels per centimeter, and the coordinate system has the origin at the upper left of the image.

## G.62.2 Constructor & Destructor Documentation

### G.62.2.1 Image() [1/2]

```
BiometricEvaluation::Image::Image::Image (
 const uint8_t * data,
 const uint64_t size,
 const Size dimensions,
 const uint32_t colorDepth,
 const uint16_t bitDepth,
 const Resolution resolution,
 const CompressionAlgorithm compression,
 const bool hasAlphaChannel)
```

Parent constructor for all **Image** (p. 355) classes.

#### Parameters

|    |                        |                                              |
|----|------------------------|----------------------------------------------|
| in | <i>data</i>            | The image data.                              |
| in | <i>size</i>            | The size of the image data, in bytes.        |
| in | <i>dimensions</i>      | The width and height of the image in pixels. |
| in | <i>colorDepth</i>      | The image color depth, in bits-per-pixel.    |
| in | <i>bitDepth</i>        | The number of bits per color component.      |
| in | <i>resolution</i>      | The resolution of the image                  |
| in | <i>compression</i>     | The CompressionAlgorithm of data.            |
| in | <i>hasAlphaChannel</i> | Presence of an alpha channel.                |

#### Exceptions

|                                      |                                                             |
|--------------------------------------|-------------------------------------------------------------|
| <b>Error::StrategyError</b> (p. 567) | <b>Error</b> (p. 108) manipulating data.                    |
| <b>Error::StrategyError</b> (p. 567) | <b>Error</b> (p. 108) while creating <b>Image</b> (p. 355). |

### G.62.2.2 Image() [2/2]

```
BiometricEvaluation::Image::Image::Image (
 const uint8_t * data,
 const uint64_t size,
 const CompressionAlgorithm compression)
```

Parent constructor for all **Image** (p. 355) classes.

#### Parameters

|    |                    |                                       |
|----|--------------------|---------------------------------------|
| in | <i>data</i>        | The image data.                       |
| in | <i>size</i>        | The size of the image data, in bytes. |
| in | <i>compression</i> | The CompressionAlgorithm of data.     |



## Exceptions

|                                                       |                                                                                               |
|-------------------------------------------------------|-----------------------------------------------------------------------------------------------|
| <i>Error::DataError</i> (p. <a href="#">295</a> )     | <b>Error</b> (p. <a href="#">108</a> ) manipulating data.                                     |
| <i>Error::StrategyError</i> (p. <a href="#">567</a> ) | <b>Error</b> (p. <a href="#">108</a> ) while creating <b>Image</b> (p. <a href="#">355</a> ). |

## G.62.3 Member Function Documentation

## G.62.3.1 getBitDepth()

```
uint16_t BiometricEvaluation::Image::Image::getBitDepth () const
```

Accessor for the number of bits per color component.

## Returns

The bit depth of the image (in bits).

## G.62.3.2 getColorDepth()

```
uint32_t BiometricEvaluation::Image::Image::getColorDepth () const
```

Accessor for the color depth of the image in bits.

## Returns

The color depth of the image (bit).

## G.62.3.3 getCompressionAlgorithm() [1/4]

```
CompressionAlgorithm BiometricEvaluation::Image::Image::getCompressionAlgorithm () const
```

Accessor for the CompressionAlgorithm of the image.

## Returns

Type of compression used on the data that will be returned from **getData()** (p. [361](#)).

## G.62.3.4 getCompressionAlgorithm() [2/4]

```
static CompressionAlgorithm BiometricEvaluation::Image::Image::getCompressionAlgorithm (
 const uint8_t * data,
 const uint64_t size) [static]
```

Determine the compression algorithm of a buffer of image data.

## Parameters

|    |             |                                       |
|----|-------------|---------------------------------------|
| in | <i>data</i> | The image data.                       |
| in | <i>size</i> | The size of the image data, in bytes. |

## Returns

Compression algorithm used in the buffer.

## Attention

CompressionAlgorithm::None is returned if no compression algorithm known to the Biometric Evaluation **Framework** (p. 117) is found.

**G.62.3.5** getCompressionAlgorithm() [3/4]

```
static CompressionAlgorithm BiometricEvaluation::Image::Image::getCompressionAlgorithm (
 const Memory::uint8Array & data) [static]
```

Determine the compression algorithm of a buffer of image data.

## Parameters

|    |             |                 |
|----|-------------|-----------------|
| in | <i>data</i> | The image data. |
|----|-------------|-----------------|

## Returns

Compression algorithm used in the buffer.

## Attention

CompressionAlgorithm::None is returned if no compression algorithm known to the Biometric Evaluation **Framework** (p. 117) is found.

**G.62.3.6** getCompressionAlgorithm() [4/4]

```
static CompressionAlgorithm BiometricEvaluation::Image::Image::getCompressionAlgorithm (
 const std::string & path) [static]
```

Determine the compression algorithm of a file.

## Parameters

|    |             |               |
|----|-------------|---------------|
| in | <i>path</i> | Path to file. |
|----|-------------|---------------|

## Returns

Compression algorithm used in the file.

## Exceptions

|                                           |                                                             |
|-------------------------------------------|-------------------------------------------------------------|
| <b>Error::ObjectDoesNotExist</b> (p. 457) | path does not exist.                                        |
| <b>Error::StrategyError</b> (p. 567)      | An error occurred when using the underlying storage system. |

## Attention

CompressionAlgorithm::None is returned if no compression algorithm known to the Biometric Evaluation Framework (p. 117) is found.

**G.62.3.7** `getData()`

**Memory::uint8Array** BiometricEvaluation::Image::Image::getData ( ) const

Accessor for the image data. The data returned is likely encoded in a specialized format.

## Returns

AutoArray holding image data.

**G.62.3.8** `getDataPointer()`

const uint8\_t\* BiometricEvaluation::Image::Image::getDataPointer ( ) const [protected]

## Returns

Const pointer to buffer underlying \_data.

**G.62.3.9** `getDataSize()`

uint64\_t BiometricEvaluation::Image::Image::getDataSize ( ) const [protected]

## Returns

**Size** (p. 549) of \_data.

**G.62.3.10** `getDimensions()`

**Size** BiometricEvaluation::Image::Image::getDimensions ( ) const

Accessor for the dimensions of the image in pixels.

## Returns

**Coordinate** (p. 284) object containing dimensions in pixels.

**G.62.3.11** `getRawData()` [1/2]

virtual **Memory::uint8Array** BiometricEvaluation::Image::Image::getRawData ( ) const [pure virtual]

Accessor for the raw image data. The data returned should not be compressed or encoded.

## Returns

AutoArray holding raw image data.

## Exceptions

|                                  |                                                 |
|----------------------------------|-------------------------------------------------|
| <b>Error::DataError</b> (p. 295) | <b>Error</b> (p. 108) decompressing image data. |
|----------------------------------|-------------------------------------------------|

Implemented in **BiometricEvaluation::Image::NetPBM** (p. 454), **BiometricEvaluation::Image::JPEG** (p. 407), **BiometricEvaluation::Image::BMP** (p. 252), **BiometricEvaluation::Image::JPEG2000** (p. 409), **BiometricEvaluation::Image::Raw** (p. 497), **BiometricEvaluation::Image::JPEG** (p. 411), **BiometricEvaluation::Image::PNG** (p. 477), **BiometricEvaluation::Image::WSQ** (p. 611), and **BiometricEvaluation::Image::TIFF** (p. 576).

**G.62.3.12** **getRawData()** [2/2]

```
virtual Memory::uint8Array BiometricEvaluation::Image::Image::getRawData (
 const bool removeAlphaChannelIfPresent) const [virtual]
```

Accessor for the raw image data. The data returned should not be compressed or encoded.

## Parameters

|    |                                    |                                                          |
|----|------------------------------------|----------------------------------------------------------|
| in | <i>removeAlphaChannelIfPresent</i> | Whether or not to remove an alpha channel if one exists. |
|----|------------------------------------|----------------------------------------------------------|

## Returns

AutoArray holding raw image data, without an alpha channel if requested.

## Exceptions

|                                       |                                                          |
|---------------------------------------|----------------------------------------------------------|
| <b>Error::DataError</b> (p. 295)      | <b>Error</b> (p. 108) decompressing image data.          |
| <b>Error::ParameterError</b> (p. 474) | Propagated from <b>Image::removeComponents</b> (p. 123). |
| <b>Error::StrategyError</b> (p. 567)  | Propagated from <b>Image::removeComponents</b> (p. 123). |

**G.62.3.13** **getRawGrayscaleData()**

```
virtual Memory::uint8Array BiometricEvaluation::Image::Image::getRawGrayscaleData (
 uint8_t depth) const [pure virtual]
```

Accessor for decompressed data in grayscale.

## Parameters

|              |                                                                                         |
|--------------|-----------------------------------------------------------------------------------------|
| <i>depth</i> | The desired bit depth of the resulting raw image. This value may either be 16, 8, or 1. |
|--------------|-----------------------------------------------------------------------------------------|

## Returns

AutoArray holding raw grayscale image data.

## Exceptions

|                                       |                                                     |
|---------------------------------------|-----------------------------------------------------|
| <b>Error::DataError</b> (p. 295)      | <b>Error</b> (p. 108) decompressing image data.     |
| <b>Error::NotImplemented</b> (p. 456) | Unsupported conversion based on source color depth. |
| <b>Error::ParameterError</b> (p. 474) | Invalid value for depth.                            |

## Note

This method does not save a cached copy of the decompressed image because the bit depth of the image can be changed between calls.

When depth is 1, this method returns an image that uses 8 bits to represent a single pixel. The depth parameter is used to adjust the number of gray levels. When depth is 1, there are only 2 gray levels (black and white), despite using 8 bits to represent each pixel.

Alpha channels are completely ignored when converting to grayscale.

Implemented in **BiometricEvaluation::Image::NetPBM** (p. 454), **BiometricEvaluation::Image::JPEG** (p. 407), **BiometricEvaluation::Image::BMP** (p. 253), **BiometricEvaluation::Image::JPEG2000** (p. 410), **BiometricEvaluation::Image::Raw** (p. 498), **BiometricEvaluation::Image::PNG** (p. 477), **BiometricEvaluation::Image::WSQ** (p. 612), **BiometricEvaluation::Image::TIFF** (p. 577), and **BiometricEvaluation::Image::JPEGL** (p. 412).

## G.62.3.14 getRawImage()

```
static BiometricEvaluation::Image::Raw BiometricEvaluation::Image::Image::getRawImage (
 const std::shared_ptr< BiometricEvaluation::Image::Image > & image) [static]
```

Obtain **Image::Raw** (p. 497) version of an **Image::Image** (p. 355).

## Parameters

|    |              |                                                    |
|----|--------------|----------------------------------------------------|
| in | <i>image</i> | Shared pointer to an <b>Image::Image</b> (p. 355). |
|----|--------------|----------------------------------------------------|

## Returns

Shared pointer to an **Image::Raw** (p. 497) version of image.

## Note

If image is already an **Image::Raw** (p. 497), image is returned to avoid a copy.

## G.62.3.15 getResolution()

```
Resolution BiometricEvaluation::Image::Image::getResolution () const
```

Accessor for the resolution of the image.

## Returns

**Resolution** (p. 531) struct

**G.62.3.16 hasAlphaChannel()**

```
bool BiometricEvaluation::Image::Image::hasAlphaChannel () const [inline]
```

Accessor for the presence of an alpha channel.

Returns

Whether or not an alpha channel is present.

**G.62.3.17 openImage() [1/3]**

```
static std::shared_ptr< Image> BiometricEvaluation::Image::Image::openImage (
 const uint8_t * data,
 const uint64_t size) [static]
```

Determine the image type of a buffer of image data and create an **Image** (p. 355) object.

Parameters

|    |             |                                       |
|----|-------------|---------------------------------------|
| in | <i>data</i> | The image data.                       |
| in | <i>size</i> | The size of the image data, in bytes. |

Returns

**Image** (p. 355) representation of the input data buffer.

Exceptions

|                                      |                                                             |
|--------------------------------------|-------------------------------------------------------------|
| <b>Error::DataError</b> (p. 295)     | <b>Error</b> (p. 108) manipulating data.                    |
| <b>Error::StrategyError</b> (p. 567) | <b>Error</b> (p. 108) while creating <b>Image</b> (p. 355). |

**G.62.3.18 openImage() [2/3]**

```
static std::shared_ptr< Image> BiometricEvaluation::Image::Image::openImage (
 const Memory::uint8Array & data) [static]
```

Determine the image type of a buffer of image data and create an **Image** (p. 355) object.

Parameters

|    |             |                 |
|----|-------------|-----------------|
| in | <i>data</i> | The image data. |
|----|-------------|-----------------|

Returns

**Image** (p. 355) representation of the input data buffer.

## Exceptions

|                                      |                                                             |
|--------------------------------------|-------------------------------------------------------------|
| <b>Error::DataError</b> (p. 295)     | <b>Error</b> (p. 108) manipulating data.                    |
| <b>Error::StrategyError</b> (p. 567) | <b>Error</b> (p. 108) while creating <b>Image</b> (p. 355). |

## G.62.3.19 openImage() [3/3]

```
static std::shared_ptr< Image> BiometricEvaluation::Image::Image::openImage (
 const std::string & path) [static]
```

Determine the image type of an image file and create an **Image** (p. 355) object.

## Parameters

|    |             |                     |
|----|-------------|---------------------|
| in | <i>path</i> | Path to image data. |
|----|-------------|---------------------|

## Returns

**Image** (p. 355) representation of the input data buffer.

## Exceptions

|                                           |                                                             |
|-------------------------------------------|-------------------------------------------------------------|
| <b>Error::DataError</b> (p. 295)          | <b>Error</b> (p. 108) manipulating data.                    |
| <b>Error::ObjectDoesNotExist</b> (p. 457) | No file at specified path.                                  |
| <b>Error::StrategyError</b> (p. 567)      | <b>Error</b> (p. 108) while creating <b>Image</b> (p. 355). |

## G.62.3.20 setBitDepth()

```
void BiometricEvaluation::Image::Image::setBitDepth (
 const uint16_t bitDepth) [protected]
```

Mutator for the number of bits per component for color components in the image, in bits.

## Parameters

|    |                 |                                         |
|----|-----------------|-----------------------------------------|
| in | <i>bitDepth</i> | The number of bits per color component. |
|----|-----------------|-----------------------------------------|

## G.62.3.21 setColorDepth()

```
void BiometricEvaluation::Image::Image::setColorDepth (
 const uint32_t colorDepth) [protected]
```

Mutator for the color depth of the image in bits.

Parameters

|    |                   |                                     |
|----|-------------------|-------------------------------------|
| in | <i>colorDepth</i> | The color depth of the image (bit). |
|----|-------------------|-------------------------------------|

### G.62.3.22 setDimensions()

```
void BiometricEvaluation::Image::Image::setDimensions (
 const Size dimensions) [protected]
```

Mutator for the dimensions of the image in pixels.

Parameters

|    |                   |                              |
|----|-------------------|------------------------------|
| in | <i>dimensions</i> | Dimensions of image (pixel). |
|----|-------------------|------------------------------|

### G.62.3.23 setHasAlphaChannel()

```
void BiometricEvaluation::Image::Image::setHasAlphaChannel (
 const bool hasAlphaChannel) [inline], [protected]
```

Mutator for the presence of an alpha channel.

Parameters

|    |                        |                                            |
|----|------------------------|--------------------------------------------|
| in | <i>hasAlphaChannel</i> | Whether or not image has an alpha channel. |
|----|------------------------|--------------------------------------------|

### G.62.3.24 setResolution()

```
void BiometricEvaluation::Image::Image::setResolution (
 const Resolution resolution) [protected]
```

Mutator for the resolution of the image .

Parameters

|    |                   |                                                     |
|----|-------------------|-----------------------------------------------------|
| in | <i>resolution</i> | <b>Resolution</b> (p. <a href="#">531</a> ) struct. |
|----|-------------------|-----------------------------------------------------|

### G.62.3.25 valueInColorspace()

```
static uint64_t BiometricEvaluation::Image::Image::valueInColorspace (
 uint64_t color,
 uint64_t maxColorValue,
 uint8_t depth) [static]
```

Calculate an equivalent color value for a color in an alternate colorspace.



## Parameters

|                      |                                                  |
|----------------------|--------------------------------------------------|
| <i>color</i>         | Value for color in original colorspace.          |
| <i>maxColorValue</i> | Maximum value for colors in original colorspace. |
| <i>depth</i>         | Desired bit-depth of the new colorspace.         |

## Returns

A value equivalent to color in depth-bit space.

## G.63 BiometricEvaluation::Feature::AN2K11EFS::ImageInfo Struct Reference

A structure representing information about the image and extended feature set region.

```
#include <be_feature_an2k11efs.h>
```

### Public Attributes

- **BiometricEvaluation::Image::ROI** **roi**
- **FPPPosition** **fpp**
- **Orientation** **ort**
- bool **has\_trv**
- TonalReversal **trv**
- bool **has\_plr**
- LateralReversal **plr**

### G.63.1 Detailed Description

A structure representing information about the image and extended feature set region.

### G.63.2 Member Data Documentation

#### G.63.2.1 fpp

**FPPPosition** BiometricEvaluation::Feature::AN2K11EFS::ImageInfo::fpp  
The Finger/Palm/Plantar Position: Mandatory field.

#### G.63.2.2 ort

**Orientation** BiometricEvaluation::Feature::AN2K11EFS::ImageInfo::ort  
The image orientation. Optional but always present due to default value.

#### G.63.2.3 plr

**LateralReversal** BiometricEvaluation::Feature::AN2K11EFS::ImageInfo::plr  
The possible latent reversal information. Optional.

### G.63.2.4 roi

**BiometricEvaluation::Image::ROI** BiometricEvaluation::Feature::AN2K11EFS::ImageInfo::roi

The region of interest: A mandatory field.

### G.63.2.5 trv

TonalReversal BiometricEvaluation::Feature::AN2K11EFS::ImageInfo::trv

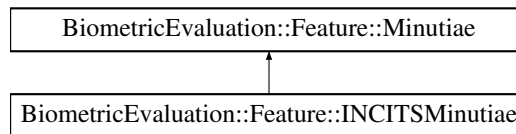
The tonal reversal information. Optional.

## G.64 BiometricEvaluation::Feature::INCITSMinutiae Class Reference

A class to represent a set of minutiae in an ANSI/INCITS record.

```
#include <be_feature_incitsminutiae.h>
```

Inheritance diagram for BiometricEvaluation::Feature::INCITSMinutiae:



### Public Member Functions

- **MinutiaeFormat** **getFormat** () const  
*Obtain the minutiae format kind.*
- **MinutiaPointSet** **getMinutiaPoints** () const  
*Obtain the set of finger minutiae data points. The set may be empty.*
- **RidgeCountItemSet** **getRidgeCountItems** () const  
*Obtain the set of ridge count data items. The set may be empty.*
- **CorePointSet** **getCores** () const  
*Obtains the set of core positions. The set may be empty.*
- **DeltaPointSet** **getDeltas** () const  
*Obtains the set of delta positions. The set may be empty.*
- **INCITSMinutiae** (const MinutiaPointSet &mps, const RidgeCountItemSet &rcis, const CorePointSet &cps, const DeltaPointSet &dps)  
*Construct an INCITS **Minutiae** (p. 442) object from its components.*
- **INCITSMinutiae** ()  
*Default constructor for an INCITS **Minutiae** (p. 442) object.*
- void **setMinutiaPoints** (const MinutiaPointSet &mps)  
*Mutator for the minutiae point set.*
- void **setRidgeCountItems** (const RidgeCountItemSet &rcis)  
*Mutator for the ridge count items.*
- void **setCorePointSet** (const CorePointSet &cps)  
*Mutator for the set of core points.*
- void **setDeltaPointSet** (const DeltaPointSet &dps)  
*Mutator for the set of delta points.*

## Static Public Attributes

- static const std::string **FMR\_ANSI\_SPEC\_VERSION**
- static const std::string **FMR\_ISO\_SPEC\_VERSION**
- static const std::string **FMR\_ANSI07\_SPEC\_VERSION**
- static const uint8\_t **FMR\_SPEC\_VERSION\_LEN** = 4
- static const uint32\_t **FED\_HEADER\_LENGTH** = 4
- static const uint32\_t **FED\_RCD\_ITEM\_LENGTH** = 3
- static const uint16\_t **FMD\_MINUTIA\_TYPE\_MASK** = 0xC000
- static const uint16\_t **FMD\_RESERVED\_MASK** = 0xC000
- static const uint16\_t **FMD\_MINUTIA\_TYPE\_SHIFT** = 14
- static const uint16\_t **FMD\_RESERVED\_SHIFT** = 14
- static const uint16\_t **FMD\_X\_COORD\_MASK** = 0x3FFF
- static const uint16\_t **FMD\_Y\_COORD\_MASK** = 0x3FFF
- static const uint16\_t **FMD\_ISO\_COMPACT\_MINUTIA\_TYPE\_MASK** = 0xC0
- static const uint16\_t **FMD\_ISO\_COMPACT\_MINUTIA\_TYPE\_SHIFT** = 6
- static const uint16\_t **FMD\_ISO\_COMPACT\_MINUTIA\_ANGLE\_MASK** = 0x3F
- static const uint16\_t **FMD\_MIN\_MINUTIA\_QUALITY** = 0
- static const uint16\_t **FMD\_MAX\_MINUTIA\_QUALITY** = 100
- static const uint16\_t **FMD\_UNKNOWN\_MINUTIA\_QUALITY** = 0
- static const uint16\_t **FMD\_MIN\_MINUTIA\_ANGLE** = 0
- static const uint16\_t **FMD\_MAX\_MINUTIA\_ANGLE** = 179
- static const uint16\_t **FMD\_MAX\_MINUTIA\_ISONC\_ANGLE** = 255
- static const uint16\_t **FMD\_MAX\_MINUTIA\_ISOCC\_ANGLE** = 63
- static const uint16\_t **FMD\_ANSI\_ANGLE\_UNIT** = 2
- static const uint16\_t **FMD\_ISO\_ANGLE\_UNIT**
- static const uint16\_t **FMD\_ISOCC\_ANGLE\_UNIT**
- static const uint16\_t **FMD\_MINUTIA\_TYPE\_OTHER** = 0
- static const uint16\_t **FMD\_MINUTIA\_TYPE\_RIDGE\_ENDING** = 1
- static const uint16\_t **FMD\_MINUTIA\_TYPE\_BIFURCATION** = 2
- static const uint16\_t **FMR\_MIN\_FINGER\_QUALITY** = 0
- static const uint16\_t **FMR\_MAX\_FINGER\_QUALITY** = 100
- static const uint16\_t **ISO\_UNKNOWN\_FINGER\_QUALITY** = 0
- static const uint16\_t **FED\_RESERVED** = 0x0000
- static const uint16\_t **FED\_RIDGE\_COUNT** = 0x0001
- static const uint16\_t **FED\_CORE\_AND\_DELTA** = 0x0002
- static const uint16\_t **RCE\_NONSPECIFIC** = 0x00
- static const uint16\_t **RCE\_FOUR\_NEIGHBOR** = 0x01
- static const uint16\_t **RCE\_EIGHT\_NEIGHBOR** = 0x02
- static const uint16\_t **CORE\_TYPE\_NONANGULAR** = 0x00
- static const uint16\_t **CORE\_TYPE\_ANGULAR** = 0x01
- static const uint16\_t **DELTA\_TYPE\_NONANGULAR** = 0x00
- static const uint16\_t **DELTA\_TYPE\_ANGULAR** = 0x01

### G.64.1 Detailed Description

A class to represent a set of minutiae in an ANSI/INCITS record.

The base INCTISMinutiae class is responsible for reading minutiae data points and extended data. Each minutiae point, ridge count item, core, and delta is represented in the native ANSI/INCITS format. Objects of this base class cannot be instantiated, but rather derived classes are used to represent minutiae data taken from the INCITS-derived record formats.

## G.64.2 Constructor & Destructor Documentation

### G.64.2.1 INCITSMinutiae()

```
BiometricEvaluation::Feature::INCITSMinutiae::INCITSMinutiae (
 const MinutiaPointSet & mps,
 const RidgeCountItemSet & rcis,
 const CorePointSet & cps,
 const DeltaPointSet & dps)
```

Construct an INCITS **Minutiae** (p. 442) object from its components.

The buffer index must be set to the location in the buffer to start reading minutiae data points and extended data.

Parameters

|    |             |                               |
|----|-------------|-------------------------------|
| in | <i>mps</i>  | The set of minutiae points.   |
| in | <i>rcis</i> | The set of ridge count items. |
| in | <i>cps</i>  | The set of core points.       |
| in | <i>dps</i>  | The set of delta points.      |

## G.64.3 Member Function Documentation

### G.64.3.1 setCorePointSet()

```
void BiometricEvaluation::Feature::INCITSMinutiae::setCorePointSet (
 const CorePointSet & cps)
```

Mutator for the set of core points.

Parameters

|    |            |                         |
|----|------------|-------------------------|
| in | <i>cps</i> | The set of core points. |
|----|------------|-------------------------|

### G.64.3.2 setDeltaPointSet()

```
void BiometricEvaluation::Feature::INCITSMinutiae::setDeltaPointSet (
 const DeltaPointSet & dps)
```

Mutator for the set of delta points.

Parameters

|    |            |                               |
|----|------------|-------------------------------|
| in | <i>dps</i> | The set of delta point items. |
|----|------------|-------------------------------|

**G.64.3.3 setMinutiaPoints()**

```
void BiometricEvaluation::Feature::INCITSMinutiae::setMinutiaPoints (
 const MinutiaPointSet & mps)
```

Mutator for the minutiae point set.

Parameters

|    |            |                      |
|----|------------|----------------------|
| in | <i>mps</i> | The minutiae points. |
|----|------------|----------------------|

**G.64.3.4 setRidgeCountItems()**

```
void BiometricEvaluation::Feature::INCITSMinutiae::setRidgeCountItems (
 const RidgeCountItemSet & rcis)
```

Mutator for the ridge count items.

Parameters

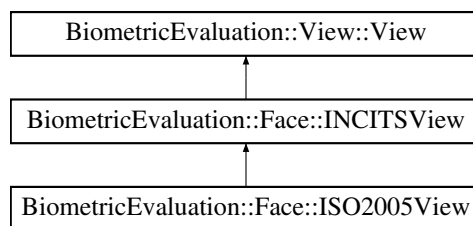
|    |             |                               |
|----|-------------|-------------------------------|
| in | <i>rcis</i> | The set of ridge count items. |
|----|-------------|-------------------------------|

**G.65 BiometricEvaluation::Face::INCITSView Class Reference**

A class to represent single facial image view and derived information.

```
#include <be_face_incitsview.h>
```

Inheritance diagram for BiometricEvaluation::Face::INCITSView:

**Public Member Functions**

- **Face::Gender** **getGender** () const  
*Obtain the gender.*
- **Face::EyeColor** **getEyeColor** () const  
*Obtain the eye color.*
- **Face::HairColor** **getHairColor** () const  
*Obtain the hair color.*
- bool **propertiesConsidered** () const  
*Indicate whether properties are specified.*
- void **getPropertySet** ( **Face::PropertySet** &propertySet) const

- Get the set of properties.*
- **BiometricEvaluation::Face::Expression** **getExpression** () const
- void **getFeaturePointSet** (BiometricEvaluation::Feature::MPEGFacePointSet &featurePointSet) const
- Obtain the set of.*
- **Face::ImageType** **getImageType** () const
- Obtain the face image type.*
- **Face::ImageDataType** **getImageDataType** () const
- Obtain the face image data type.*
- **Face::PoseAngle** **getPoseAngle** () const
- Obtain the face pose angle.*
- **Face::ColorSpace** **getColorSpace** () const
- Obtain the color space.*
- **Face::SourceType** **getSourceType** () const
- Obtain the source type.*
- uint16\_t **getDeviceType** () const
- Obtain the device type.*

## Protected Member Functions

- **INCITSView** (const std::string &filename, const uint32\_t viewNumber)
- Construct the common components of an INCITS face view from records contained in files.*
- **INCITSView** (const **Memory::uint8Array** &buffer, const uint32\_t viewNumber)
- Construct an INCITS face view from a record contained in a buffer.*
- **Memory::uint8Array** const & **getFIDData** () const
- Obtain a reference to the face image record data buffer.*
- virtual void **readHeader** ( **BiometricEvaluation::Memory::IndexedBuffer** &buf, const uint32\_t formatStandard)
- Read the common face image data record header from an INCITS record, excepting the format identifier and version number data items.*
- virtual void **readFaceView** ( **Memory::IndexedBuffer** &buf)
- Read the common face representation information from an INCITS record.*

## Static Protected Attributes

- static const uint32\_t **ISO2005\_STANDARD** = 1
- static const uint32\_t **BASE\_FORMAT\_ID** = 0x46414300

## G.65.1 Detailed Description

A class to represent single facial image view and derived information.

A base **Face::INCITSView** (p. 371) class represents an INCITS/ANSI or ISO face view. This class defines the common interface for all ANSI/ISO views as well as common implementations. Subclasses specialize this class in order to represent other versions of the ANSI/ISO specs. Objects of this class cannot be created.

## G.65.2 Constructor & Destructor Documentation

## G.65.2.1 INCITSView() [1/2]

```
BiometricEvaluation::Face::INCITSView::INCITSView (
 const std::string & filename,
 const uint32_t viewNumber) [protected]
```

Construct the common components of an INCITS face view from records contained in files.

See documentation in child classes of INCITS for information on constructing INCITS-derived face views.

## Parameters

|    |                   |                                                                      |
|----|-------------------|----------------------------------------------------------------------|
| in | <i>filename</i>   | The name of the file containing the complete face image data record. |
| in | <i>viewNumber</i> | The eye number to use.                                               |

## Exceptions

|                                  |                                   |
|----------------------------------|-----------------------------------|
| <b>Error::DataError</b> (p. 295) | Invalid record format.            |
| <b>Error::FileError</b> (p. 316) | Could not open or read from file. |

## G.65.2.2 INCITSView() [2/2]

```
BiometricEvaluation::Face::INCITSView::INCITSView (
 const Memory::uint8Array & buffer,
 const uint32_t viewNumber) [protected]
```

Construct an INCITS face view from a record contained in a buffer.

See documentation in child classes of INCITS for information on constructing INCITS-derived face views.

## Parameters

|    |                   |                                                            |
|----|-------------------|------------------------------------------------------------|
| in | <i>buffer</i>     | The buffer containing the complete face image data record. |
| in | <i>viewNumber</i> | The eye number to use.                                     |

## Exceptions

|                                  |                        |
|----------------------------------|------------------------|
| <b>Error::DataError</b> (p. 295) | Invalid record format. |
|----------------------------------|------------------------|

## G.65.3 Member Function Documentation

## G.65.3.1 getColorSpace()

```
Face::ColorSpace BiometricEvaluation::Face::INCITSView::getColorSpace () const
```

Obtain the color space.

Returns

The color space code.

### G.65.3.2 getDeviceType()

```
uint16_t BiometricEvaluation::Face::INCITSView::getDeviceType () const
```

Obtain the device type.

Returns

The device type vendor code.

### G.65.3.3 getEyeColor()

```
Face::EyeColor BiometricEvaluation::Face::INCITSView::getEyeColor () const
```

Obtain the eye color.

Returns

The eye color code.

### G.65.3.4 getFeaturePointSet()

```
void BiometricEvaluation::Face::INCITSView::getFeaturePointSet (
 BiometricEvaluation::Feature::MPEGFacePointSet & featurePointSet) const
```

Obtain the set of.

Parameters

|     |                        |                            |
|-----|------------------------|----------------------------|
| out | <i>featurePointSet</i> | The set of feature points. |
|-----|------------------------|----------------------------|

### G.65.3.5 getFIDData()

```
Memory::uint8Array const& BiometricEvaluation::Face::INCITSView::getFIDData () const [protected]
```

Obtain a reference to the face image record data buffer.

Returns

The entire face image record data.

### G.65.3.6 getGender()

```
Face::Gender BiometricEvaluation::Face::INCITSView::getGender () const
```

Obtain the gender.



Returns

The gender code.

### G.65.3.7 getHairColor()

**Face::HairColor** BiometricEvaluation::Face::INCITSView::getHairColor ( ) const

Obtain the hair color.

Returns

The hair color code.

### G.65.3.8 getImageDataType()

**Face::ImageDataType** BiometricEvaluation::Face::INCITSView::getImageDataType ( ) const

Obtain the face image data type.

Returns

The image data type.

### G.65.3.9 getImageType()

**Face::ImageType** BiometricEvaluation::Face::INCITSView::getImageType ( ) const

Obtain the face image type.

Returns

The image type.

### G.65.3.10 getPoseAngle()

**Face::PoseAngle** BiometricEvaluation::Face::INCITSView::getPoseAngle ( ) const

Obtain the face pose angle.

Returns

The pose angle.

### G.65.3.11 getPropertySet()

void BiometricEvaluation::Face::INCITSView::getPropertySet (

**Face::PropertySet** & *propertySet* ) const

Get the set of properties.

Returns

The set of properties.

**G.65.3.12 getSourceType()**

**Face::SourceType** BiometricEvaluation::Face::INCITSView::getSourceType ( ) const

Obtain the source type.

Returns

The source type code.

**G.65.3.13 propertiesConsidered()**

bool BiometricEvaluation::Face::INCITSView::propertiesConsidered ( ) const

Indicate whether properties are specified.

Returns

true if properties are specified, false otherwise.

**G.65.3.14 readFaceView()**

virtual void BiometricEvaluation::Face::INCITSView::readFaceView (   
 **Memory::IndexedBuffer** & buf ) [protected], [virtual]

Read the common face representation information from an INCITS record.

An **Face** (p. 109) representation from an INCITS record includes image information, gender, pose angle, etc.

Parameters

|         |     |                                                                                                                                             |
|---------|-----|---------------------------------------------------------------------------------------------------------------------------------------------|
| in, out | buf | The indexed buffer containing the record data. The index of the buffer will be changed to the location after the Facial information record. |
|---------|-----|---------------------------------------------------------------------------------------------------------------------------------------------|

Exceptions

|                  |                                                |
|------------------|------------------------------------------------|
| <i>DataError</i> | The INCITS record has invalid or missing data. |
|------------------|------------------------------------------------|

**G.65.3.15 readHeader()**

virtual void BiometricEvaluation::Face::INCITSView::readHeader (   
 **BiometricEvaluation::Memory::IndexedBuffer** & buf,   
 const uint32\_t formatStandard ) [protected], [virtual]

Read the common face image data record header from an INCITS record, excepting the format identifier and version number data items.

Parameters

|    |     |                                                                                                                                                                                                                                |
|----|-----|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| in | buf | The indexed buffer containing the record data, with the index starting at the first octet after the format identifier and version number data items. The index of the buffer will be changed to the location after the header. |
|----|-----|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

## Parameters

|    |                       |                                                                         |
|----|-----------------------|-------------------------------------------------------------------------|
| in | <i>formatStandard</i> | Value indicating which header version to read; must be ISO2005.STANDARD |
|----|-----------------------|-------------------------------------------------------------------------|

## Exceptions

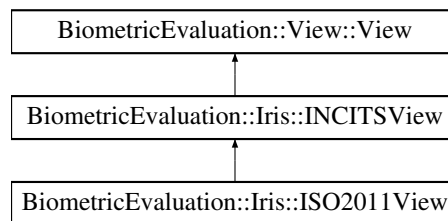
|                       |                                                |
|-----------------------|------------------------------------------------|
| <i>ParameterError</i> | The formatStandard parameter is incorrect.     |
| <i>DataError</i>      | The INCITS record has invalid or missing data. |

## G.66 BiometricEvaluation::Iris::INCITSView Class Reference

A class to represent single iris view and derived information.

```
#include <be_iris_incitsview.h>
```

Inheritance diagram for BiometricEvaluation::Iris::INCITSView:



### Classes

- struct **QualitySubBlock**  
*Representation of an iris quality block.*

### Public Types

- typedef std::vector< **QualitySubBlock** > **QualitySet**

### Public Member Functions

- uint8\_t **getCertificationFlag** () const  
*Obtain the certification flag.*
- std::string **getCaptureDateString** () const  
*Obtain the capture date as a string.*
- **Iris::CaptureDeviceTechnology** **getCaptureDeviceTechnology** () const  
*Obtain the capture device technology.*
- uint16\_t **getCaptureDeviceVendor** () const  
*Obtain the capture device vendor.*
- uint16\_t **getCaptureDeviceType** () const  
*Obtain the capture device type.*
- void **getQualitySet** (Iris::INCITSView::QualitySet &qualitySet) const

- Obtain the set of quality sub-blocks.*
- **Iris::EyeLabel** **getEyeLabel** () const  
*Obtain the eye label type.*
- **Iris::ImageType** **getImageType** () const  
*Obtain the iris image type.*
- void **getImageProperties** ( **BiometricEvaluation::Iris::Orientation** &horizontalOrientation, **BiometricEvaluation::Iris::Orientation** &verticalOrientation, **BiometricEvaluation::Iris::ImageCompression** &compressionHistory) const  
*Obtain the iris image properties.*
- uint16\_t **getCameraRange** ()  
*Obtain the camera range.*
- void **getRollAngleInfo** (uint16\_t &rollAngle, uint16\_t &rollAngleUncertainty)  
*Obtain the roll angle information.*
- void **getIrisCenterInfo** (uint16\_t &irisCenterSmallestX, uint16\_t &irisCenterSmallestY, uint16\_t &irisCenterLargestX, uint16\_t &irisCenterLargestY, uint16\_t &irisDiameterSmallest, uint16\_t &irisDiameterLargest)  
*Obtain the iris center information. COORDINATE\_UNDEF may be returned for any of the out parameters.*

### Static Public Attributes

- static const uint16\_t **RANGE\_UNASSIGNED** = 0
- static const uint16\_t **RANGE\_FAILED** = 1
- static const uint16\_t **RANGE\_OVERFLOW** = 65535
- static const uint16\_t **ROLL\_ANGLE\_UNDEF** = 65535
- static const uint16\_t **ROLL\_UNCERTAIN\_UNDEF** = 65535
- static const uint16\_t **COORDINATE\_UNDEF** = 0

### Protected Member Functions

- **INCITSView** (const std::string &filename, const uint32\_t viewNumber)  
*Construct the common components of an INCITS iris view from records contained in files.*
- **INCITSView** (const **Memory::uint8Array** &buffer, const uint32\_t viewNumber)  
*Construct an INCITS iris view from a record contained in a buffer.*
- **Memory::uint8Array** const & **getIIRData** () const  
*Obtain a reference to the iris image record data buffer.*
- virtual void **readHeader** ( **BiometricEvaluation::Memory::IndexedBuffer** &buf, const uint32\_t formatStandard)  
*Read the common iris image record header from an INCITS record, excepting the format identifier and version number data items.*
- virtual void **readIrisView** ( **Memory::IndexedBuffer** &buf)  
*Read the common iris representation information from an INCITS record.*

### Static Protected Attributes

- static const uint32\_t **ISO2011\_STANDARD** = 1
- static const uint32\_t **BASE\_FORMAT\_ID** = 0x49495200
- static const uint8\_t **CAPTURE\_DATE\_LENGTH** = 9

## G.66.1 Detailed Description

A class to represent single iris view and derived information.

A base **Iris::INCITSView** (p. 377) class represents an INCITS/ANSI or ISO iris view. This class defines the common interface for all ANSI/ISO views as well as common implementations. Subclasses specialize this class in order to represent other versions of the ANSI/ISO specs. Objects of this class cannot be created.

## G.66.2 Constructor & Destructor Documentation

### G.66.2.1 INCITSView() [1/2]

```
BiometricEvaluation::Iris::INCITSView::INCITSView (
 const std::string & filename,
 const uint32_t viewNumber) [protected]
```

Construct the common components of an INCITS iris view from records contained in files.

See documentation in child classes of INCITS for information on constructing INCITS-derived iris views.

Parameters

|    |                   |                                                                 |
|----|-------------------|-----------------------------------------------------------------|
| in | <i>filename</i>   | The name of the file containing the complete iris image record. |
| in | <i>viewNumber</i> | The eye number to use.                                          |

Exceptions

|                                  |                                   |
|----------------------------------|-----------------------------------|
| <b>Error::DataError</b> (p. 295) | Invalid record format.            |
| <b>Error::FileError</b> (p. 316) | Could not open or read from file. |

### G.66.2.2 INCITSView() [2/2]

```
BiometricEvaluation::Iris::INCITSView::INCITSView (
 const Memory::uint8Array & buffer,
 const uint32_t viewNumber) [protected]
```

Construct an INCITS iris view from a record contained in a buffer.

See documentation in child classes of INCITS for information on constructing INCITS-derived iris views.

Parameters

|    |                   |                                                       |
|----|-------------------|-------------------------------------------------------|
| in | <i>buffer</i>     | The buffer containing the complete iris image record. |
| in | <i>viewNumber</i> | The eye number to use.                                |

Exceptions

|                                  |                        |
|----------------------------------|------------------------|
| <b>Error::DataError</b> (p. 295) | Invalid record format. |
|----------------------------------|------------------------|

## G.66.3 Member Function Documentation

### G.66.3.1 getCameraRange()

```
uint16_t BiometricEvaluation::Iris::INCITSView::getCameraRange ()
```

Obtain the camera range.

RANGE\_UNASSIGNED, RANGE\_FAILED, or RANGE\_OVERFLOW may be returned.

Returns

The camera range.

### G.66.3.2 getCaptureDateString()

```
std::string BiometricEvaluation::Iris::INCITSView::getCaptureDateString () const
```

Obtain the capture date as a string.

Returns

The capture data and time.

### G.66.3.3 getCaptureDeviceTechnology()

```
Iris::CaptureDeviceTechnology BiometricEvaluation::Iris::INCITSView::getCaptureDeviceTechnology
() const
```

Obtain the capture device technology.

Returns

The capture device technology identifier.

### G.66.3.4 getCaptureDeviceType()

```
uint16_t BiometricEvaluation::Iris::INCITSView::getCaptureDeviceType () const
```

Obtain the capture device type.

Returns

The capture device type ID.

### G.66.3.5 getCaptureDeviceVendor()

```
uint16_t BiometricEvaluation::Iris::INCITSView::getCaptureDeviceVendor () const
```

Obtain the capture device vendor.

Returns

The capture device vendor ID.

### G.66.3.6 getCertificationFlag()

`uint8_t BiometricEvaluation::Iris::INCITSView::getCertificationFlag ( ) const`  
Obtain the certification flag.

Returns

The certification flag.

### G.66.3.7 getEyeLabel()

`Iris::EyeLabel BiometricEvaluation::Iris::INCITSView::getEyeLabel ( ) const`  
Obtain the eye label type.

Returns

The eye label.

### G.66.3.8 getIIRData()

`Memory::uint8Array const& BiometricEvaluation::Iris::INCITSView::getIIRData ( ) const [protected]`  
Obtain a reference to the iris image record data buffer.

Returns

The entire iris image record data.

### G.66.3.9 getImageProperties()

`void BiometricEvaluation::Iris::INCITSView::getImageProperties (`  
    `BiometricEvaluation::Iris::Orientation & horizontalOrientation,`  
    `BiometricEvaluation::Iris::Orientation & verticalOrientation,`  
    `BiometricEvaluation::Iris::ImageCompression & compressionHistory ) const`  
Obtain the iris image properties.

Parameters

|     |                              |                                |
|-----|------------------------------|--------------------------------|
| out | <i>horizontalOrientation</i> | The horizontal orientation.    |
| out | <i>verticalOrientation</i>   | The vertical orientation.      |
| out | <i>compressionHistory</i>    | The image compression history. |

### G.66.3.10 getImageType()

`Iris::ImageType BiometricEvaluation::Iris::INCITSView::getImageType ( ) const`  
Obtain the iris image type.

## Returns

The image type.

**G.66.3.11 getIrisCenterInfo()**

```
void BiometricEvaluation::Iris::INCITSView::getIrisCenterInfo (
 uint16_t & irisCenterSmallestX,
 uint16_t & irisCenterSmallestY,
 uint16_t & irisCenterLargestX,
 uint16_t & irisCenterLargestY,
 uint16_t & irisDiameterSmallest,
 uint16_t & irisDiameterLargest)
```

Obtain the iris center information. COORDINATE\_UNDEF may be returned for any of the out parameters.

## Parameters

|     |                             |                                                       |
|-----|-----------------------------|-------------------------------------------------------|
| out | <i>irisCenterSmallestX</i>  | Smallest expected iris center X coordinate in pixels. |
| out | <i>irisCenterSmallestY</i>  | Smallest expected iris center Y coordinate in pixels. |
| out | <i>irisCenterLargestX</i>   | Largest expected iris center X coordinate in pixels.  |
| out | <i>irisCenterLargestY</i>   | Largest expected iris center Y coordinate in pixels.  |
| out | <i>irisDiameterSmallest</i> | Smallest expected iris diameter in pixels.            |
| out | <i>irisDiameterLargest</i>  | Largest expected iris diameter in pixels.             |

**G.66.3.12 getQualitySet()**

```
void BiometricEvaluation::Iris::INCITSView::getQualitySet (
 Iris::INCITSView::QualitySet & qualitySet) const
```

Obtain the set of quality sub-blocks.

## Parameters

|     |                   |                                |
|-----|-------------------|--------------------------------|
| out | <i>qualitySet</i> | The set of quality sub-blocks. |
|-----|-------------------|--------------------------------|

**G.66.3.13 getRollAngleInfo()**

```
void BiometricEvaluation::Iris::INCITSView::getRollAngleInfo (
 uint16_t & rollAngle,
 uint16_t & rollAngleUncertainty)
```

Obtain the roll angle information.

## Parameters

|     |                             |                             |
|-----|-----------------------------|-----------------------------|
| out | <i>rollAngle</i>            | The roll angle.             |
| out | <i>rollAngleUncertainty</i> | The roll angle uncertainty. |



G.66.3.14 readHeader()

```
virtual void BiometricEvaluation::Iris::INCITSView::readHeader (
 BiometricEvaluation::Memory::IndexedBuffer & buf,
 const uint32_t formatStandard) [protected], [virtual]
```

Read the common iris image record header from an INCITS record, excepting the format identifier and version number data items.

Parameters

|    |                       |                                                                                                                                                                                                                                |
|----|-----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| in | <i>buf</i>            | The indexed buffer containing the record data, with the index starting at the first octet after the format identifier and version number data items. The index of the buffer will be changed to the location after the header. |
| in | <i>formatStandard</i> | Value indicating which header version to read; must be ISO2011_STANDARD                                                                                                                                                        |

Exceptions

|                       |                                                |
|-----------------------|------------------------------------------------|
| <i>ParameterError</i> | The specVersion parameter is incorrect.        |
| <i>DataError</i>      | The INCITS record has invalid or missing data. |

G.66.3.15 readIrisView()

```
virtual void BiometricEvaluation::Iris::INCITSView::readIrisView (
 Memory::IndexedBuffer & buf) [protected], [virtual]
```

Read the common iris representation information from an INCITS record.  
An **Iris** (p. 138) Representation from an INCITS record includes image information, cropping information, etc.

Parameters

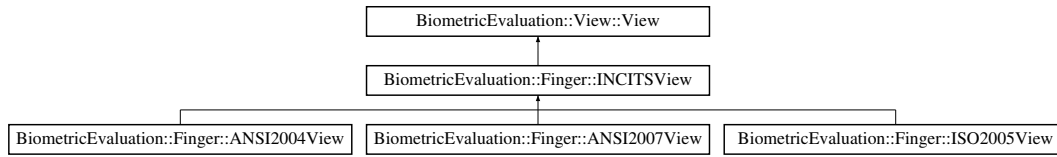
|         |            |                                                                                                                                                       |
|---------|------------|-------------------------------------------------------------------------------------------------------------------------------------------------------|
| in, out | <i>buf</i> | The indexed buffer containing the record data. The index of the buffer will be changed to the location after the <b>Iris</b> (p. 138) Representation. |
|---------|------------|-------------------------------------------------------------------------------------------------------------------------------------------------------|

Exceptions

|                  |                                                |
|------------------|------------------------------------------------|
| <i>DataError</i> | The INCITS record has invalid or missing data. |
|------------------|------------------------------------------------|

G.67 BiometricEvaluation::Finger::INCITSView Class Reference

A class to represent single finger view and derived information.  
#include <be\_finger\_incitsview.h>  
Inheritance diagram for BiometricEvaluation::Finger::INCITSView:



## Public Member Functions

- **Feature::INCITSMinutiae** **getMinutiaeData** () const  
*Obtain the set of minutiae records.*
- **Finger::Position** **getPosition** () const  
*Obtain the finger position.*
- **Finger::Impression** **getImpressionType** () const  
*Obtain the finger impression code.*
- uint32\_t **getQuality** () const  
*Obtain the finger quality value.*
- uint16\_t **getCaptureEquipmentID** () const  
*Obtain the capture equipment identifier.*
- bool **isAppendixFCompliant** () const  
*Obtain the capture equipment compliance indicator for 'Appendix F'.*
- uint16\_t **getProductIDOwner** () const  
*Obtain the CBEFF product identifier owner.*
- uint16\_t **getProductIDType** () const  
*Obtain the CBEFF product identifier type.*
- uint32\_t **getRecordLength** () const
- uint8\_t **getNumFingerViews** () const
- uint8\_t **getFMRReservedByte** () const
- uint32\_t **getViewNumber** () const
- uint16\_t **getEDBLength** () const
- std::vector< uint8\_t > **getMinutiaeReservedData** () const
- void **setMinutiaeData** (const **Feature::INCITSMinutiae** &fmd)  
*Mutator for the **Feature::INCITSMinutiae** (p. 368) item.*
- void **setMinutiaeReservedData** (const std::vector< uint8\_t > &reservedBits)  
*Mutator for the FMD reserved bits vector.*

## Static Public Member Functions

- static **Finger::Position** **convertPosition** (int incitsFGP)  
*Convert a finger position code from an INCITS finger record to the common code.*
- static **Finger::Impression** **convertImpression** (int incitsIMP)  
*Convert a impression type code from an INCITS finger record to the common code.*

## Protected Member Functions

- **INCITSView** (const std::string &fmrFilename, const std::string &firFilename, const uint32\_t view↵  
Number)  
*Construct the common components of an INCITS finger view from records contained in files.*
- **INCITSView** (const **Memory::uint8Array** &fmrBuffer, const **Memory::uint8Array** &firBuffer,  
const uint32\_t viewNumber)  
*Construct an INCITS finger view from records contained in buffers.*
- **Memory::uint8Array** const & **getFMRData** () const  
*Obtain a reference to the finger minutiae record data buffer.*
- **Memory::uint8Array** const & **getFIRData** () const  
*Obtain a reference to the finger image record data buffer.*
- void **setPosition** (const **Finger::Position** &position)  
*Mutator for the position.*
- void **setImpressionType** (const **Finger::Impression** &impression)  
*Mutator for the impression type.*
- void **setQuality** (uint32\_t quality)  
*Mutator for the finger quality value.*
- void **setViewNumber** (uint32\_t viewNumber)  
*Mutator for the finger view number.*
- void **setCaptureEquipmentID** (uint16\_t id)  
*Mutator for the equipment ID.*
- void **setCBEFFProductIDs** (uint16\_t owner, uint16\_t type)  
*Mutator for the CBEFF Product ID owner and type.*
- void **setAppendixFCompliance** (bool flag)  
*Mutator for the Appendix F compliance indicator.*
- void **readFMRHeader** ( **Memory::IndexedBuffer** &buf, const uint32\_t formatStandard)  
*Read the common finger minutiae record header from an INCITS record.*
- void **readFVMR** ( **Memory::IndexedBuffer** &buf)  
*Read the common finger view record information from an INCITS record.*
- virtual std::tuple< Feature::MinutiaPointSet, std::vector< uint8\_t > > **readMinutiaeDataPoints** ( **Memory↵  
::IndexedBuffer** &buf, uint32\_t count)  
*Read the minutiae data points, and extended data blocks.*
- virtual void **readExtendedDataBlock** ( **Memory::IndexedBuffer** &buf)  
*Read the common extended data block.*
- virtual Feature::RidgeCountItemSet **readRidgeCountData** ( **Memory::IndexedBuffer** &buf, uint32↵  
\_t dataLength)  
*Read the ridge count data.*
- virtual void **readCoreDeltaData** ( **Memory::IndexedBuffer** &buf, uint32\_t dataLength, Feature::↵  
CorePointSet &cores, Feature::DeltaPointSet &deltas)=0  
*Read the core points data.*

## Static Protected Attributes

- static const uint32\_t **FMR\_BASE\_FORMAT\_ID** = 0x464D5200
- static const uint32\_t **ANSI2004\_STANDARD** = 1  
*The type of record that will be read by the subclass.*
- static const uint32\_t **ISO2005\_STANDARD** = 2
- static const uint32\_t **ANSI2007\_STANDARD** = 3

### G.67.1 Detailed Description

A class to represent single finger view and derived information.

A base **Finger::INCITSView** (p. 383) object represents an INCITS/ANSI or ISO finger view. This class defines the common interface for all ANSI/ISO views as well as common implementations. Subclasses specialize this class in order to represent other versions of the ANSI/ISO specs. Objects of this class cannot be created.

### G.67.2 Constructor & Destructor Documentation

#### G.67.2.1 INCITSView() [1/2]

```
BiometricEvaluation::Finger::INCITSView::INCITSView (
 const std::string & fmrFilename,
 const std::string & firFilename,
 const uint32_t viewNumber) [protected]
```

Construct the common components of an INCITS finger view from records contained in files.

See documentation in child classes of INCITS for information on constructing INCITS-derived finger views.

Parameters

|    |                    |                                                                      |
|----|--------------------|----------------------------------------------------------------------|
| in | <i>fmrFilename</i> | The name of the file containing the complete finger minutiae record. |
| in | <i>firFilename</i> | The name of the file containing the complete finger image record.    |
| in | <i>viewNumber</i>  | The finger view number to use.                                       |

Exceptions

|                                  |                                   |
|----------------------------------|-----------------------------------|
| <b>Error::DataError</b> (p. 295) | Invalid record format.            |
| <b>Error::FileError</b> (p. 316) | Could not open or read from file. |

#### G.67.2.2 INCITSView() [2/2]

```
BiometricEvaluation::Finger::INCITSView::INCITSView (
 const Memory::uint8Array & fmrBuffer,
 const Memory::uint8Array & firBuffer,
 const uint32_t viewNumber) [protected]
```

Construct an INCITS finger view from records contained in buffers.

See documentation in child classes of INCITS for information on constructing INCITS-derived finger views.

Parameters

|    |                   |                                                            |
|----|-------------------|------------------------------------------------------------|
| in | <i>fmrBuffer</i>  | The buffer containing the complete finger minutiae record. |
| in | <i>firBuffer</i>  | The buffer containing the complete finger image record.    |
| in | <i>viewNumber</i> | The finger view number to use.                             |

## Exceptions

|                                  |                        |
|----------------------------------|------------------------|
| <b>Error::DataError</b> (p. 295) | Invalid record format. |
|----------------------------------|------------------------|

## G.67.3 Member Function Documentation

### G.67.3.1 convertImpression()

```
static Finger::Impression BiometricEvaluation::Finger::INCITSView::convertImpression (
 int incitsIMP) [static]
```

Convert a impression type code from an INCITS finger record to the common code.

## Parameters

|    |                  |                                                                  |
|----|------------------|------------------------------------------------------------------|
| in | <i>incitsIMP</i> | A finger impression type code as defined by the INCITS standard. |
|----|------------------|------------------------------------------------------------------|

## Exceptions

|                                  |                                      |
|----------------------------------|--------------------------------------|
| <b>Error::DataError</b> (p. 295) | The impression type code is invalid. |
|----------------------------------|--------------------------------------|

## Returns

The finger impression type code in common notation.

### G.67.3.2 convertPosition()

```
static Finger::Position BiometricEvaluation::Finger::INCITSView::convertPosition (
 int incitsFGP) [static]
```

Convert a finger postion code from an INCITS finger record to the common code.

## Parameters

|    |                  |                                                           |
|----|------------------|-----------------------------------------------------------|
| in | <i>incitsFGP</i> | A finger position code as defined by the INCITS standard. |
|----|------------------|-----------------------------------------------------------|

## Exceptions

|                                  |                               |
|----------------------------------|-------------------------------|
| <b>Error::DataError</b> (p. 295) | The position code is invalid. |
|----------------------------------|-------------------------------|

## Returns

The finger position code in common notation.

### G.67.3.3 getCaptureEquipmentID()

```
uint16_t BiometricEvaluation::Finger::INCITSView::getCaptureEquipmentID () const
```

Obtain the capture equipment identifier.

Returns

The equipment ID.

### G.67.3.4 getEDBLength()

```
uint16_t BiometricEvaluation::Finger::INCITSView::getEDBLength () const
```

Returns

Length of extended data block, as recorded in the record.

### G.67.3.5 getFIRData()

```
Memory::uint8Array const& BiometricEvaluation::Finger::INCITSView::getFIRData () const [protected]
```

Obtain a reference to the finger image record data buffer.

Returns

The entire finger image record data.

### G.67.3.6 getFMRData()

```
Memory::uint8Array const& BiometricEvaluation::Finger::INCITSView::getFMRData () const [protected]
```

Obtain a reference to the finger minutiae record data buffer.

Returns

The entire finger minutiae record data.

### G.67.3.7 getFMRReservedByte()

```
uint8_t BiometricEvaluation::Finger::INCITSView::getFMRReservedByte () const
```

Returns

Reserved byte from FMR header.

### G.67.3.8 getImpressionType()

```
Finger::Impression BiometricEvaluation::Finger::INCITSView::getImpressionType () const
```

Obtain the finger impression code.

Returns

The finger impression code.

**G.67.3.9 getMinutiaeReservedData()**

```
std::vector<uint8_t> BiometricEvaluation::Finger::INCITSView::getMinutiaeReservedData () const
```

Returns

FMD reserved bits.

Note

Only lowest 2 bits are relevant.

**G.67.3.10 getNumFingerViews()**

```
uint8_t BiometricEvaluation::Finger::INCITSView::getNumFingerViews () const
```

Returns

Number of finger views, as recorded in the record.

**G.67.3.11 getPosition()**

```
Finger::Position BiometricEvaluation::Finger::INCITSView::getPosition () const
```

Obtain the finger position.

Returns

The finger position.

**G.67.3.12 getProductIDOwner()**

```
uint16_t BiometricEvaluation::Finger::INCITSView::getProductIDOwner () const [inline]
```

Obtain the CBEFF product identifier owner.

Returns

CBEFF product identifier owner.

**G.67.3.13 getProductIDType()**

```
uint16_t BiometricEvaluation::Finger::INCITSView::getProductIDType () const [inline]
```

Obtain the CBEFF product identifier type.

Returns

CBEFF product identifier type.

**G.67.3.14** `getQuality()`

```
uint32_t BiometricEvaluation::Finger::INCITSView::getQuality () const
```

Obtain the finger quality value.

Returns

The finger quality value.

**G.67.3.15** `getRecordLength()`

```
uint32_t BiometricEvaluation::Finger::INCITSView::getRecordLength () const
```

Returns

Length of record, as recorded in the record.

**G.67.3.16** `getViewNumber()`

```
uint32_t BiometricEvaluation::Finger::INCITSView::getViewNumber () const
```

Returns

**View** (p. 164) number, as recorded in the record.

**G.67.3.17** `isAppendixFCompliant()`

```
bool BiometricEvaluation::Finger::INCITSView::isAppendixFCompliant () const [inline]
```

Obtain the capture equipment compliance indicator for 'Appendix F'.

Returns

True if 'Appendix F' compliant, false otherwise.

**G.67.3.18** `readCoreDeltaData()`

```
virtual void BiometricEvaluation::Finger::INCITSView::readCoreDeltaData (
 Memory::IndexedBuffer & buf,
 uint32_t dataLength,
 Feature::CorePointSet & cores,
 Feature::DeltaPointSet & deltas) [protected], [pure virtual]
```

Read the core points data.

This method must be overridden by derived classes to read data in a specific record format.

Parameters

|         |                   |                                                                                                                                                    |
|---------|-------------------|----------------------------------------------------------------------------------------------------------------------------------------------------|
| in, out | <i>buf</i>        | The indexed buffer containing the record data. On function exit, the buffer index will be set to the location after the last core point data item. |
| out     | <i>cores</i>      | The set of core data items.                                                                                                                        |
| out     | <i>deltas</i>     | The set of delta data items.                                                                                                                       |
| in      | <i>dataLength</i> | The length of the entire ridge count data block.                                                                                                   |



Implemented in **BiometricEvaluation::Finger::ANSI2007View** (p. 216), **BiometricEvaluation::Finger::ISO2005View** (p. 404), and **BiometricEvaluation::Finger::ANSI2004View** (p. 214).

G.67.3.19 readExtendedDataBlock()

```
virtual void BiometricEvaluation::Finger::INCITSView::readExtendedDataBlock (
 Memory::IndexedBuffer & buf) [protected], [virtual]
```

Read the common extended data block.

Parameters

|         |     |                                                                                                                                       |
|---------|-----|---------------------------------------------------------------------------------------------------------------------------------------|
| in, out | buf | The indexed buffer containing the record data. The index of the buffer will be changed to the location after the extended data block. |
|---------|-----|---------------------------------------------------------------------------------------------------------------------------------------|

Exceptions

|           |                                                |
|-----------|------------------------------------------------|
| DataError | The INCITS record has invalid or missing data. |
|-----------|------------------------------------------------|

G.67.3.20 readFMRHeader()

```
void BiometricEvaluation::Finger::INCITSView::readFMRHeader (
 Memory::IndexedBuffer & buf,
 const uint32_t formatStandard) [protected]
```

Read the common finger minutiae record header from an INCITS record.  
For ANSI-2004 and ISO-2005 record formats, the finger minutiae record header is (mostly) the same.

Parameters

|    |                |                                                                                                                                                                                                          |
|----|----------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| in | buf            | The indexed buffer containing the record data. The index must start after the Format ID and spec version fields in the header. The index of the buffer will be changed to the location after the header. |
| in | formatStandard | Value indicating which header version to read; one of ANSI2004_STANDARD or ISO2005_STANDARD.                                                                                                             |

Exceptions

|                |                                                |
|----------------|------------------------------------------------|
| ParameterError | The specVersion parameter is incorrect.        |
| DataError      | The INCITS record has invalid or missing data. |

G.67.3.21 readFVMR()

```
void BiometricEvaluation::Finger::INCITSView::readFVMR (
 Memory::IndexedBuffer & buf) [protected]
```

Read the common finger view record information from an INCITS record.

A **Finger** (p. 115) **View** (p. 164) from an INCITS record includes image information, minutiae, and extended data ridge counts, cores/deltas, etc.) For ANSI-2004 and ISO-2005 record formats, the finger view representation is the same, so this functions parses those record formats. The minutiae data items are also read, as well as any extended data.

#### Parameters

|                |            |                                                                                                                                                            |
|----------------|------------|------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>in, out</i> | <i>buf</i> | The indexed buffer containing the record data. The index of the buffer will be changed to the location after the finger view, including the extended data. |
|----------------|------------|------------------------------------------------------------------------------------------------------------------------------------------------------------|

#### Exceptions

|                  |                                                |
|------------------|------------------------------------------------|
| <i>DataError</i> | The INCITS record has invalid or missing data. |
|------------------|------------------------------------------------|

### G.67.3.22 readMinutiaeDataPoints()

```
virtual std::tuple<Feature::MinutiaPointSet, std::vector<uint8_t> > BiometricEvaluation::Finger::INCITSView::readMinutiaeDataPoints (
 Memory::IndexedBuffer & buf,
 uint32_t count) [protected], [virtual]
```

Read the minutiae data points, and extended data blocks.

Function to be implemented by derived classes to read the minutiae data points and extended data block according to the specific standard they represent.

#### Parameters

|           |              |                                                                                                                                                            |
|-----------|--------------|------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>in</i> | <i>buf</i>   | The indexed buffer containing the record data. The index of the buffer will be changed to the location after the finger view, including the extended data. |
| <i>in</i> | <i>count</i> | Number of minutiae data points to read.                                                                                                                    |

#### Exceptions

|                  |                                                |
|------------------|------------------------------------------------|
| <i>DataError</i> | The INCITS record has invalid or missing data. |
|------------------|------------------------------------------------|

### G.67.3.23 readRidgeCountData()

```
virtual Feature::RidgeCountItemSet BiometricEvaluation::Finger::INCITSView::readRidgeCountData (
 Memory::IndexedBuffer & buf,
 uint32_t dataLength) [protected], [virtual]
```

Read the ridge count data.

This method reads data in the base INCITS format as defined in INCITS/ANSI 378-2004. This method may be overridden by derived classes to read data in a different record format.

## Parameters

|         |                   |                                                                                                                                                |
|---------|-------------------|------------------------------------------------------------------------------------------------------------------------------------------------|
| in, out | <i>buf</i>        | The indexed buffer containing the record data. On function exit, the buffer index will be set to the location after the last ridge count item. |
| in      | <i>dataLength</i> | The length of the entire ridge count data block.                                                                                               |

**G.67.3.24 setAppendixFCompliance()**

```
void BiometricEvaluation::Finger::INCITSView::setAppendixFCompliance (
 bool flag) [protected]
 Mutator for the Appendix F compliance indicator.
```

## Parameters

|    |             |                                                                        |
|----|-------------|------------------------------------------------------------------------|
| in | <i>flag</i> | True if the capture equipment is 'Appendix F' compliant, false if not. |
|----|-------------|------------------------------------------------------------------------|

**G.67.3.25 setCaptureEquipmentID()**

```
void BiometricEvaluation::Finger::INCITSView::setCaptureEquipmentID (
 uint16_t id) [protected]
 Mutator for the equipment ID.
```

## Parameters

|    |           |                         |
|----|-----------|-------------------------|
| in | <i>id</i> | The equipment ID value. |
|----|-----------|-------------------------|

**G.67.3.26 setCBEFFProductIDs()**

```
void BiometricEvaluation::Finger::INCITSView::setCBEFFProductIDs (
 uint16_t owner,
 uint16_t type) [protected]
 Mutator for the CBEFF Product ID owner and type.
```

## Parameters

|    |              |                                    |
|----|--------------|------------------------------------|
| in | <i>owner</i> | The CBEFF ID of the product owner. |
| in | <i>type</i>  | The CBEFF ID of the product type.  |

**G.67.3.27 setImpressionType()**

```
void BiometricEvaluation::Finger::INCITSView::setImpressionType (
 const Finger::Impression & impression) [protected]
```

Mutator for the impression type.

Parameters

|    |                   |                                  |
|----|-------------------|----------------------------------|
| in | <i>impression</i> | The finger impression type code. |
|----|-------------------|----------------------------------|

### G.67.3.28 setMinutiaeData()

```
void BiometricEvaluation::Finger::INCITSView::setMinutiaeData (
 const Feature::INCITSMinutiae & fmd)
```

Mutator for the **Feature::INCITSMinutiae** (p. 368) item.

Parameters

|    |            |                           |
|----|------------|---------------------------|
| in | <i>fmd</i> | The minutiae data object. |
|----|------------|---------------------------|

### G.67.3.29 setMinutiaeReservedData()

```
void BiometricEvaluation::Finger::INCITSView::setMinutiaeReservedData (
 const std::vector< uint8_t > & reservedBits)
```

Mutator for the FMD reserved bits vector.

Parameters

|    |                     |                         |
|----|---------------------|-------------------------|
| in | <i>reservedBits</i> | Reserved bits from FMD. |
|----|---------------------|-------------------------|

### G.67.3.30 setPosition()

```
void BiometricEvaluation::Finger::INCITSView::setPosition (
 const Finger::Position & position) [protected]
```

Mutator for the position.

Parameters

|    |                 |                      |
|----|-----------------|----------------------|
| in | <i>position</i> | The finger position. |
|----|-----------------|----------------------|

### G.67.3.31 setQuality()

```
void BiometricEvaluation::Finger::INCITSView::setQuality (
 uint32_t quality) [protected]
```

Mutator for the finger quality value.

Parameters

|    |                |                    |
|----|----------------|--------------------|
| in | <i>quality</i> | The quality value. |
|----|----------------|--------------------|

### G.67.3.32 setViewNumber()

```
void BiometricEvaluation::Finger::INCITSView::setViewNumber (
 uint32_t viewNumber) [protected]
```

Mutator for the finger view number.

Parameters

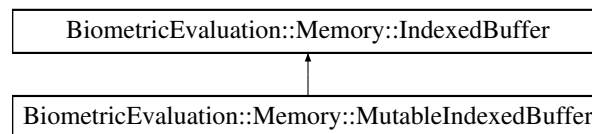
|    |                   |                        |
|----|-------------------|------------------------|
| in | <i>viewNumber</i> | The view number value. |
|----|-------------------|------------------------|

## G.68 BiometricEvaluation::Memory::IndexedBuffer Class Reference

Wrap a memory buffer with an index.

```
#include <be_memory_indexedbuffer.h>
```

Inheritance diagram for BiometricEvaluation::Memory::IndexedBuffer:



### Public Member Functions

- **IndexedBuffer** ()
- **IndexedBuffer** (const uint8\_t \*data, uint64\_t size)  
*Wrap an existing buffer of a given length.*
- **IndexedBuffer** (const **uint8Array** &aa)  
*Wrap an existing uint8Array.*
- **IndexedBuffer** (const **IndexedBuffer** &copy)=default
- uint32\_t **getSize** () const  
*Obtain the current size of the buffer.*
- uint32\_t **getIndex** () const  
*Obtain the current index into the buffer.*
- void **setIndex** (uint64\_t index)  
*Set the current index into the buffer.*
- uint8\_t **scanU8Val** ()  
*Obtain the next element of the buffer and increment the current index value.*
- uint16\_t **scanU16Val** ()  
*Obtain the next two elements of the buffer and increment the current index value.*

- `uint16_t scanBeU16Val ()`  
*Obtain the next two elements of the buffer, scanned as a big-endian value, and increment the current index value.*
- `uint32_t scanU32Val ()`  
*Obtain the next four elements of the buffer and increment the current index value by four.*
- `uint32_t scanBeU32Val ()`  
*Obtain the next four elements of the buffer, scanned as a big-endian value, and increment the current index value.*
- `uint64_t scanU64Val ()`  
*Obtain the next eight elements of the buffer and increment the current index value by eight.*
- `uint64_t scan (void *buf, uint64_t len)`  
*Obtain the next 'n' elements of the buffer and increment the current index value by n.*
- `virtual const uint8_t * get () const`  
*Returns a pointer to the managed buffer.*
- `virtual ~IndexedBuffer ()=default`

### G.68.1 Detailed Description

Wrap a memory buffer with an index.

The memory buffer is treated as an array of unsigned eight bit values. This class provides safe access to the array with methods to retrieve 8/16/32/64-bit elements, or and arbitrary segment starting at the index, from the array while advancing the current index. An exception is thrown by these methods whenever the retrieval would reach beyond the size of the buffer. IndexedBuffers do not own the memory of the buffers they wrap.

### G.68.2 Constructor & Destructor Documentation

#### G.68.2.1 IndexedBuffer() [1/4]

```
BiometricEvaluation::Memory::IndexedBuffer::IndexedBuffer ()
```

Wrap a nullptr buffer.

#### G.68.2.2 IndexedBuffer() [2/4]

```
BiometricEvaluation::Memory::IndexedBuffer::IndexedBuffer (
 const uint8_t * data,
 uint64_t size)
```

Wrap an existing buffer of a given length.

Parameters

|             |                 |
|-------------|-----------------|
| <i>data</i> | Buffer to wrap. |
| <i>size</i> | Size of buffer. |

#### G.68.2.3 IndexedBuffer() [3/4]

```
BiometricEvaluation::Memory::IndexedBuffer::IndexedBuffer (
 const uint8Array & aa)
```

Wrap an existing uint8Array.

## Parameters

|           |                     |
|-----------|---------------------|
| <i>aa</i> | uint8Array to wrap. |
|-----------|---------------------|

**G.68.2.4 IndexedBuffer()** [4/4]

```
BiometricEvaluation::Memory::IndexedBuffer::IndexedBuffer (
 const IndexedBuffer & copy) [default]
```

Copy constructor (default).

**G.68.2.5 ~IndexedBuffer()**

```
virtual BiometricEvaluation::Memory::IndexedBuffer::~~IndexedBuffer () [virtual], [default]
```

Destructor (default).

**G.68.3 Member Function Documentation****G.68.3.1 get()**

```
virtual const uint8_t* BiometricEvaluation::Memory::IndexedBuffer::get () const [virtual]
```

Returns a pointer to the managed buffer.

## Returns

Pointer to the managed buffer.

Reimplemented in **BiometricEvaluation::Memory::MutableIndexedBuffer** (p. 447).

**G.68.3.2 getIndex()**

```
uint32_t BiometricEvaluation::Memory::IndexedBuffer::getIndex () const
```

Obtain the current index into the buffer.

## Returns

The current buffer index.

## Note

When **getIndex()** (p. 397) == **getSize()** (p. 397), the buffer is exhausted from scanning.

**G.68.3.3 getSize()**

```
uint32_t BiometricEvaluation::Memory::IndexedBuffer::getSize () const
```

Obtain the current size of the buffer.

## Returns

The current buffer size.

### G.68.3.4 scan()

```
uint64_t BiometricEvaluation::Memory::IndexedBuffer::scan (
 void * buf,
 uint64_t len)
```

Obtain the next 'n' elements of the buffer and increment the current index value by n.

#### Parameters

|    |            |                                              |
|----|------------|----------------------------------------------|
| in | <i>buf</i> | Buffer to store the copied data, or nullptr. |
| in | <i>len</i> | The number of elements to copy.              |

#### Exceptions

|                                  |                          |
|----------------------------------|--------------------------|
| <b>Error::DataError</b> (p. 295) | The buffer is exhausted. |
|----------------------------------|--------------------------|

#### Returns

The number of elements copied.

### G.68.3.5 scanBeU16Val()

```
uint16_t BiometricEvaluation::Memory::IndexedBuffer::scanBeU16Val ()
```

Obtain the next two elements of the buffer, scanned as a big-endian value, and increment the current index value.

#### Returns

The next element of the buffer as an unsigned 16-bit value.

#### Exceptions

|                                  |                          |
|----------------------------------|--------------------------|
| <b>Error::DataError</b> (p. 295) | The buffer is exhausted. |
|----------------------------------|--------------------------|

### G.68.3.6 scanBeU32Val()

```
uint32_t BiometricEvaluation::Memory::IndexedBuffer::scanBeU32Val ()
```

Obtain the next four elements of the buffer, scanned as a big-endian value, and increment the current index value.

#### Returns

The next element of the buffer as an unsigned 32-bit value.

#### Exceptions

|                                  |                          |
|----------------------------------|--------------------------|
| <b>Error::DataError</b> (p. 295) | The buffer is exhausted. |
|----------------------------------|--------------------------|



### G.68.3.7 scanU16Val()

```
uint16_t BiometricEvaluation::Memory::IndexedBuffer::scanU16Val ()
```

Obtain the next two elements of the buffer and increment the current index value.

#### Returns

The next element of the buffer as an unsigned 16-bit value.

#### Exceptions

|                                  |                          |
|----------------------------------|--------------------------|
| <b>Error::DataError</b> (p. 295) | The buffer is exhausted. |
|----------------------------------|--------------------------|

### G.68.3.8 scanU32Val()

```
uint32_t BiometricEvaluation::Memory::IndexedBuffer::scanU32Val ()
```

Obtain the next four elements of the buffer and increment the current index value by four.

#### Returns

The next element of the buffer as an unsigned 32-bit value.

#### Exceptions

|                                  |                          |
|----------------------------------|--------------------------|
| <b>Error::DataError</b> (p. 295) | The buffer is exhausted. |
|----------------------------------|--------------------------|

### G.68.3.9 scanU64Val()

```
uint64_t BiometricEvaluation::Memory::IndexedBuffer::scanU64Val ()
```

Obtain the next eight elements of the buffer and increment the current index value by eight.

#### Returns

The next element of the buffer as an unsigned 64-bit value.

#### Exceptions

|                                  |                          |
|----------------------------------|--------------------------|
| <b>Error::DataError</b> (p. 295) | The buffer is exhausted. |
|----------------------------------|--------------------------|

### G.68.3.10 scanU8Val()

```
uint8_t BiometricEvaluation::Memory::IndexedBuffer::scanU8Val ()
```

Obtain the next element of the buffer and increment the current index value.

## Returns

The next element of the buffer as an unsigned 8-bit value.

## Exceptions

|                                  |                          |
|----------------------------------|--------------------------|
| <b>Error::DataError</b> (p. 295) | The buffer is exhausted. |
|----------------------------------|--------------------------|

**G.68.3.11 setIndex()**

```
void BiometricEvaluation::Memory::IndexedBuffer::setIndex (
 uint64_t index)
```

Set the current index into the buffer.

## Parameters

|    |              |                         |
|----|--------------|-------------------------|
| in | <i>index</i> | The index value to set. |
|----|--------------|-------------------------|

## Exceptions

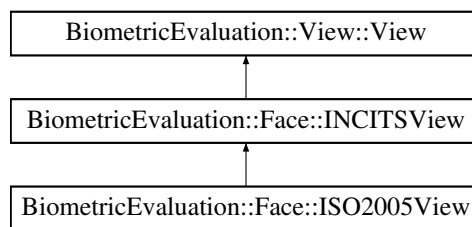
|                                       |                                   |
|---------------------------------------|-----------------------------------|
| <b>Error::ParameterError</b> (p. 474) | The index parameter is too large. |
|---------------------------------------|-----------------------------------|

**G.69 BiometricEvaluation::Face::ISO2005View Class Reference**

A class to represent single face view and derived information.

```
#include <be_face_iso2005view.h>
```

Inheritance diagram for BiometricEvaluation::Face::ISO2005View:

**Public Member Functions**

- **ISO2005View** ()  
*Construct an empty ISO2005 **Face** (p. 109) **Image** (p. 120) Data record.*
- **ISO2005View** (const std::string &filename, const uint32\_t viewNumber)  
*Construct an ISO 2005 face view from the named file.*
- **ISO2005View** (const **Memory::uint8Array** &buffer, const uint32\_t viewNumber)  
*Construct an ISO 2005 face view from a record contained in a buffer.*

## Protected Member Functions

- void **readISOHeader** ( **BiometricEvaluation::Memory::IndexedBuffer** &buf)  
*Read the face image data record header from an ISO 2005 record.*

## Static Protected Attributes

- static const uint32\_t **BASE\_SPEC\_VERSION** = 0x30313000

## G.69.1 Detailed Description

A class to represent single face view and derived information.

A base **Face::ISO2005View** (p. 400) class represents an ISO 2005 face image data view.

## G.69.2 Constructor & Destructor Documentation

### G.69.2.1 ISO2005View() [1/2]

```
BiometricEvaluation::Face::ISO2005View::ISO2005View (
 const std::string & filename,
 const uint32_t viewNumber)
```

Construct an ISO 2005 face view from the named file.

The entire face image data record is passed into this method, with the specific instance of the facial image that is to be extraced from the record.

Parameters

|    |                   |                                                                      |
|----|-------------------|----------------------------------------------------------------------|
| in | <i>filename</i>   | The name of the file containing the complete face image data record. |
| in | <i>viewNumber</i> | The facial information instance to read.                             |

Exceptions

|                                  |                                   |
|----------------------------------|-----------------------------------|
| <b>Error::DataError</b> (p. 295) | Invalid record format.            |
| <b>Error::FileError</b> (p. 316) | Could not open or read from file. |

### G.69.2.2 ISO2005View() [2/2]

```
BiometricEvaluation::Face::ISO2005View::ISO2005View (
 const Memory::uint8Array & buffer,
 const uint32_t viewNumber)
```

Construct an ISO 2005 face view from a record contained in a buffer.

The entire face image data record is passed into this method, with the specific instance of the facial image that is to be extraced from the record.

Parameters

|    |               |                                                            |
|----|---------------|------------------------------------------------------------|
| in | <i>buffer</i> | The buffer containing the complete face image data record. |
|----|---------------|------------------------------------------------------------|

## Parameters

|    |                   |                                          |
|----|-------------------|------------------------------------------|
| in | <i>viewNumber</i> | The facial information instance to read. |
|----|-------------------|------------------------------------------|

## Exceptions

|                                  |                        |
|----------------------------------|------------------------|
| <b>Error::DataError</b> (p. 295) | Invalid record format. |
|----------------------------------|------------------------|

## G.69.3 Member Function Documentation

### G.69.3.1 readISOHeader()

```
void BiometricEvaluation::Face::ISO2005View::readISOHeader (
 BiometricEvaluation::Memory::IndexedBuffer & buf) [protected]
 Read the face image data record header from an ISO 2005 record.
```

## Parameters

|    |            |                                                                                                                          |
|----|------------|--------------------------------------------------------------------------------------------------------------------------|
| in | <i>buf</i> | The indexed buffer containing the record data. The index of the buffer will be changed to the location after the header. |
|----|------------|--------------------------------------------------------------------------------------------------------------------------|

## Exceptions

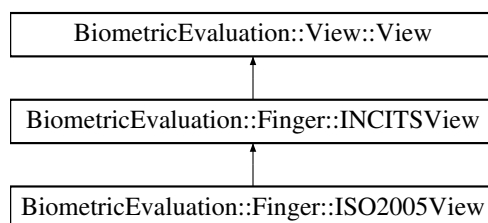
|                  |                                         |
|------------------|-----------------------------------------|
| <b>DataError</b> | The record has invalid or missing data. |
|------------------|-----------------------------------------|

## G.70 BiometricEvaluation::Finger::ISO2005View Class Reference

A class to represent single finger view and derived information.

```
#include <be_finger_iso2005view.h>
```

Inheritance diagram for BiometricEvaluation::Finger::ISO2005View:



### Public Member Functions

- **ISO2005View** (const std::string &fmrFilename, const std::string &firFilename, const uint32\_t view← Number)

*Construct an ISO-2005 finger view from records contained in files.*

- **ISO2005View** (const **Memory::uint8Array** &fmrBuffer, const **Memory::uint8Array** &firBuffer, const uint32\_t viewNumber)

*Construct an ISO-2005 finger view from records contained in buffers.*

## Protected Member Functions

- void **readFMRHeader** ( **Memory::IndexedBuffer** &buf)
- void **readCoreDeltaData** ( **Memory::IndexedBuffer** &buf, uint32\_t dataLength, Feature::CorePointSet &cores, Feature::DeltaPointSet &deltas)

*Read the core points data.*

## Static Protected Attributes

- static const uint32\_t **BASE\_SPEC\_VERSION** = 0x20323000

## Additional Inherited Members

### G.70.1 Detailed Description

A class to represent single finger view and derived information.

A **Finger::ISO2005View** (p. 402) object represents a finger view from a ISO/IEC-2005 **Finger** (p. 115) Minutiae Record.

### G.70.2 Constructor & Destructor Documentation

#### G.70.2.1 ISO2005View() [1/2]

```
BiometricEvaluation::Finger::ISO2005View::ISO2005View (
 const std::string & fmrFilename,
 const std::string & firFilename,
 const uint32_t viewNumber)
```

Construct an ISO-2005 finger view from records contained in files.

A view can be constructed from a single record, with information missing as appropriate. For example, if a view is constructed with just the minutiae record, no image would be part of the view. However, the image size etc. would be present because that information is also present in the minutiae record.

Parameters

|    |                    |                                                                      |
|----|--------------------|----------------------------------------------------------------------|
| in | <i>fmrFilename</i> | The name of the file containing the complete finger minutiae record. |
| in | <i>firFilename</i> | The name of the file containing the complete finger image record.    |
| in | <i>viewNumber</i>  | The finger view number to use.                                       |

#### G.70.2.2 ISO2005View() [2/2]

```
BiometricEvaluation::Finger::ISO2005View::ISO2005View (
 const Memory::uint8Array & fmrBuffer,
```

```
const Memory::uint8Array & firBuffer,
const uint32_t viewNumber)
```

Construct an ISO-2005 finger view from records contained in buffers.

A view can be constructed from a single record, with information missing as appropriate. For example, if a view is constructed with just the minutiae record, no image would be part of the view. However, the image size etc. would be present because that information is also present in the minutiae record.

Parameters

|    |                   |                                                            |
|----|-------------------|------------------------------------------------------------|
| in | <i>fmrBuffer</i>  | The buffer containing the complete finger minutiae record. |
| in | <i>firBuffer</i>  | The buffer containing the complete finger image record.    |
| in | <i>viewNumber</i> | The finger view number to use.                             |

Exceptions

|                                  |                        |
|----------------------------------|------------------------|
| <b>Error::DataError</b> (p. 295) | Invalid record format. |
|----------------------------------|------------------------|

## G.70.3 Member Function Documentation

### G.70.3.1 readCoreDeltaData()

```
void BiometricEvaluation::Finger::ISO2005View::readCoreDeltaData (
 Memory::IndexedBuffer & buf,
 uint32_t dataLength,
 Feature::CorePointSet & cores,
 Feature::DeltaPointSet & deltas) [protected], [virtual]
```

Read the core points data.

This method must be overridden by derived classes to read data in a specific record format.

Parameters

|         |                   |                                                                                                                                                    |
|---------|-------------------|----------------------------------------------------------------------------------------------------------------------------------------------------|
| in, out | <i>buf</i>        | The indexed buffer containing the record data. On function exit, the buffer index will be set to the location after the last core point data item. |
| out     | <i>cores</i>      | The set of core data items.                                                                                                                        |
| out     | <i>deltas</i>     | The set of delta data items.                                                                                                                       |
| in      | <i>dataLength</i> | The length of the entire ridge count data block.                                                                                                   |

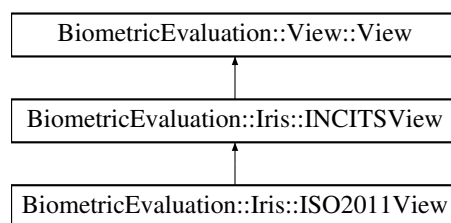
Implements **BiometricEvaluation::Finger::INCITSView** (p. 390).

## G.71 BiometricEvaluation::Iris::ISO2011View Class Reference

A class to represent single iris view and derived information.

```
#include <be_iris_iso2011view.h>
```

Inheritance diagram for BiometricEvaluation::Iris::ISO2011View:



## Public Member Functions

- **ISO2011View** ()  
*Construct an empty ISO 2011 iris view.*
- **ISO2011View** (const std::string &filename, const uint32\_t viewNumber)  
*Construct an ISO 2011 iris view from the named file.*
- **ISO2011View** (const **Memory::uint8Array** &buffer, const uint32\_t viewNumber)  
*Construct an ISO 2011 iris view from a record contained in a buffer.*

## Protected Member Functions

- void **readISOHeader** ( **BiometricEvaluation::Memory::IndexedBuffer** &buf)

## Static Protected Attributes

- static const uint32\_t **BASE\_SPEC\_VERSION** = 0x30323000

## Additional Inherited Members

### G.71.1 Detailed Description

A class to represent single iris view and derived information.

An Iris::ISO2011VIEW class represents an ISO 19794-6 iris image record view.

### G.71.2 Constructor & Destructor Documentation

#### G.71.2.1 ISO2011View() [1/2]

```

BiometricEvaluation::Iris::ISO2011View::ISO2011View (
 const std::string & filename,
 const uint32_t viewNumber)

```

Construct an ISO 2011 iris view from the named file.

Parameters

|    |                   |                                                                 |
|----|-------------------|-----------------------------------------------------------------|
| in | <i>filename</i>   | The name of the file containing the complete iris image record. |
| in | <i>viewNumber</i> | The eye number to use.                                          |

## Exceptions

|                                  |                                   |
|----------------------------------|-----------------------------------|
| <b>Error::DataError</b> (p. 295) | Invalid record format.            |
| <b>Error::FileError</b> (p. 316) | Could not open or read from file. |

**G.71.2.2 ISO2011View() [2/2]**

```
BiometricEvaluation::Iris::ISO2011View::ISO2011View (
 const Memory::uint8Array & buffer,
 const uint32_t viewNumber)
```

Construct an ISO 2011 iris view from a record contained in a buffer.

## Parameters

|    |                   |                                                       |
|----|-------------------|-------------------------------------------------------|
| in | <i>buffer</i>     | The buffer containing the complete iris image record. |
| in | <i>viewNumber</i> | The eye number to use.                                |

## Exceptions

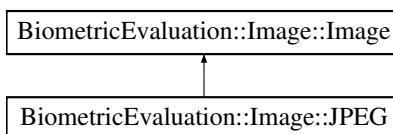
|                                  |                        |
|----------------------------------|------------------------|
| <b>Error::DataError</b> (p. 295) | Invalid record format. |
|----------------------------------|------------------------|

**G.72 BiometricEvaluation::Image::JPEG Class Reference**

A JPEG-encoded image.

```
#include <be_image_jpeg.h>
```

Inheritance diagram for BiometricEvaluation::Image::JPEG:

**Public Member Functions**

- **JPEG** (const uint8\_t \*data, const uint64\_t size)
- **JPEG** (const **Memory::uint8Array** &data)
- **Memory::uint8Array** **getRawGrayscaleData** (uint8\_t depth) const  
*Accessor for decompressed data in grayscale.*
- **Memory::uint8Array** **getRawData** () const  
*Accessor for the raw image data. The data returned should not be compressed or encoded.*



Static Public Member Functions

- static bool **isJPEG** (const uint8\_t \*data, uint64\_t size)
- static int **getc\_skip\_marker\_segment** (const unsigned short marker, unsigned char \*\*cbufptr, unsigned char \*ebufptr)

Additional Inherited Members

G.72.1 Detailed Description

A JPEG-encoded image.

G.72.2 Member Function Documentation

G.72.2.1 getRawData()

**Memory::uint8Array** BiometricEvaluation::Image::JPEG::getRawData ( ) const [virtual]  
Accessor for the raw image data. The data returned should not be compressed or encoded.

Returns

AutoArray holding raw image data.

Exceptions

|                                  |                                                 |
|----------------------------------|-------------------------------------------------|
| <i>Error::DataError</i> (p. 295) | <b>Error</b> (p. 108) decompressing image data. |
|----------------------------------|-------------------------------------------------|

Implements **BiometricEvaluation::Image::Image** (p. 361).

G.72.2.2 getRawGrayscaleData()

**Memory::uint8Array** BiometricEvaluation::Image::JPEG::getRawGrayscaleData (   
uint8\_t depth ) const [virtual]  
Accessor for decompressed data in grayscale.

Parameters

|              |                                                                                         |
|--------------|-----------------------------------------------------------------------------------------|
| <i>depth</i> | The desired bit depth of the resulting raw image. This value may either be 16, 8, or 1. |
|--------------|-----------------------------------------------------------------------------------------|

Returns

AutoArray holding raw grayscale image data.

Exceptions

|                                       |                                                     |
|---------------------------------------|-----------------------------------------------------|
| <i>Error::DataError</i> (p. 295)      | <b>Error</b> (p. 108) decompressing image data.     |
| <i>Error::NotImplemented</i> (p. 456) | Unsupported conversion based on source color depth. |
| <i>Error::ParameterError</i> (p. 474) | Invalid value for depth.                            |

## Note

This method does not save a cached copy of the decompressed image because the bit depth of the image can be changed between calls.

When depth is 1, this method returns an image that uses 8 bits to represent a single pixel. The depth parameter is used to adjust the number of gray levels. When depth is 1, there are only 2 gray levels (black and white), despite using 8 bits to represent each pixel.

Alpha channels are completely ignored when converting to grayscale.

Implements **BiometricEvaluation::Image::Image** (p. 362).

### G.72.2.3 isJPEG()

```
static bool BiometricEvaluation::Image::JPEG::isJPEG (
 const uint8_t * data,
 uint64_t size) [static]
```

Whether or not data is a Lossy **JPEG** (p. 406) image.

## Parameters

|    |             |                      |
|----|-------------|----------------------|
| in | <i>data</i> | The buffer to check. |
| in | <i>size</i> | The size of data.    |

## Returns

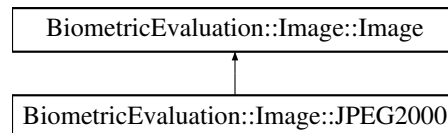
true if data appears to be a Lossy **JPEG** (p. 406) image, false otherwise

## G.73 BiometricEvaluation::Image::JPEG2000 Class Reference

A JPEG-2000-encoded image.

```
#include <be_image_jpeg2000.h>
```

Inheritance diagram for BiometricEvaluation::Image::JPEG2000:



### Public Member Functions

- **JPEG2000** (const uint8\_t \*data, const uint64\_t size, const int8\_t codecFormat=2)

Create a new **JPEG2000** (p. 408) object.

- **JPEG2000** (const **Memory::uint8Array** &data)
- **Memory::uint8Array** **getRawData** () const

Accessor for the raw image data. The data returned should not be compressed or encoded.

- **Memory::uint8Array** **getRawGrayscaleData** (uint8\_t depth) const

Accessor for decompressed data in grayscale.

Static Public Member Functions

- static bool **isJPEG2000** (const uint8\_t \*data, uint64\_t size)

Additional Inherited Members

G.73.1 Detailed Description

A JPEG-2000-encoded image.

G.73.2 Constructor & Destructor Documentation

G.73.2.1 JPEG2000()

```
BiometricEvaluation::Image::JPEG2000::JPEG2000 (
 const uint8_t * data,
 const uint64_t size,
 const int8_t codecFormat = 2)
```

Create a new **JPEG2000** (p. 408) object.

Parameters

|    |              |                                           |
|----|--------------|-------------------------------------------|
| in | <i>data</i>  | The image data.                           |
| in | <i>size</i>  | The size of the image data, in bytes.     |
| in | <i>codec</i> | The OPJ_CODEC_FORMAT used to encode data. |

Exceptions

|                                      |                                                             |
|--------------------------------------|-------------------------------------------------------------|
| <b>Error::DataError</b> (p. 295)     | <b>Error</b> (p. 108) manipulating data.                    |
| <b>Error::StrategyError</b> (p. 567) | <b>Error</b> (p. 108) while creating <b>Image</b> (p. 355). |

G.73.3 Member Function Documentation

G.73.3.1 getRawData()

**Memory::uint8Array** BiometricEvaluation::Image::JPEG2000::getRawData ( ) const [virtual]  
Accessor for the raw image data. The data returned should not be compressed or encoded.

Returns

AutoArray holding raw image data.

Exceptions

|                                  |                                                 |
|----------------------------------|-------------------------------------------------|
| <b>Error::DataError</b> (p. 295) | <b>Error</b> (p. 108) decompressing image data. |
|----------------------------------|-------------------------------------------------|

Implements **BiometricEvaluation::Image::Image** (p. 361).

### G.73.3.2 getRawGrayscaleData()

```
Memory::uint8Array BiometricEvaluation::Image::JPEG2000::getRawGrayscaleData (
 uint8_t depth) const [virtual]
```

Accessor for decompressed data in grayscale.

Parameters

|              |                                                                                         |
|--------------|-----------------------------------------------------------------------------------------|
| <i>depth</i> | The desired bit depth of the resulting raw image. This value may either be 16, 8, or 1. |
|--------------|-----------------------------------------------------------------------------------------|

Returns

AutoArray holding raw grayscale image data.

Exceptions

|                                       |                                                     |
|---------------------------------------|-----------------------------------------------------|
| <b>Error::DataError</b> (p. 295)      | <b>Error</b> (p. 108) decompressing image data.     |
| <b>Error::NotImplemented</b> (p. 456) | Unsupported conversion based on source color depth. |
| <b>Error::ParameterError</b> (p. 474) | Invalid value for depth.                            |

Note

This method does not save a cached copy of the decompressed image because the bit depth of the image can be changed between calls.

When depth is 1, this method returns an image that uses 8 bits to represent a single pixel. The depth parameter is used to adjust the number of gray levels. When depth is 1, there are only 2 gray levels (black and white), despite using 8 bits to represent each pixel.

Alpha channels are completely ignored when converting to grayscale.

Implements **BiometricEvaluation::Image::Image** (p. 362).

### G.73.3.3 isJPEG2000()

```
static bool BiometricEvaluation::Image::JPEG2000::isJPEG2000 (
 const uint8_t * data,
 uint64_t size) [static]
```

Whether or not data is a JPEG-2000 image.

Parameters

|    |             |                      |
|----|-------------|----------------------|
| in | <i>data</i> | The buffer to check. |
| in | <i>size</i> | The size of data.    |

Returns

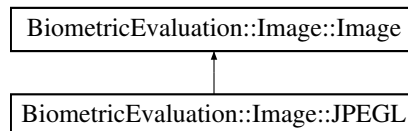
true if data appears to be a JPEG-2000 image, false otherwise.

## G.74 BiometricEvaluation::Image::JPEGL Class Reference

A Lossless JPEG-encoded image.

```
#include <be_image_jpeg1.h>
```

Inheritance diagram for BiometricEvaluation::Image::JPEGL:



### Public Member Functions

- **JPEGL** (const uint8\_t \*data, const uint64\_t size)
- **JPEGL** (const **Memory::uint8Array** &data)
- **Memory::uint8Array** **getRawGrayscaleData** (uint8\_t depth) const  
*Accessor for decompressed data in grayscale.*
- **Memory::uint8Array** **getRawData** () const  
*Accessor for the raw image data. The data returned should not be compressed or encoded.*

### Static Public Member Functions

- static bool **isJPEGL** (const uint8\_t \*data, uint64\_t size)

### Additional Inherited Members

#### G.74.1 Detailed Description

A Lossless JPEG-encoded image.

#### G.74.2 Member Function Documentation

##### G.74.2.1 getRawData()

```
Memory::uint8Array BiometricEvaluation::Image::JPEGL::getRawData () const [virtual]
```

Accessor for the raw image data. The data returned should not be compressed or encoded.

Returns

AutoArray holding raw image data.

Exceptions

|                                  |                                                 |
|----------------------------------|-------------------------------------------------|
| <b>Error::DataError</b> (p. 295) | <b>Error</b> (p. 108) decompressing image data. |
|----------------------------------|-------------------------------------------------|

Implements **BiometricEvaluation::Image::Image** (p. 361).

### G.74.2.2 getRawGrayscaleData()

```
Memory::uint8Array BiometricEvaluation::Image::JPEG::getRawGrayscaleData (
 uint8_t depth) const [virtual]
```

Accessor for decompressed data in grayscale.

Parameters

|              |                                                                                         |
|--------------|-----------------------------------------------------------------------------------------|
| <i>depth</i> | The desired bit depth of the resulting raw image. This value may either be 16, 8, or 1. |
|--------------|-----------------------------------------------------------------------------------------|

Returns

AutoArray holding raw grayscale image data.

Exceptions

|                                       |                                                     |
|---------------------------------------|-----------------------------------------------------|
| <b>Error::DataError</b> (p. 295)      | <b>Error</b> (p. 108) decompressing image data.     |
| <b>Error::NotImplemented</b> (p. 456) | Unsupported conversion based on source color depth. |
| <b>Error::ParameterError</b> (p. 474) | Invalid value for depth.                            |

Note

This method does not save a cached copy of the decompressed image because the bit depth of the image can be changed between calls.

When depth is 1, this method returns an image that uses 8 bits to represent a single pixel. The depth parameter is used to adjust the number of gray levels. When depth is 1, there are only 2 gray levels (black and white), despite using 8 bits to represent each pixel.

Alpha channels are completely ignored when converting to grayscale.

Implements **BiometricEvaluation::Image::Image** (p. 362).

### G.74.2.3 isJPEG()

```
static bool BiometricEvaluation::Image::JPEG::isJPEG (
 const uint8_t * data,
 uint64_t size) [static]
```

Whether or not data is a Lossless **JPEG** (p. 406) image.

Parameters

|    |             |                      |
|----|-------------|----------------------|
| in | <i>data</i> | The buffer to check. |
| in | <i>size</i> | The size of data.    |

Returns

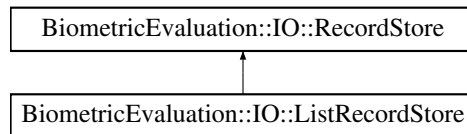
true if data appears to be a Lossless **JPEG** (p. 406) image, false otherwise.

## G.75 BiometricEvaluation::IO::ListRecordStore Class Reference

**RecordStore** (p. 504) that reads a list of keys from a text file, and retrieves the data from another **RecordStore** (p. 504).

```
#include <be_io_listrecstore.h>
```

Inheritance diagram for BiometricEvaluation::IO::ListRecordStore:



### Public Member Functions

- **ListRecordStore** (const std::string &pathname)
- **~ListRecordStore** ()
- void **insert** (const std::string &key, const void \*const data, const uint64\_t size) override
- void **remove** (const std::string &key) override
- **Memory::uint8Array read** (const std::string &key) const override
 

*Read a complete record from a store.*
- void **replace** (const std::string &key, const void \*const data, const uint64\_t size) override final
- uint64\_t **length** (const std::string &key) const override
- void **flush** (const std::string &key) const override
- void **sync** () const override
- **RecordStore::Record sequence** (int cursor= **BE\_RECSTORE\_SEQ\_NEXT**) override
 

*Sequence through a **RecordStore** (p. 504), returning the key/data pairs.*
- std::string **sequenceKey** (int cursor= **BE\_RECSTORE\_SEQ\_NEXT**) override
 

*Sequence through a **RecordStore** (p. 504), returning the key.*
- void **setCursorAtKey** (const std::string &key) override
- void **move** (const std::string &pathname) override
 

*Move the **RecordStore** (p. 504).*
- uint64\_t **getSpaceUsed** () const override
 

*Obtain real storage utilization.*
- unsigned int **getCount** () const override
- std::string **getPathname** () const override
- std::string **getDescription** () const override
- void **changeDescription** (const std::string &description) override

## Additional Inherited Members

### G.75.1 Detailed Description

**RecordStore** (p. 504) that reads a list of keys from a text file, and retrieves the data from another **RecordStore** (p. 504).

ListRecordStores must be hand-crafted by first setting the 'Source Record Store', 'Type', and 'Count' properties in the .rscontrol.prop file. 'Source Record Store' is the complete path of the **RecordStore** (p. 504) containing the actual data records. Type must be 'List'. Count should match the number of entries in the file created next. Other properties are as in a "normal" **RecordStore** (p. 504); see example below.

Second, create a file called 'KeyList.txt' in the **RecordStore** (p. 504) directory containing a list of keys, one per line.

ListRecordStores can also be created and modified with versions of rstool(1) from 2013 or later.

Example .rscontrol.prop file: Count = 10 Description = Search records for SDK TESTSDK Name = Test↵  
LRS Type = List Source Record Store = /Users/wsalamon/sandbox/SD29.rs

Note

List RecordStores must be opened read-only.

### G.75.2 Constructor & Destructor Documentation

#### G.75.2.1 ListRecordStore()

```
BiometricEvaluation::IO::ListRecordStore::ListRecordStore (
 const std::string & pathname)
```

Constructor, always opening read-only

#### G.75.2.2 ~ListRecordStore()

```
BiometricEvaluation::IO::ListRecordStore::~~ListRecordStore ()
```

Destructor

### G.75.3 Member Function Documentation

#### G.75.3.1 changeDescription()

```
void BiometricEvaluation::IO::ListRecordStore::changeDescription (
 const std::string & description) [override], [virtual]
```

Change the description of the **RecordStore** (p. 504).

Parameters

|    |                    |                      |
|----|--------------------|----------------------|
| in | <i>description</i> | The new description. |
|----|--------------------|----------------------|

Exceptions

|                                      |                                                             |
|--------------------------------------|-------------------------------------------------------------|
| <b>Error::StrategyError</b> (p. 567) | An error occurred when using the underlying storage system. |
|--------------------------------------|-------------------------------------------------------------|



Implements **BiometricEvaluation::IO::RecordStore** (p. 506).

### G.75.3.2 flush()

```
void BiometricEvaluation::IO::ListRecordStore::flush (
 const std::string & key) const [override], [virtual]
```

Commit the record's data to storage.

Parameters

|    |     |                                      |
|----|-----|--------------------------------------|
| in | key | The key of the record to be flushed. |
|----|-----|--------------------------------------|

Exceptions

|                                           |                                                             |
|-------------------------------------------|-------------------------------------------------------------|
| <b>Error::ObjectDoesNotExist</b> (p. 457) | A record for the key does not exist.                        |
| <b>Error::StrategyError</b> (p. 567)      | An error occurred when using the underlying storage system. |

Implements **BiometricEvaluation::IO::RecordStore** (p. 508).

### G.75.3.3 getCount()

```
unsigned int BiometricEvaluation::IO::ListRecordStore::getCount () const [override], [virtual]
```

Obtain the number of items in the **RecordStore** (p. 504).

Returns

The number of items in the **RecordStore** (p. 504).

Implements **BiometricEvaluation::IO::RecordStore** (p. 508).

### G.75.3.4 getDescription()

```
std::string BiometricEvaluation::IO::ListRecordStore::getDescription () const [override], [virtual]
```

Obtain a textual description of the **RecordStore** (p. 504).

Returns

The **RecordStore** (p. 504)'s description.

Implements **BiometricEvaluation::IO::RecordStore** (p. 508).

### G.75.3.5 getPathname()

```
std::string BiometricEvaluation::IO::ListRecordStore::getPathname () const [override], [virtual]
```

Return the path name of the **RecordStore** (p. 504).

Returns

Where in the file system the **RecordStore** (p. 504) is located.

Implements **BiometricEvaluation::IO::RecordStore** (p. 509).

### G.75.3.6 `getSpaceUsed()`

```
uint64_t BiometricEvaluation::IO::ListRecordStore::getSpaceUsed () const [override], [virtual]
```

Obtain real storage utilization.

The amount of disk space used, for example. This is the actual space allocated by the underlying storage mechanism, in bytes.

Returns

The amount of backing storage used by the **RecordStore** (p. 504).

Exceptions

|                                      |                                                             |
|--------------------------------------|-------------------------------------------------------------|
| <b>Error::StrategyError</b> (p. 567) | An error occurred when using the underlying storage system. |
|--------------------------------------|-------------------------------------------------------------|

Implements **BiometricEvaluation::IO::RecordStore** (p. 509).

### G.75.3.7 `insert()`

```
void BiometricEvaluation::IO::ListRecordStore::insert (
 const std::string & key,
 const void *const data,
 const uint64_t size) [override], [virtual]
```

Insert a record into the store.

Parameters

|    |             |                                       |
|----|-------------|---------------------------------------|
| in | <i>key</i>  | The key of the record to be inserted. |
| in | <i>data</i> | The data for the record.              |
| in | <i>size</i> | The size of the record, in bytes.     |

Exceptions

|                                      |                                                                                                                     |
|--------------------------------------|---------------------------------------------------------------------------------------------------------------------|
| <b>Error::ObjectExists</b> (p. 458)  | A record with the given key is already present.                                                                     |
| <b>Error::StrategyError</b> (p. 567) | The <b>RecordStore</b> (p. 504) is opened read-only, or an error occurred when using the underlying storage system. |

Implements **BiometricEvaluation::IO::RecordStore** (p. 510).

### G.75.3.8 `length()`

```
uint64_t BiometricEvaluation::IO::ListRecordStore::length (
 const std::string & key) const [override], [virtual]
```

Return the length of a record.

Parameters

|    |            |                        |
|----|------------|------------------------|
| in | <i>key</i> | The key of the record. |
|----|------------|------------------------|

## Returns

The record length.

## Exceptions

|                                           |                                                             |
|-------------------------------------------|-------------------------------------------------------------|
| <b>Error::ObjectDoesNotExist</b> (p. 457) | A record for the key does not exist.                        |
| <b>Error::StrategyError</b> (p. 567)      | An error occurred when using the underlying storage system. |

Implements **BiometricEvaluation::IO::RecordStore** (p. 510).

## G.75.3.9 move()

```
void BiometricEvaluation::IO::ListRecordStore::move (
 const std::string & pathname) [override], [virtual]
```

Move the **RecordStore** (p. 504).

The **RecordStore** (p. 504) can be moved to a new path in the file system.

## Parameters

|    |          |                                                  |
|----|----------|--------------------------------------------------|
| in | pathname | The new path of the <b>RecordStore</b> (p. 504). |
|----|----------|--------------------------------------------------|

## Exceptions

|                                      |                                                             |
|--------------------------------------|-------------------------------------------------------------|
| <b>Error::StrategyError</b> (p. 567) | An error occurred when using the underlying storage system. |
|--------------------------------------|-------------------------------------------------------------|

Implements **BiometricEvaluation::IO::RecordStore** (p. 511).

## G.75.3.10 read()

```
Memory::uint8Array BiometricEvaluation::IO::ListRecordStore::read (
 const std::string & key) const [override], [virtual]
```

Read a complete record from a store.

The AutoArray will be resized to match the size of the data.

## Parameters

|    |     |                                   |
|----|-----|-----------------------------------|
| in | key | The key of the record to be read. |
|----|-----|-----------------------------------|

## Returns

The record associated with the key.

## Exceptions

|                                           |                                                             |
|-------------------------------------------|-------------------------------------------------------------|
| <b>Error::ObjectDoesNotExist</b> (p. 457) | A record for the key does not exist.                        |
| <b>Error::StrategyError</b> (p. 567)      | An error occurred when using the underlying storage system. |

Implements **BiometricEvaluation::IO::RecordStore** (p. 512).

### G.75.3.11 remove()

```
void BiometricEvaluation::IO::ListRecordStore::remove (
 const std::string & key) [override], [virtual]
```

Remove a record from the store.

Parameters

|    |     |                                      |
|----|-----|--------------------------------------|
| in | key | The key of the record to be removed. |
|----|-----|--------------------------------------|

Exceptions

|                                           |                                                             |
|-------------------------------------------|-------------------------------------------------------------|
| <b>Error::ObjectDoesNotExist</b> (p. 457) | A record for the key does not exist.                        |
| <b>Error::StrategyError</b> (p. 567)      | An error occurred when using the underlying storage system. |

Implements **BiometricEvaluation::IO::RecordStore** (p. 513).

### G.75.3.12 replace()

```
void BiometricEvaluation::IO::ListRecordStore::replace (
 const std::string & key,
 const void *const data,
 const uint64_t size) [final], [override], [virtual]
```

Replace a complete record in a **RecordStore** (p. 504).

Parameters

|    |      |                                       |
|----|------|---------------------------------------|
| in | key  | The key of the record to be replaced. |
| in | data | The data for the record.              |
| in | size | The size of the record, in bytes.     |

Exceptions

|                                           |                                                                                                                     |
|-------------------------------------------|---------------------------------------------------------------------------------------------------------------------|
| <b>Error::ObjectDoesNotExist</b> (p. 457) | A record for the key does not exist.                                                                                |
| <b>Error::StrategyError</b> (p. 567)      | The <b>RecordStore</b> (p. 504) is opened read-only, or an error occurred when using the underlying storage system. |

Reimplemented from **BiometricEvaluation::IO::RecordStore** (p. 514).

### G.75.3.13 sequence()

```
RecordStore::Record BiometricEvaluation::IO::ListRecordStore::sequence (
 int cursor = BE_RECSTORE_SEQ_NEXT) [override], [virtual]
```

Sequence through a **RecordStore** (p. 504), returning the key/data pairs.

Sequencing means to start at some point in the store and return the record, then repeatedly calling the function to return the next record. The starting point is typically the first record, and is set to that when the **RecordStore** (p. 504) object is created. The starting point can be reset by calling this method with the cursor parameter set to BE\_RECSTORE\_SEQ\_START.

Parameters

|    |               |                                                                  |
|----|---------------|------------------------------------------------------------------|
| in | <i>cursor</i> | The location within the sequence of the key/data pair to return. |
|----|---------------|------------------------------------------------------------------|

Returns

The record that is currently in sequence.

Exceptions

|                                           |                                                             |
|-------------------------------------------|-------------------------------------------------------------|
| <b>Error::ObjectDoesNotExist</b> (p. 457) | End of sequencing.                                          |
| <b>Error::StrategyError</b> (p. 567)      | An error occurred when using the underlying storage system. |

Implements **BiometricEvaluation::IO::RecordStore** (p. 515).

#### G.75.3.14 sequenceKey()

```
std::string BiometricEvaluation::IO::ListRecordStore::sequenceKey (
 int cursor = BE_RECSTORE_SEQ_NEXT) [override], [virtual]
```

Sequence through a **RecordStore** (p. 504), returning the key.

Sequencing means to start at some point in the store and return the key, then repeatedly calling the function to return the next key. The starting point is typically the first record, and is set to that when the **RecordStore** (p. 504) object is created. The starting point can be reset by calling this method with the cursor parameter set to BE\_RECSTORE\_SEQ\_START.

Parameters

|    |               |                                                                  |
|----|---------------|------------------------------------------------------------------|
| in | <i>cursor</i> | The location within the sequence of the key/data pair to return. |
|----|---------------|------------------------------------------------------------------|

Returns

The key of the currently sequenced record.

Exceptions

|                                           |                                                             |
|-------------------------------------------|-------------------------------------------------------------|
| <b>Error::ObjectDoesNotExist</b> (p. 457) | End of sequencing.                                          |
| <b>Error::StrategyError</b> (p. 567)      | An error occurred when using the underlying storage system. |

Implements **BiometricEvaluation::IO::RecordStore** (p. 515).

**G.75.3.15 setCursorAtKey()**

```
void BiometricEvaluation::IO::ListRecordStore::setCursorAtKey (
 const std::string & key) [override], [virtual]
```

Set the sequence cursor to an arbitrary position within the **RecordStore** (p. 504), starting at key. Key will be the first record returned from the next call to **sequence()** (p. 418).

## Parameters

|    |     |                                                                                                          |
|----|-----|----------------------------------------------------------------------------------------------------------|
| in | key | The key of the record which will be returned by the first subsequent call to <b>sequence()</b> (p. 418). |
|----|-----|----------------------------------------------------------------------------------------------------------|

## Exceptions

|                                           |                                                             |
|-------------------------------------------|-------------------------------------------------------------|
| <b>Error::ObjectDoesNotExist</b> (p. 457) | A record for the key does not exist.                        |
| <b>Error::StrategyError</b> (p. 567)      | An error occurred when using the underlying storage system. |

Implements **BiometricEvaluation::IO::RecordStore** (p. 516).

**G.75.3.16 sync()**

```
void BiometricEvaluation::IO::ListRecordStore::sync () const [override], [virtual]
```

Synchronize the entire record store to persistent storage.

## Exceptions

|                                      |                                                             |
|--------------------------------------|-------------------------------------------------------------|
| <b>Error::StrategyError</b> (p. 567) | An error occurred when using the underlying storage system. |
|--------------------------------------|-------------------------------------------------------------|

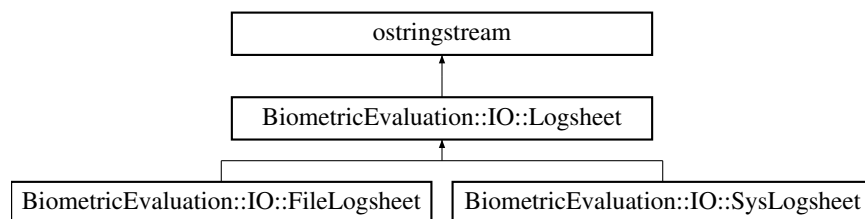
Implements **BiometricEvaluation::IO::RecordStore** (p. 516).

**G.76 BiometricEvaluation::IO::Logsheet Class Reference**

A class to represent a logging mechanism.

```
#include <be_io_logsheet.h>
```

Inheritance diagram for BiometricEvaluation::IO::Logsheet:

**Public Types**

- enum **Kind** { **Kind::Null**, **Kind::File**, **Kind::Syslog** }

## Public Member Functions

- **Logsheet ()**  
Create a **Logsheet** (p. 420) that has no backing store. A log entry is maintained, but cannot be permanently stored. This is the Null **Logsheet** (p. 420).
- virtual **~Logsheet ()**
- void **newEntry ()**  
Start a new entry, causing the existing entry to be closed and written.
- std::string **getCurrentEntry ()** const  
Obtain the contents of the current entry currently under construction.
- void **resetCurrentEntry ()**
- uint32\_t **getCurrentEntryNumber ()** const  
Obtain the current entry number.
- virtual void **write** (const std::string &entry)  
Write a string as an entry to the backing store.
- virtual void **writeComment** (const std::string &entry)  
Write a string as a comment to the backing store.
- virtual void **writeDebug** (const std::string &entry)  
Write a string as a debug entry to the backing store.
- void **setCommit** (const bool state)  
Enable or disable the commitment of normal entries to the backing log storage.
- bool **getCommit ()** const  
Get the current entry commit state.
- void **setDebugCommit** (const bool state)  
Enable or disable the commitment of debug entries to the backing log storage.
- bool **getDebugCommit ()** const  
Get the current debug entry commit state.
- void **setCommentCommit** (const bool state)  
Enable or disable the commitment of comment entries to the backing log storage.
- bool **getCommentCommit ()** const  
Get the current comment entry commit state.
- virtual void **sync ()**  
Synchronize any buffered data to the underlying backing store.
- void **setAutoSync** (bool state)
- bool **getAutoSync ()** const

## Static Public Member Functions

- static **Logsheet::Kind** **getTypeFromURL** (const std::string &url)  
Map the URL scheme, taken from a string containing the entire URL, into a **Logsheet** (p. 420) type.
- static bool **lineIsEntry** (const std::string &line)  
Helper function to determine whether a string is a valid log entry.
- static bool **lineIsComment** (const std::string &line)  
Helper function to determine whether a string is a valid comment log entry.
- static bool **lineIsDebug** (const std::string &line)  
Helper function to determine whether a string is a valid debug log entry.
- static std::string **trim** (const std::string &entry)  
Trim delimiters from **Logsheet** (p. 420) entries.

## Static Public Attributes

- static const char **CommentDelimiter** = '#'
- static const char **EntryDelimiter** = 'E'
- static const char **DebugDelimiter** = 'D'
- static const std::string **DescriptionTag**
- static const std::string **FILEURLSCHEME**
- static const std::string **SYSLOGURLSCHEME**

## Protected Member Functions

- void **incrementEntryNumber** ()  
*Increment the current entry number.*
- std::string **getCurrentEntryNumberAsString** () const  
*Obtain the current entry 'tag', in 'Edddd' format.*

### G.76.1 Detailed Description

A class to represent a logging mechanism.

A **Logsheet** (p. 420) is a string stream, so applications can write into the stream as a staging area using the << operator, then start a new entry by calling **newEntry()** (p. 425). Entries in the log are prefixed with an entry number, which is incremented when the entry is written (either by directly calling **write()** (p. 427), or calling **newEntry()** (p. 425)).

How the log data is stored is implemented by subclasses of **Logsheet** (p. 420).

Note

By default, the entries in the **Logsheet** (p. 420) may not be immediately written to the backing store, depending on the buffering behavior of the operating system. Applications can force a write by invoking **sync()** (p. 427), or force a write at every new log entry by invoking **setAutoSync(true)**.

Entries created by applications may be composed of more than one line (each separated by the newline character). The text at the beginning of a line should not "look like" an entry number:

Edddd

i.e. the entry delimiter followed by some digits. **Logsheet** (p. 420) won't check for that condition, but any existing **Logsheet** (p. 420) that is re-opened for append may have an incorrect starting entry number.

### G.76.2 Member Enumeration Documentation

#### G.76.2.1 Kind

```
enum BiometricEvaluation::IO::Logsheet::Kind [strong]
```

Enumerator

|        |                             |
|--------|-----------------------------|
| Null   | No backing store log sheet  |
| File   | File-based log sheet        |
| Syslog | Syslog daemon backing store |



## G.76.3 Constructor & Destructor Documentation

### G.76.3.1 ~Logsheet()

```
virtual BiometricEvaluation::IO::Logsheet::~~Logsheet () [virtual]
 Destructor
```

## G.76.4 Member Function Documentation

### G.76.4.1 getAutoSync()

```
bool BiometricEvaluation::IO::Logsheet::getAutoSync () const
 Return the current auto-sync state.
```

Returns

true if auto-sync is on, false otherwise.

### G.76.4.2 getCommentCommit()

```
bool BiometricEvaluation::IO::Logsheet::getCommentCommit () const
 Get the current comment entry commit state.
```

Returns

true if comment entries are committed to the backing store, false otherwise.

### G.76.4.3 getCommit()

```
bool BiometricEvaluation::IO::Logsheet::getCommit () const
 Get the current entry commit state.
```

Returns

true if normal entries are to be committed, false if not.

### G.76.4.4 getCurrentEntry()

```
std::string BiometricEvaluation::IO::Logsheet::getCurrentEntry () const
 Obtain the contents of the current entry currently under construction.
```

Returns

The text of the current entry.

**G.76.4.5** `getCurrentEntryNumber()`

```
uint32_t BiometricEvaluation::IO::Logsheet::getCurrentEntryNumber () const
```

Obtain the current entry number.

Returns

The current entry number.

**G.76.4.6** `getCurrentEntryNumberAsString()`

```
std::string BiometricEvaluation::IO::Logsheet::getCurrentEntryNumberAsString () const [protected]
```

Obtain the current entry 'tag', in 'Edddd' format.

Returns

The text of the current entry tag.

**G.76.4.7** `getDebugCommit()`

```
bool BiometricEvaluation::IO::Logsheet::getDebugCommit () const
```

Get the current debug entry commit state.

Returns

true if debug entries are committed to the backing store, false otherwise.

**G.76.4.8** `getTypeFromURL()`

```
static Logsheet::Kind BiometricEvaluation::IO::Logsheet::getTypeFromURL (
 const std::string & url) [static]
```

Map the URL scheme, taken from a string containing the entire URL, into a **Logsheet** (p. 420) type.

Parameters

|    |            |                                                               |
|----|------------|---------------------------------------------------------------|
| in | <i>url</i> | The uniform resource locator of the <b>Logsheet</b> (p. 420). |
|----|------------|---------------------------------------------------------------|

Returns

The type of **Logsheet** (p. 420) represented by the URL.

Exceptions

|                                       |                                       |
|---------------------------------------|---------------------------------------|
| <i>Error::ParameterError</i> (p. 474) | The URL scheme is missing or invalid. |
|---------------------------------------|---------------------------------------|

#### G.76.4.9 lineIsComment()

```
static bool BiometricEvaluation::IO::Logsheet::lineIsComment (
 const std::string & line) [static]
```

Helper function to determine whether a string is a valid comment log entry.

Parameters

|    |             |                                                    |
|----|-------------|----------------------------------------------------|
| in | <i>line</i> | The string potentially containing a comment entry. |
|----|-------------|----------------------------------------------------|

Returns

true if the string is a comment entry, false otherwise.

#### G.76.4.10 lineIsDebug()

```
static bool BiometricEvaluation::IO::Logsheet::lineIsDebug (
 const std::string & line) [static]
```

Helper function to determine whether a string is a valid debug log entry.

Parameters

|    |             |                                                  |
|----|-------------|--------------------------------------------------|
| in | <i>line</i> | The string potentially containing a debug entry. |
|----|-------------|--------------------------------------------------|

Returns

true if the string is a debug entry, false otherwise.

#### G.76.4.11 lineIsEntry()

```
static bool BiometricEvaluation::IO::Logsheet::lineIsEntry (
 const std::string & line) [static]
```

Helper function to determine whether a string is a valid log entry.

Parameters

|    |             |                                                |
|----|-------------|------------------------------------------------|
| in | <i>line</i> | The string potentially containing a log entry. |
|----|-------------|------------------------------------------------|

Returns

true if the string is a log entry, false otherwise.

#### G.76.4.12 newEntry()

```
void BiometricEvaluation::IO::Logsheet::newEntry ()
```

Start a new entry, causing the existing entry to be closed and written.

Applications do not have to call this method for the first entry, however, as the stream is ready for writing upon construction.

Exceptions

|                                                              |                                                            |
|--------------------------------------------------------------|------------------------------------------------------------|
| <b><i>Error::StrategyError</i></b> (p. <a href="#">567</a> ) | An error occurred when using the underlying backing store. |
|--------------------------------------------------------------|------------------------------------------------------------|

#### G.76.4.13 resetCurrentEntry()

```
void BiometricEvaluation::IO::Logsheet::resetCurrentEntry ()
```

Reset the current entry buffer to the beginning.

#### G.76.4.14 setAutoSync()

```
void BiometricEvaluation::IO::Logsheet::setAutoSync (
 bool state)
```

Turn on/off auto-sync of the data. Applications may gain performance by turning off auto-sync, or gain reliability by turning it on.

Parameters

|              |                                                                                                                                                                                                                                   |
|--------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>state</i> | When true, the data is sync'd whenever <b>newEntry()</b> (p. <a href="#">425</a> ) is or <b>write()</b> (p. <a href="#">427</a> ) is called. When false, <b>sync()</b> (p. <a href="#">427</a> ) must be called to force a write. |
|--------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

#### G.76.4.15 setCommentCommit()

```
void BiometricEvaluation::IO::Logsheet::setCommentCommit (
 const bool state)
```

Enable or disable the commitment of comment entries to the backing log storage.

When comment entry commitment is disabled, calls to writeComment may still be made, but those entries do not appear in the log backing store.

Parameters

|           |              |                                                            |
|-----------|--------------|------------------------------------------------------------|
| <i>in</i> | <i>state</i> | true if comment entries are to be committed, false if not. |
|-----------|--------------|------------------------------------------------------------|

#### G.76.4.16 setCommit()

```
void BiometricEvaluation::IO::Logsheet::setCommit (
 const bool state)
```

Enable or disable the commitment of normal entries to the backing log storage.

When entry commitment is disabled, the entry number is not incremented. Entries may be streamed into the object, and new entries created.

## Parameters

|    |       |                                                           |
|----|-------|-----------------------------------------------------------|
| in | state | True if normal entries are to be committed, false if not. |
|----|-------|-----------------------------------------------------------|

**G.76.4.17 setDebugCommit()**

```
void BiometricEvaluation::IO::Logsheet::setDebugCommit (
 const bool state)
```

Enable or disable the commitment of debug entries to the backing log storage.

When debug entry commitment is disabled, calls to writeDebug may still be made, but those entries do not appear in the log backing store.

## Parameters

|    |       |                                                          |
|----|-------|----------------------------------------------------------|
| in | state | true if debug entries are to be committed, false if not. |
|----|-------|----------------------------------------------------------|

**G.76.4.18 sync()**

```
virtual void BiometricEvaluation::IO::Logsheet::sync () [virtual]
```

Synchronize any buffered data to the underlying backing store.

This syncing is dependent on the behavior of the underlying storage mechanism.

## Exceptions

|                                                       |                                                            |
|-------------------------------------------------------|------------------------------------------------------------|
| <b>Error::StrategyError</b> (p. <a href="#">567</a> ) | An error occurred when using the underlying backing store. |
|-------------------------------------------------------|------------------------------------------------------------|

Reimplemented in **BiometricEvaluation::IO::FileLogsheet** (p. [323](#)), and **BiometricEvaluation::IO::SysLogsheet** (p. [573](#)).

**G.76.4.19 trim()**

```
static std::string BiometricEvaluation::IO::Logsheet::trim (
 const std::string & entry) [static]
```

Trim delimiters from **Logsheet** (p. [420](#)) entries.

Works for comments and numbered entries.

## Parameters

|    |       |                    |
|----|-------|--------------------|
| in | entry | The entry to trim. |
|----|-------|--------------------|

## Returns

Delimiter-less entry.

**G.76.4.20 write()**

```
virtual void BiometricEvaluation::IO::Logsheet::write (
 const std::string & entry) [virtual]
```

Write a string as an entry to the backing store.

This does not affect the current log entry buffer, but does increment the entry number.

## Parameters

|    |              |                            |
|----|--------------|----------------------------|
| in | <i>entry</i> | The text of the log entry. |
|----|--------------|----------------------------|

## Exceptions

|                                      |                                                            |
|--------------------------------------|------------------------------------------------------------|
| <b>Error::StrategyError</b> (p. 567) | An error occurred when using the underlying backing store. |
|--------------------------------------|------------------------------------------------------------|

Reimplemented in **BiometricEvaluation::IO::FileLogsheet** (p. 323), and **BiometricEvaluation::IO↵  
::SysLogsheet** (p. 573).

**G.76.4.21 writeComment()**

```
virtual void BiometricEvaluation::IO::Logsheet::writeComment (
 const std::string & entry) [virtual]
```

Write a string as a comment to the backing store.

This does not affect the current log entry buffer, and does not increment the entry number. A comment line is prefixed with CommentDelimiter followed by a space by this method.

## Parameters

|    |              |                          |
|----|--------------|--------------------------|
| in | <i>entry</i> | The text of the comment. |
|----|--------------|--------------------------|

## Exceptions

|                                      |                                                            |
|--------------------------------------|------------------------------------------------------------|
| <b>Error::StrategyError</b> (p. 567) | An error occurred when using the underlying backing store. |
|--------------------------------------|------------------------------------------------------------|

Reimplemented in **BiometricEvaluation::IO::FileLogsheet** (p. 324), and **BiometricEvaluation::IO↵  
::SysLogsheet** (p. 573).

**G.76.4.22 writeDebug()**

```
virtual void BiometricEvaluation::IO::Logsheet::writeDebug (
 const std::string & entry) [virtual]
```

Write a string as a debug entry to the backing store.

This does not affect the current log entry buffer, and does not increment the entry number. A debug line is prefixed with DebugDelimiter followed by a space.

Parameters

Parameters

|    |       |                                |
|----|-------|--------------------------------|
| in | entry | The text of the debug message. |
|----|-------|--------------------------------|

Exceptions

|                                      |                                 |
|--------------------------------------|---------------------------------|
| <i>Error::StrategyError</i> (p. 567) | An error occurred when logging. |
|--------------------------------------|---------------------------------|

Reimplemented in **BiometricEvaluation::IO::FileLogsheet** (p. 324), and **BiometricEvaluation::IO::SysLogsheet** (p. 574).

G.76.5 Member Data Documentation

G.76.5.1 CommentDelimiter

```
const char BiometricEvaluation::IO::Logsheet::CommentDelimiter = '#' [static]
```

Delimiter for a comment line in the log sheet.

G.76.5.2 DebugDelimiter

```
const char BiometricEvaluation::IO::Logsheet::DebugDelimiter = 'D' [static]
```

Delimiter for an debug line in the log sheet.

G.76.5.3 DescriptionTag

```
const std::string BiometricEvaluation::IO::Logsheet::DescriptionTag [static]
```

The tag for the description string.

G.76.5.4 EntryDelimiter

```
const char BiometricEvaluation::IO::Logsheet::EntryDelimiter = 'E' [static]
```

Delimiter for an entry line in the log sheet.

G.76.5.5 FILEURLSCHEME

```
const std::string BiometricEvaluation::IO::Logsheet::FILEURLSCHEME [static]
```

The URL scheme to be used for **FileLogsheet** (p. 319) URL strings.

G.76.5.6 SYSLOGURLSCHEME

```
const std::string BiometricEvaluation::IO::Logsheet::SYSLOGURLSCHEME [static]
```

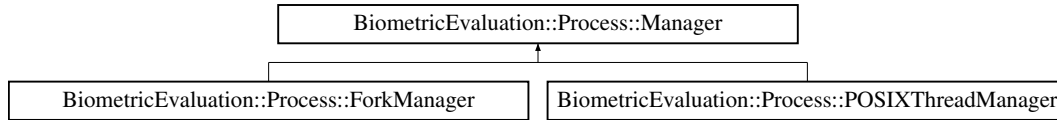
The URL scheme to be used for **SysLogsheet** (p. 569) URL strings.

## G.77 BiometricEvaluation::Process::Manager Class Reference

An interface for intranode process management classes.

```
#include <be_process_manager.h>
```

Inheritance diagram for BiometricEvaluation::Process::Manager:



### Public Member Functions

- **Manager** ()  
*Manager* (p. 430) constructor.
- virtual std::shared\_ptr< **WorkerController** > **addWorker** (std::shared\_ptr< **Worker** > worker)=0  
*Adds a Worker* (p. 595) *to be managed by this Manager* (p. 430).
- virtual uint32\_t **getNumCompletedWorkers** () const  
*Obtain the number of Workers that have exited.*
- virtual uint32\_t **getNumActiveWorkers** () const  
*Obtain the number of Workers that are still working.*
- virtual uint32\_t **getTotalWorkers** () const  
*Obtain the number of Workers this class is handling.*
- virtual void **startWorkers** (bool wait=true, bool communicate=false)=0  
*Begin Worker* (p. 595)'s work.
- virtual void **startWorker** (std::shared\_ptr< **WorkerController** > worker, bool wait=true, bool communicate=false)=0  
*Start a Worker* (p. 595).
- virtual void **waitForWorkerExit** ()=0  
*Block until all Workers have exited.*
- virtual void **reset** ()  
*Reuse all Workers.*
- virtual void **stopWorker** (std::shared\_ptr< **WorkerController** > worker)=0  
*Ask Worker* (p. 595) *to return as soon as possible.*
- virtual bool **waitForMessage** (std::shared\_ptr< **WorkerController** > &sender, int \*nextFD=nullptr, int numSeconds=-1) const  
*Wait for a message from a Worker* (p. 595).
- virtual bool **getNextMessage** (std::shared\_ptr< **WorkerController** > &sender, **Memory::uint8Array** &message, int numSeconds=-1) const  
*Obtain a message from a Worker* (p. 595).
- virtual void **broadcastMessage** (**Memory::uint8Array** &message) const  
*Send one message to all Workers.*
- virtual ~**Manager** ()  
*Manager* (p. 430) destructor.



## Protected Member Functions

- virtual void `_wait ()=0`

*Do not return until all spawned processes exited.*

## Protected Attributes

- `std::vector< std::shared_ptr< WorkerController > > _workers`
- `std::vector< std::shared_ptr< WorkerController > > _pendingExit`

### G.77.1 Detailed Description

An interface for intranode process management classes.

### G.77.2 Member Function Documentation

#### G.77.2.1 addWorker()

```
virtual std::shared_ptr< WorkerController> BiometricEvaluation::Process::Manager::addWorker (
 std::shared_ptr< Worker > worker) [pure virtual]
```

Adds a **Worker** (p. 595) to be managed by this **Manager** (p. 430).

Parameters

|               |                                           |
|---------------|-------------------------------------------|
| <i>worker</i> | A <b>Worker</b> (p. 595) instance to run. |
|---------------|-------------------------------------------|

Returns

`shared_ptr` to worker.

Implemented in **BiometricEvaluation::Process::ForkManager** (p. 336), and **BiometricEvaluation::Process::POSIXThreadManager** (p. 481).

#### G.77.2.2 broadcastMessage()

```
virtual void BiometricEvaluation::Process::Manager::broadcastMessage (
 Memory::uint8Array & message) const [virtual]
```

Send one message to all Workers.

Parameters

|                |                                     |
|----------------|-------------------------------------|
| <i>message</i> | The message to send to all Workers. |
|----------------|-------------------------------------|

Exceptions

|                                      |                                                                             |
|--------------------------------------|-----------------------------------------------------------------------------|
| <b>Error::StrategyError</b> (p. 567) | <b>Error</b> (p. 108) propagated from the <b>WorkerController</b> (p. 601). |
|--------------------------------------|-----------------------------------------------------------------------------|

### G.77.2.3 getNextMessage()

```
virtual bool BiometricEvaluation::Process::Manager::getNextMessage (
 std::shared_ptr< WorkerController > & sender,
 Memory::uint8Array & message,
 int numSeconds = -1) const [virtual]
```

Obtain a message from a **Worker** (p. 595).

#### Parameters

|     |                   |                                                                                              |
|-----|-------------------|----------------------------------------------------------------------------------------------|
| out | <i>sender</i>     | Reference to a shared pointer of the <b>WorkerController</b> (p. 601) that sent the message. |
| out | <i>message</i>    | Reference to a buffer to hold the message.                                                   |
| in  | <i>numSeconds</i> | Number of seconds to wait for a message, or < 0 to block.                                    |

#### Returns

true if there is a message, false otherwise.

#### Exceptions

|                                           |                                          |
|-------------------------------------------|------------------------------------------|
| <b>Error::ObjectDoesNotExist</b> (p. 457) | (Unexpected) widowed pipe.               |
| <b>Error::StrategyError</b> (p. 567)      | <b>Error</b> (p. 108) receiving message. |

### G.77.2.4 getNumActiveWorkers()

```
virtual uint32_t BiometricEvaluation::Process::Manager::getNumActiveWorkers () const [virtual]
```

Obtain the number of Workers that are still working.

#### Returns

The number of Workers that are still working.

#### Exceptions

|                                      |                                      |
|--------------------------------------|--------------------------------------|
| <b>Error::StrategyError</b> (p. 567) | No Workers have started working yet. |
|--------------------------------------|--------------------------------------|

### G.77.2.5 getNumCompletedWorkers()

```
virtual uint32_t BiometricEvaluation::Process::Manager::getNumCompletedWorkers () const [virtual]
```

Obtain the number of Workers that have exited.

## Returns

The number of Workers that have exited.

## Exceptions

|                                      |                                      |
|--------------------------------------|--------------------------------------|
| <b>Error::StrategyError</b> (p. 567) | No Workers have started working yet. |
|--------------------------------------|--------------------------------------|

**G.77.2.6 getTotalWorkers()**

```
virtual uint32_t BiometricEvaluation::Process::Manager::getTotalWorkers () const [virtual]
```

Obtain the number of Workers this class is handling.

## Returns

Number of Workers.

**G.77.2.7 reset()**

```
virtual void BiometricEvaluation::Process::Manager::reset () [virtual]
```

Reuse all Workers.

## Exceptions

|                                     |                                                       |
|-------------------------------------|-------------------------------------------------------|
| <b>Error::ObjectExists</b> (p. 458) | At least one <b>Worker</b> (p. 595) is still working. |
|-------------------------------------|-------------------------------------------------------|

**G.77.2.8 startWorker()**

```
virtual void BiometricEvaluation::Process::Manager::startWorker (
 std::shared_ptr< WorkerController > worker,
 bool wait = true,
 bool communicate = false) [pure virtual]
```

Start a **Worker** (p. 595).

## Parameters

|    |                    |                                                                                                               |
|----|--------------------|---------------------------------------------------------------------------------------------------------------|
|    | <i>worker</i>      | Pointer to a <b>WorkerController</b> (p. 601) that is being managed by this <b>Manager</b> (p. 430) instance. |
|    | <i>wait</i>        | Whether or not to wait for this <b>Worker</b> (p. 595) to exit before returning control to the caller.        |
| in | <i>communicate</i> | Whether or not to enable communication among the Workers and Managers.                                        |

## Exceptions

|                                     |                            |
|-------------------------------------|----------------------------|
| <b>Error::ObjectExists</b> (p. 458) | worker is already working. |
|-------------------------------------|----------------------------|

## Exceptions

|                                      |                                                                 |
|--------------------------------------|-----------------------------------------------------------------|
| <b>Error::StrategyError</b> (p. 567) | worker is not managed by this <b>Manager</b> (p. 430) instance. |
|--------------------------------------|-----------------------------------------------------------------|

## Note

Some implementations of this interface may call the system exit function from this routine. Therefore, the application's implementation of workerMain() should release all resources before returning.

Implemented in **BiometricEvaluation::Process::ForkManager** (p. 339), and **BiometricEvaluation::Process::POSIXThreadManager** (p. 482).

**G.77.2.9 startWorkers()**

```
virtual void BiometricEvaluation::Process::Manager::startWorkers (
 bool wait = true,
 bool communicate = false) [pure virtual]
```

Begin **Worker** (p. 595)'s work.

## Parameters

|    |                    |                                                                        |
|----|--------------------|------------------------------------------------------------------------|
| in | <i>wait</i>        | Whether or not to wait for all Workers to return before returning.     |
| in | <i>communicate</i> | Whether or not to enable communication among the Workers and Managers. |

## Exceptions

|                                      |                                                         |
|--------------------------------------|---------------------------------------------------------|
| <b>Error::ObjectExists</b> (p. 458)  | At least one <b>Worker</b> (p. 595) is already working. |
| <b>Error::StrategyError</b> (p. 567) | Problem starting Workers.                               |

Implemented in **BiometricEvaluation::Process::ForkManager** (p. 339), and **BiometricEvaluation::Process::POSIXThreadManager** (p. 482).

**G.77.2.10 stopWorker()**

```
virtual void BiometricEvaluation::Process::Manager::stopWorker (
 std::shared_ptr< WorkerController > worker) [pure virtual]
```

Ask **Worker** (p. 595) to return as soon as possible.

## Parameters

|               |                                                                         |
|---------------|-------------------------------------------------------------------------|
| <i>worker</i> | Pointer to the <b>WorkerController</b> (p. 601) that should be stopped. |
|---------------|-------------------------------------------------------------------------|

## Exceptions

|                                           |                                |
|-------------------------------------------|--------------------------------|
| <b>Error::ObjectDoesNotExist</b> (p. 457) | worker is not working.         |
| <b>Error::StrategyError</b> (p. 567)      | Problem asking worker to stop. |

Implemented in **BiometricEvaluation::Process::ForkManager** (p. 340), and **BiometricEvaluation::Process::POSIXThreadManager** (p. 483).

### G.77.2.11 waitForMessage()

```
virtual bool BiometricEvaluation::Process::Manager::waitForMessage (
 std::shared_ptr< WorkerController > & sender,
 int * nextFD = nullptr,
 int numSeconds = -1) const [virtual]
```

Wait for a message from a **Worker** (p. 595).

Parameters

|         |                   |                                                                                              |
|---------|-------------------|----------------------------------------------------------------------------------------------|
| out     | <i>sender</i>     | Reference to a shared pointer of the <b>WorkerController</b> (p. 601) that sent the message. |
| in, out | <i>nextFD</i>     | Location to store a pipe that has data to read.                                              |
| in      | <i>numSeconds</i> | Number of seconds to wait for a message, or < 0 to block.                                    |

Returns

true if there is a **Worker** (p. 595) sending a message false otherwise or if an error occurred.

### G.77.2.12 waitForWorkerExit()

```
virtual void BiometricEvaluation::Process::Manager::waitForWorkerExit () [pure virtual]
```

Block until all Workers have exited.

Use this method if wait=false was set during a call to startWorker(s) but now wait=true is desired.

Implemented in **BiometricEvaluation::Process::ForkManager** (p. 340), and **BiometricEvaluation::Process::POSIXThreadManager** (p. 483).

## G.77.3 Member Data Documentation

### G.77.3.1 \_pendingExit

```
std::vector<std::shared_ptr< WorkerController > > BiometricEvaluation::Process::Manager::_pendingExit [protected]
```

Workers that are about to exit (stop requested).

### G.77.3.2 \_workers

```
std::vector<std::shared_ptr< WorkerController > > BiometricEvaluation::Process::Manager::_workers [protected]
```

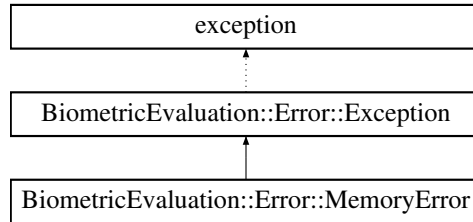
Workers that have been added.

## G.78 BiometricEvaluation::Error::MemoryError Class Reference

An error occurred when allocating an object.

```
#include <be_error_exception.h>
```

Inheritance diagram for BiometricEvaluation::Error::MemoryError:



### Public Member Functions

- **MemoryError** ()
- **MemoryError** (const std::string &info)

### G.78.1 Detailed Description

An error occurred when allocating an object.

### G.78.2 Constructor & Destructor Documentation

#### G.78.2.1 MemoryError() [1/2]

```
BiometricEvaluation::Error::MemoryError::MemoryError ()
```

Construct a **MemoryError** (p. 436) object with the default information string.

#### G.78.2.2 MemoryError() [2/2]

```
BiometricEvaluation::Error::MemoryError::MemoryError (
 const std::string & info)
```

Construct a **MemoryError** (p. 436) object with an information string appended to the default information string.

## G.79 BiometricEvaluation::Process::MessageCenter Class Reference

```
#include <be_process_messagecenter.h>
```

### Public Member Functions

- **MessageCenter** (uint32\_t port= **MessageCenter::DEFAULT\_PORT**)  
*Constructor.*
- bool **hasUnseenMessages** () const  
*Determine whether or not there are unseen messages.*
- bool **getNextMessage** (uint32\_t &clientId, **Memory::uint8Array** &message, int numSeconds=-1)

- *Get the next available message.*
- void **sendResponse** (uint32\_t clientID, const **Memory::uint8Array** &message) const  
*Send a message to a client.*
- void **disconnectClient** (uint32\_t clientID)  
*Break the connection with a client.*

## Static Public Attributes

- static const int **CONNECTION\_BACKLOG** = 10
- static const uint16\_t **DEFAULT\_PORT** = 7899
- static const int **DEFAULT\_TIMEOUT** = 1
- static const uint64\_t **MAX\_MESSAGE\_LENGTH** = 255

## G.79.1 Detailed Description

Convenience for asynchronous TCP socket message passing.

## G.79.2 Constructor & Destructor Documentation

### G.79.2.1 MessageCenter()

```
BiometricEvaluation::Process::MessageCenter::MessageCenter (
 uint32_t port = MessageCenter::DEFAULT_PORT)
```

Constructor.

Parameters

|             |                 |
|-------------|-----------------|
| <i>port</i> | Listening port. |
|-------------|-----------------|

## G.79.3 Member Function Documentation

### G.79.3.1 disconnectClient()

```
void BiometricEvaluation::Process::MessageCenter::disconnectClient (
 uint32_t clientID)
```

Break the connection with a client.

Parameters

|                 |                                 |
|-----------------|---------------------------------|
| <i>clientID</i> | ID of the client to disconnect. |
|-----------------|---------------------------------|

### G.79.3.2 getNextMessage()

```
bool BiometricEvaluation::Process::MessageCenter::getNextMessage (
```

```
uint32_t & clientID,
 Memory::uint8Array & message,
 int numSeconds = -1)
```

Get the next available message.

Parameters

|         |                   |                                                                        |
|---------|-------------------|------------------------------------------------------------------------|
| out     | <i>clientID</i>   | ID of the client that sent the message.                                |
| in, out | <i>message</i>    | Message received.                                                      |
| in      | <i>numSeconds</i> | Number of seconds to wait for a message, or < 0 to block indefinitely. |

Returns

true if a message was received before timing out.

### G.79.3.3 hasUnseenMessages()

```
bool BiometricEvaluation::Process::MessageCenter::hasUnseenMessages () const
```

Determine whether or not there are unseen messages.

Returns

true if a message has been received and not read.

Note

Returns immediately.

### G.79.3.4 sendResponse()

```
void BiometricEvaluation::Process::MessageCenter::sendResponse (
 uint32_t clientID,
 const Memory::uint8Array & message) const
```

Send a message to a client.

Parameters

|                 |                                  |
|-----------------|----------------------------------|
| <i>clientID</i> | ID of client to receive message. |
| <i>message</i>  | Message to send client.          |

## G.79.4 Member Data Documentation

### G.79.4.1 CONNECTION\_BACKLOG

```
const int BiometricEvaluation::Process::MessageCenter::CONNECTION_BACKLOG = 10 [static]
```



Number of outstanding connections.

#### G.79.4.2 DEFAULT\_PORT

```
const uint16_t BiometricEvaluation::Process::MessageCenter::DEFAULT_PORT = 7899 [static]
```

Default port used for messages.

#### G.79.4.3 DEFAULT\_TIMEOUT

```
const int BiometricEvaluation::Process::MessageCenter::DEFAULT_TIMEOUT = 1 [static]
```

Default number of seconds to wait between polls.

#### G.79.4.4 MAX\_MESSAGE\_LENGTH

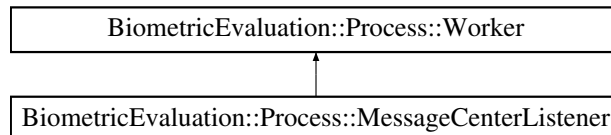
```
const uint64_t BiometricEvaluation::Process::MessageCenter::MAX_MESSAGE_LENGTH = 255 [static]
```

Maximum length of a message.

## G.80 BiometricEvaluation::Process::MessageCenterListener Class Reference

```
#include <be.process.mcllistener.h>
```

Inheritance diagram for BiometricEvaluation::Process::MessageCenterListener:



### Public Member Functions

- `int32_t workerMain ()`

*The method that will get called to start execution by a ProcessManager.*

### Static Public Attributes

- `static const std::string PARAM_PORT`

### Additional Inherited Members

#### G.80.1 Detailed Description

Accepts new connections and spawns message receivers.

#### G.80.2 Member Function Documentation

### G.80.2.1 workerMain()

```
int32_t BiometricEvaluation::Process::MessageCenterListener::workerMain () [virtual]
```

The method that will get called to start execution by a ProcessManager.

Returns

Status code.

Note

If an object of this class is added to a **Process::ForkManager** (p. 335) object, the implementation of **Process::Worker::workerMain()** (p. 600) should release all resources prior to returning.

Any exceptions thrown by this method will cause the worker to exit with a return status of EXIT\_FAILURE. The type and contents of the exception is not maintained.

Implements **BiometricEvaluation::Process::Worker** (p. 600).

## G.80.3 Member Data Documentation

### G.80.3.1 PARAM\_PORT

```
const std::string BiometricEvaluation::Process::MessageCenterListener::PARAM_PORT [static]
```

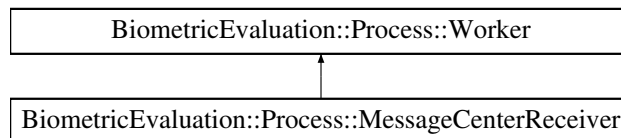
Parameter used to pass port number

## G.81 BiometricEvaluation::Process::MessageCenterReceiver Class Reference

Receives message from a client, forwarding to the central **MessageCenter** (p. 436).

```
#include <be_process_mcreceiver.h>
```

Inheritance diagram for BiometricEvaluation::Process::MessageCenterReceiver:



### Public Member Functions

- `int32_t workerMain ()`
- `MessageCenterReceiver ()=default`
- `~MessageCenterReceiver ()=default`

### Static Public Attributes

- `static const std::string PARAM_CLIENT_SOCKET`
- `static const std::string PARAM_CLIENT_ID`
- `static const std::string MSG_DISCONNECT`

## Additional Inherited Members

### G.81.1 Detailed Description

Receives message from a client, forwarding to the central **MessageCenter** (p. [436](#)).

### G.81.2 Constructor & Destructor Documentation

#### G.81.2.1 MessageCenterReceiver()

```
BiometricEvaluation::Process::MessageCenterReceiver::MessageCenterReceiver () [default]
```

Default constructor.

#### G.81.2.2 ~MessageCenterReceiver()

```
BiometricEvaluation::Process::MessageCenterReceiver::~~MessageCenterReceiver () [default]
```

Default destructor.

### G.81.3 Member Function Documentation

#### G.81.3.1 workerMain()

```
int32_t BiometricEvaluation::Process::MessageCenterReceiver::workerMain () [virtual]
```

Receive loop.

Implements **BiometricEvaluation::Process::Worker** (p. [600](#)).

### G.81.4 Member Data Documentation

#### G.81.4.1 MSG\_DISCONNECT

```
const std::string BiometricEvaluation::Process::MessageCenterReceiver::MSG_DISCONNECT [static]
```

Message sent when client should disconnect.

#### G.81.4.2 PARAM\_CLIENT\_ID

```
const std::string BiometricEvaluation::Process::MessageCenterReceiver::PARAM_CLIENT_ID [static]
```

Parameter used to pass an ID to the client.

#### G.81.4.3 PARAM\_CLIENT\_SOCKET

```
const std::string BiometricEvaluation::Process::MessageCenterReceiver::PARAM_CLIENT_SOCKET [static]
```

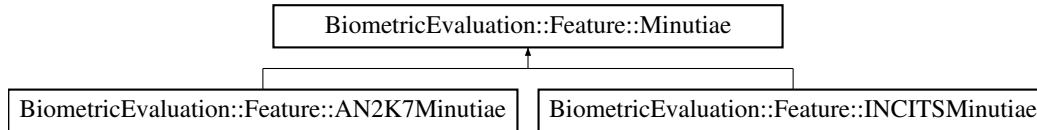
Parameter used to pass client socket FD.

## G.82 BiometricEvaluation::Feature::Minutiae Class Reference

A class to represent a set of minutiae data points.

```
#include <be_feature_minutiae.h>
```

Inheritance diagram for BiometricEvaluation::Feature::Minutiae:



### Public Member Functions

- virtual **MinutiaeFormat** **getFormat** () const =0  
*Obtain the minutiae format kind.*
- virtual MinutiaPointSet **getMinutiaPoints** () const =0  
*Obtain the set of finger minutiae data points. The set may be empty.*
- virtual RidgeCountItemSet **getRidgeCountItems** () const =0  
*Obtain the set of ridge count data items. The set may be empty.*
- virtual CorePointSet **getCores** () const =0  
*Obtains the set of core positions. The set may be empty.*
- virtual DeltaPointSet **getDeltas** () const =0  
*Obtains the set of delta positions. The set may be empty.*

### G.82.1 Detailed Description

A class to represent a set of minutiae data points.

Each set includes the core and delta data points, if they are included in the source record. This class represents an interface that subclasses of this class will implement, providing more information on the minutiae that is specific to the record format represented by that class.

## G.83 BiometricEvaluation::Feature::AN2K11EFS::MinutiaeRidgeCount Struct Reference

Representation of an extended feature set ridge count info.

```
#include <be_feature_an2k11efs.h>
```

### Public Attributes

- int **mia**
- int **mib**
- int **mir**
- bool **has\_mrn**
- int **mrn**
- bool **has\_mrs**
- int **mrs**

### G.83.1 Detailed Description

Representation of an extended feature set ridge count info.

### G.83.2 Member Data Documentation

#### G.83.2.1 mia

`int BiometricEvaluation::Feature::AN2K11EFS::MinutiaeRidgeCount::mia`  
minutia index A

#### G.83.2.2 mib

`int BiometricEvaluation::Feature::AN2K11EFS::MinutiaeRidgeCount::mib`  
minutia index B

#### G.83.2.3 mir

`int BiometricEvaluation::Feature::AN2K11EFS::MinutiaeRidgeCount::mir`  
ridge count

#### G.83.2.4 mrn

`int BiometricEvaluation::Feature::AN2K11EFS::MinutiaeRidgeCount::mrn`  
reference number, optional

#### G.83.2.5 mrs

`int BiometricEvaluation::Feature::AN2K11EFS::MinutiaeRidgeCount::mrs`  
residual, optional

## G.84 BiometricEvaluation::Feature::AN2K11EFS::MinutiaeRidgeCountConfidence Struct Reference↵

Representation of an extended feature set minutiae ridge count confidence item.

```
#include <be_feature_an2k11efs.h>
```

### Public Attributes

- `Image::Coordinate pointA`
- `Image::Coordinate pointB`
- `MethodOfRidgeCounting morc`
- `int mcv`

### G.84.1 Detailed Description

Representation of an extended feature set minutiae ridge count confidence item.

## G.85 BiometricEvaluation::Feature::AN2K11EFS::MinutiaeRidgeCount Info Struct Reference

All the ridge count information in one place.

```
#include <be_feature_an2k11efs.h>
```

### Public Attributes

- bool **has\_mra**
- MinutiaeRidgeCountAlgorithm **mra**
- bool **has\_mrca**
- MinutiaeRidgeCountSet **mrca**
- bool **has\_rcca**
- MinutiaeRidgeCountConfidenceSet **rcca**

### G.85.1 Detailed Description

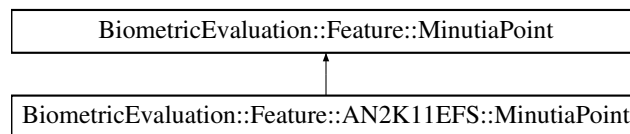
All the ridge count information in one place.

## G.86 BiometricEvaluation::Feature::MinutiaPoint Struct Reference

Representation of a finger minutiae data point.

```
#include <be_feature_minutiae.h>
```

Inheritance diagram for BiometricEvaluation::Feature::MinutiaPoint:



### Public Attributes

- unsigned int **index**
- bool **has\_type**
- **MinutiaeType** **type**
- **Image::Coordinate** **coordinate**
- unsigned int **theta**
- bool **has\_quality**
- unsigned int **quality**

### G.86.1 Detailed Description

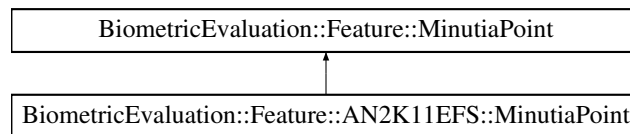
Representation of a finger minutiae data point.

## G.87 BiometricEvaluation::Feature::AN2K11EFS::MinutiaPoint Struct Reference

Representation of an extended feature set minutia data point.

```
#include <be_feature_an2k11efs.h>
```

Inheritance diagram for BiometricEvaluation::Feature::AN2K11EFS::MinutiaPoint:



### Public Attributes

- bool **has\_mru**
- int **mru**
- bool **has\_mdu**
- int **mdu**

### G.87.1 Detailed Description

Representation of an extended feature set minutia data point.

### G.87.2 Member Data Documentation

#### G.87.2.1 mdu

```
int BiometricEvaluation::Feature::AN2K11EFS::MinutiaPoint::mdu
 minutiae direction uncertainty
```

#### G.87.2.2 mru

```
int BiometricEvaluation::Feature::AN2K11EFS::MinutiaPoint::mru
 radius of position uncertainty
```

## G.88 BiometricEvaluation::Feature::MPEGFacePoint Struct Reference

Representation of a feature point and a set of points.

```
#include <be_feature_mpegfacepoint.h>
```

### Public Attributes

- uint8\_t **type**
- uint8\_t **major**
- uint8\_t **minor**
- **BiometricEvaluation::Image::Coordinate** **coordinate**

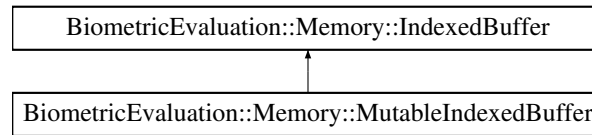
### G.88.1 Detailed Description

Representation of a feature point and a set of points.

## G.89 BiometricEvaluation::Memory::MutableIndexedBuffer Class Reference

```
#include <be_memory_mutableindexedbuffer.h>
```

Inheritance diagram for BiometricEvaluation::Memory::MutableIndexedBuffer:



### Public Member Functions

- **MutableIndexedBuffer** (uint8\_t \*data, uint64\_t size)  
*Wrap an existing buffer of a given length.*
- **MutableIndexedBuffer** ( uint8Array &aa)  
*Wrap an existing uint8Array.*
- **MutableIndexedBuffer** (const **MutableIndexedBuffer** &copy)=default
- uint64\_t **push** (const void \*buf, uint64\_t len)  
*Push elements into the buffer, increasing the index.*
- uint8\_t **pushU8Val** (uint8\_t val)  
*Push an element into the managed buffer at the current index, incrementing the index.*
- uint16\_t **pushU16Val** (uint16\_t val)  
*Push two elements into the managed buffer at the current index, incrementing the index.*
- uint16\_t **pushBeU16Val** (uint16\_t val)  
*Push two elements into the managed buffer at the current index as a big endian value, incrementing the index.*
- uint32\_t **pushU32Val** (uint32\_t val)  
*Push four elements into the managed buffer at the current index, incrementing the index.*
- uint32\_t **pushBeU32Val** (uint32\_t val)  
*Push four elements into the managed buffer at the current index as a big endian value, incrementing the index.*
- uint64\_t **pushU64Val** (uint64\_t val)  
*Push eight elements into the managed buffer at the current index, incrementing the index.*
- virtual const uint8\_t \* **get** () const  
*Returns a pointer to the managed buffer.*
- virtual ~**MutableIndexedBuffer** ()=default

### G.89.1 Detailed Description

Mutable version of an **IndexedBuffer** (p. 395).

### G.89.2 Constructor & Destructor Documentation



G.89.2.1 MutableIndexedBuffer() [1/3]

```
BiometricEvaluation::Memory::MutableIndexedBuffer::MutableIndexedBuffer (
 uint8_t * data,
 uint64_t size)
```

Wrap an existing buffer of a given length.

Parameters

|             |                 |
|-------------|-----------------|
| <i>data</i> | Buffer to wrap. |
| <i>size</i> | Size of buffer. |

G.89.2.2 MutableIndexedBuffer() [2/3]

```
BiometricEvaluation::Memory::MutableIndexedBuffer::MutableIndexedBuffer (
 uint8Array & aa)
```

Wrap an existing uint8Array.

Parameters

|           |                     |
|-----------|---------------------|
| <i>aa</i> | uint8Array to wrap. |
|-----------|---------------------|

G.89.2.3 MutableIndexedBuffer() [3/3]

```
BiometricEvaluation::Memory::MutableIndexedBuffer::MutableIndexedBuffer (
 const MutableIndexedBuffer & copy) [default]
```

Copy constructor (default).

G.89.2.4 ~MutableIndexedBuffer()

```
virtual BiometricEvaluation::Memory::MutableIndexedBuffer::~~MutableIndexedBuffer () [virtual],
[default]
```

Destructor (default).

G.89.3 Member Function Documentation

G.89.3.1 get()

```
virtual const uint8_t* BiometricEvaluation::Memory::MutableIndexedBuffer::get () const [virtual]
```

Returns a pointer to the managed buffer.

Returns

Pointer to the managed buffer.

Reimplemented from **BiometricEvaluation::Memory::IndexedBuffer** (p. 397).

**G.89.3.2 push()**

```
uint64_t BiometricEvaluation::Memory::MutableIndexedBuffer::push (
 const void * buf,
 uint64_t len)
```

Push elements into the buffer, increasing the index.

Parameters

|    |            |                                                     |
|----|------------|-----------------------------------------------------|
| in | <i>buf</i> | The buffer to push. If nullptr, 0 will be inserted. |
| in | <i>len</i> | The number of elements from buf to copy.            |

Exceptions

|                                         |                                       |
|-----------------------------------------|---------------------------------------|
| <b><i>Error::DataError</i></b> (p. 295) | Not enough room to copy len elements. |
|-----------------------------------------|---------------------------------------|

Returns

The number of elements copied.

**G.89.3.3 pushBeU16Val()**

```
uint16_t BiometricEvaluation::Memory::MutableIndexedBuffer::pushBeU16Val (
 uint16_t val)
```

Push two elements into the managed buffer at the current index as a big endian value, incrementing the index.

Parameters

|            |                |
|------------|----------------|
| <i>val</i> | Value to push. |
|------------|----------------|

Exceptions

|                                         |                                       |
|-----------------------------------------|---------------------------------------|
| <b><i>Error::DataError</i></b> (p. 295) | Not enough room to copy the elements. |
|-----------------------------------------|---------------------------------------|

Returns

The number of elements copied (2).

**G.89.3.4 pushBeU32Val()**

```
uint32_t BiometricEvaluation::Memory::MutableIndexedBuffer::pushBeU32Val (
 uint32_t val)
```

Push four elements into the managed buffer at the current index as a big endian value, incrementing the index.

Parameters

|            |                |
|------------|----------------|
| <i>val</i> | Value to push. |
|------------|----------------|

Exceptions

|                                  |                                       |
|----------------------------------|---------------------------------------|
| <b>Error::DataError</b> (p. 295) | Not enough room to copy the elements. |
|----------------------------------|---------------------------------------|

Returns

The number of elements copied (4).

G.89.3.5 pushU16Val()

```
uint16_t BiometricEvaluation::Memory::MutableIndexedBuffer::pushU16Val (
 uint16_t val)
```

Push two elements into the managed buffer at the current index, incrementing the index.

Parameters

|            |                |
|------------|----------------|
| <i>val</i> | Value to push. |
|------------|----------------|

Exceptions

|                                  |                                       |
|----------------------------------|---------------------------------------|
| <b>Error::DataError</b> (p. 295) | Not enough room to copy the elements. |
|----------------------------------|---------------------------------------|

Returns

The number of elements copied (2).

G.89.3.6 pushU32Val()

```
uint32_t BiometricEvaluation::Memory::MutableIndexedBuffer::pushU32Val (
 uint32_t val)
```

Push four elements into the managed buffer at the current index, incrementing the index.

Parameters

|            |                |
|------------|----------------|
| <i>val</i> | Value to push. |
|------------|----------------|

Exceptions

|                                  |                                       |
|----------------------------------|---------------------------------------|
| <b>Error::DataError</b> (p. 295) | Not enough room to copy the elements. |
|----------------------------------|---------------------------------------|

## Returns

The number of elements copied (4).

**G.89.3.7 pushU64Val()**

```
uint64_t BiometricEvaluation::Memory::MutableIndexedBuffer::pushU64Val (
 uint64_t val)
```

Push eight elements into the managed buffer at the current index, incrementing the index.

## Parameters

|            |                |
|------------|----------------|
| <i>val</i> | Value to push. |
|------------|----------------|

## Exceptions

|                                  |                                       |
|----------------------------------|---------------------------------------|
| <b>Error::DataError</b> (p. 295) | Not enough room to copy the elements. |
|----------------------------------|---------------------------------------|

## Returns

The number of elements copied (8).

**G.89.3.8 pushU8Val()**

```
uint8_t BiometricEvaluation::Memory::MutableIndexedBuffer::pushU8Val (
 uint8_t val)
```

Push an element into the managed buffer at the current index, incrementing the index.

## Parameters

|            |                |
|------------|----------------|
| <i>val</i> | Value to push. |
|------------|----------------|

## Exceptions

|                                  |                                      |
|----------------------------------|--------------------------------------|
| <b>Error::DataError</b> (p. 295) | Not enough room to copy the element. |
|----------------------------------|--------------------------------------|

## Returns

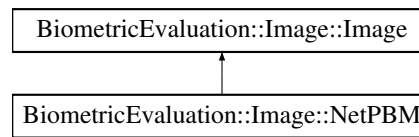
The number of elements copied (1).

**G.90 BiometricEvaluation::Image::NetPBM Class Reference**

A NetPBM-encoded image.

```
#include <be_image_netpbm.h>
```

Inheritance diagram for BiometricEvaluation::Image::NetPBM:



## Public Types

- enum **Kind** {  
**ASCIIPortableBitmap** = 1, **ASCIIPortableGraymap** = 2, **ASCIIPortablePixmap** = 3, **BinaryPortableBitmap** = 4,  
**BinaryPortableGraymap** = 5, **BinaryPortablePixmap** = 6 }

## Public Member Functions

- NetPBM** (const uint8\_t \*data, const uint64\_t size)
- NetPBM** (const **Memory::uint8Array** &data)
- Memory::uint8Array** **getRawData** () const  
*Accessor for the raw image data. The data returned should not be compressed or encoded.*
- Memory::uint8Array** **getRawGrayscaleData** (uint8\_t depth) const  
*Accessor for decompressed data in grayscale.*

## Static Public Member Functions

- static bool **isNetPBM** (const uint8\_t \*data, uint64\_t size)
- static void **skipLine** (const uint8\_t \*data, size\_t dataSize, size\_t &offset)  
*Skip an entire line of input, placing offset at the first character after the newline.*
- static void **skipComment** (const uint8\_t \*data, size\_t dataSize, size\_t &offset)  
*Skip a block of comments in input.*
- static std::string **getNextValue** (const uint8\_t \*data, size\_t dataSize, size\_t &offset, size\_t sizeOfValue=0)  
*Obtain the next space-separated value from data, beginning at offset.*
- static **Memory::uint8Array** **ASCIIBitmapTo8Bit** (const uint8\_t \*bitmap, uint64\_t bitmapSize, uint32\_t width, uint32\_t height)  
*Convert an ASCII bitmap (1-bit depth) buffer into an 8-bit depth buffer.*
- static **Memory::uint8Array** **ASCIIPixmapToBinaryPixmap** (const uint8\_t \*ASCIIBuf, uint64\_t ASCIIBufSize, uint32\_t width, uint32\_t height, uint8\_t depth, uint32\_t maxColor)  
*Convert an ASCII pixel map buffer into a binary pixel map buffer.*
- static **Memory::uint8Array** **BinaryBitmapTo8Bit** (const uint8\_t \*bitmap, uint64\_t bitmapSize, uint32\_t width, uint32\_t height)  
*Convert an binary bitmap (1-bit depth) buffer into an 8-bit depth buffer.*

## Additional Inherited Members

### G.90.1 Detailed Description

A NetPBM-encoded image.

Note

While a **NetPBM** (p. 450) file can contain more than one image, this class will only support the first image found in any file, also known as the "plain" **NetPBM** (p. 450) format.

## G.90.2 Member Function Documentation

### G.90.2.1 ASCIIBitmapTo8Bit()

```
static Memory::uint8Array BiometricEvaluation::Image::NetPBM::ASCIIBitmapTo8Bit (
 const uint8_t * bitmap,
 uint64_t bitmapSize,
 uint32_t width,
 uint32_t height) [static]
```

Convert an ASCII bitmap (1-bit depth) buffer into an 8-bit depth buffer.

Parameters

|                   |                                 |
|-------------------|---------------------------------|
| <i>bitmap</i>     | Bitmap data buffer.             |
| <i>bitmapSize</i> | <b>Size</b> (p. 549) of bitmap. |
| <i>width</i>      | Width of image in bitmap.       |
| <i>height</i>     | Height of image in bitmap.      |

Returns

8-bit depth representation of bitmap

Exceptions

|                     |                                                           |
|---------------------|-----------------------------------------------------------|
| <i>out_of_range</i> | <b>Error</b> (p. 108) extracting a value from the bitmap. |
|---------------------|-----------------------------------------------------------|

### G.90.2.2 ASCIIPixmapToBinaryPixmap()

```
static Memory::uint8Array BiometricEvaluation::Image::NetPBM::ASCIIPixmapToBinaryPixmap (
 const uint8_t * ASCIIBuf,
 uint64_t ASCIIBufSize,
 uint32_t width,
 uint32_t height,
 uint8_t depth,
 uint32_t maxColor) [static]
```

Convert an ASCII pixel map buffer into a binary pixel map buffer.

Parameters

|                     |                                                                                |
|---------------------|--------------------------------------------------------------------------------|
| <i>ASCIIBuf</i>     | ASCII pixel map data buffer.                                                   |
| <i>ASCIIBufSize</i> | <b>Size</b> (p. 549) of <i>ASCIIBuf</i> .                                      |
| <i>width</i>        | Width of image in pixel map.                                                   |
| <i>height</i>       | Height of image in pixel map.                                                  |
| <i>depth</i>        | Depth of image in pixel map.                                                   |
| <i>maxColor</i>     | Maximum color value per pixel. Intensities will be scaled based on this value. |

## Returns

Binary pixel map representation of the ASCII pixel map in the same depth as the original.

## Exceptions

|                                       |                                                              |
|---------------------------------------|--------------------------------------------------------------|
| <i>out_of_range</i>                   | <b>Error</b> (p. 108) extracting a value from the pixel map. |
| <b>Error::ParameterError</b> (p. 474) | Invalid value for depth, must be a multiple of 8.            |

## G.90.2.3 BinaryBitmapTo8Bit()

```
static Memory::uint8Array BiometricEvaluation::Image::NetPBM::BinaryBitmapTo8Bit (
 const uint8_t * bitmap,
 uint64_t bitmapSize,
 uint32_t width,
 uint32_t height) [static]
```

Convert an binary bitmap (1-bit depth) buffer into an 8-bit depth buffer.

## Parameters

|                   |                                 |
|-------------------|---------------------------------|
| <i>bitmap</i>     | Bitmap data buffer.             |
| <i>bitmapSize</i> | <b>Size</b> (p. 549) of bitmap. |
| <i>width</i>      | Width of image in bitmap.       |
| <i>height</i>     | Height of image in bitmap.      |

## Returns

8-bit depth representation of bitmap

## Exceptions

|                     |                                                           |
|---------------------|-----------------------------------------------------------|
| <i>out_of_range</i> | <b>Error</b> (p. 108) extracting a value from the bitmap. |
|---------------------|-----------------------------------------------------------|

## G.90.2.4 getNextValue()

```
static std::string BiometricEvaluation::Image::NetPBM::getNextValue (
 const uint8_t * data,
 size_t dataSize,
 size_t & offset,
 size_t sizeOfValue = 0) [static]
```

Obtain the next space-separated value from data, beginning at offset.

## Parameters

|             |                                           |
|-------------|-------------------------------------------|
| <i>data</i> | Buffer where next value will be obtained. |
|-------------|-------------------------------------------|

## Parameters

|                    |                                                                                                                                                    |
|--------------------|----------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>dataSize</i>    | <b>Size</b> (p. 549) of data.                                                                                                                      |
| <i>offset</i>      | Current starting position within data.                                                                                                             |
| <i>sizeOfValue</i> | In the event that the values in data are not space-separated, return a value when it reaches <i>sizeOfValue</i> length. 0 assumes space-separated. |

## Returns

Next value from data.

**G.90.2.5 getRawData()**

**Memory::uint8Array** `BiometricEvaluation::Image::NetPBM::getRawData ( ) const [virtual]`  
 Accessor for the raw image data. The data returned should not be compressed or encoded.

## Returns

AutoArray holding raw image data.

## Exceptions

|                                       |                                                 |
|---------------------------------------|-------------------------------------------------|
| <b>Error::DataError</b> (p. 295)      | <b>Error</b> (p. 108) decompressing image data. |
| <b>Error::NotImplemented</b> (p. 456) | Compression type not supported.                 |

## Note

The raw data returned from this method is encoded at the same bit depth as the compressed data, except in the case of 1-bit (bitmap) images, which are expanded to 8-bit.

Implements **BiometricEvaluation::Image::Image** (p. 361).

**G.90.2.6 getRawGrayscaleData()**

**Memory::uint8Array** `BiometricEvaluation::Image::NetPBM::getRawGrayscaleData ( uint8_t depth ) const [virtual]`  
 Accessor for decompressed data in grayscale.

## Parameters

|              |                                                                                         |
|--------------|-----------------------------------------------------------------------------------------|
| <i>depth</i> | The desired bit depth of the resulting raw image. This value may either be 16, 8, or 1. |
|--------------|-----------------------------------------------------------------------------------------|

## Returns

AutoArray holding raw grayscale image data.



## Exceptions

|                                       |                                                     |
|---------------------------------------|-----------------------------------------------------|
| <b>Error::DataError</b> (p. 295)      | <b>Error</b> (p. 108) decompressing image data.     |
| <b>Error::NotImplemented</b> (p. 456) | Unsupported conversion based on source color depth. |
| <b>Error::ParameterError</b> (p. 474) | Invalid value for depth.                            |

## Note

This method does not save a cached copy of the decompressed image because the bit depth of the image can be changed between calls.

When depth is 1, this method returns an image that uses 8 bits to represent a single pixel. The depth parameter is used to adjust the number of gray levels. When depth is 1, there are only 2 gray levels (black and white), despite using 8 bits to represent each pixel.

Alpha channels are completely ignored when converting to grayscale.

Implements **BiometricEvaluation::Image::Image** (p. 362).

## G.90.2.7 isNetPBM()

```
static bool BiometricEvaluation::Image::NetPBM::isNetPBM (
 const uint8_t * data,
 uint64_t size) [static]
```

Whether or not data is a netpbm image.

## Parameters

|    |             |                      |
|----|-------------|----------------------|
| in | <i>data</i> | The buffer to check. |
| in | <i>size</i> | The size of data.    |

## Returns

true if data appears to be a netpbm image, false otherwise.

## G.90.2.8 skipComment()

```
static void BiometricEvaluation::Image::NetPBM::skipComment (
 const uint8_t * data,
 size_t dataSize,
 size_t & offset) [static]
```

Skip a block of comments in input.

## Parameters

|                 |                                                                      |
|-----------------|----------------------------------------------------------------------|
| <i>data</i>     | Buffer with comment to be skipped.                                   |
| <i>dataSize</i> | <b>Size</b> (p. 549) of data                                         |
| <i>offset</i>   | Position within data from which the rest of the line should be read. |

## Exceptions

|                     |                                                                         |
|---------------------|-------------------------------------------------------------------------|
| <i>out_of_range</i> | End of line not encountered before end of data or on last line of data. |
|---------------------|-------------------------------------------------------------------------|

**G.90.2.9 skipLine()**

```
static void BiometricEvaluation::Image::NetPBM::skipLine (
 const uint8_t * data,
 size_t dataSize,
 size_t & offset) [static]
```

Skip an entire line of input, placing offset at the first character after the newline.

## Parameters

|                 |                                                                      |
|-----------------|----------------------------------------------------------------------|
| <i>data</i>     | Buffer with line to be skipped.                                      |
| <i>dataSize</i> | <b>Size</b> (p. 549) of data.                                        |
| <i>offset</i>   | Position within data from which the rest of the line should be read. |

## Exceptions

|                     |                                                                         |
|---------------------|-------------------------------------------------------------------------|
| <i>out_of_range</i> | End of line not encountered before end of data or on last line of data. |
|---------------------|-------------------------------------------------------------------------|

**G.91 BiometricEvaluation::Feature::AN2K11EFS::NoFeaturesPresent Struct Reference**

A set of flags indicating "No features present" indicators contained within the extended feature set.

```
#include <be_feature_an2k11efs.h>
```

**Public Attributes**

- bool **cores**
- bool **deltas**
- bool **minutiae**

**G.91.1 Detailed Description**

A set of flags indicating "No features present" indicators contained within the extended feature set.

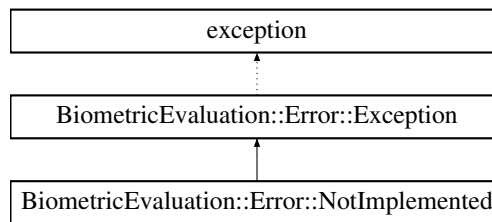
A flag is set to true when the Type-9 field is set to 'Y', indicating that analysis of the image has determined that there are no instances of that feature present in the image. Otherwise the Type-9 field is not present and the flag will be false.

**G.92 BiometricEvaluation::Error::NotImplemented Class Reference**

A **NotImplemented** (p. 456) object is thrown when the underlying implementation of this interface has not or could not be created.

```
#include <be_error_exception.h>
```

Inheritance diagram for BiometricEvaluation::Error::NotImplemented:



## Public Member Functions

- **NotImplemented** ()
- **NotImplemented** (const std::string &info)

### G.92.1 Detailed Description

A **NotImplemented** (p. 456) object is thrown when the underlying implementation of this interface has not or could not be created.

### G.92.2 Constructor & Destructor Documentation

#### G.92.2.1 NotImplemented() [1/2]

```
BiometricEvaluation::Error::NotImplemented::NotImplemented ()
```

Construct a **NotImplemented** (p. 456) object with the default information string.

#### G.92.2.2 NotImplemented() [2/2]

```
BiometricEvaluation::Error::NotImplemented::NotImplemented (
 const std::string & info)
```

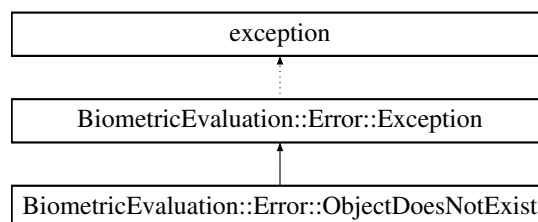
Construct a **NotImplemented** (p. 456) object with an information string appended to the default information string.

## G.93 BiometricEvaluation::Error::ObjectDoesNotExist Class Reference

The named object does not exist.

```
#include <be_error_exception.h>
```

Inheritance diagram for BiometricEvaluation::Error::ObjectDoesNotExist:



## Public Member Functions

- **ObjectDoesNotExist** ()
- **ObjectDoesNotExist** (const std::string &info)

### G.93.1 Detailed Description

The named object does not exist.

### G.93.2 Constructor & Destructor Documentation

#### G.93.2.1 ObjectDoesNotExist() [1/2]

BiometricEvaluation::Error::ObjectDoesNotExist::ObjectDoesNotExist ( )  
Construct a **ObjectDoesNotExist** (p. 457) object with the default information string.

#### G.93.2.2 ObjectDoesNotExist() [2/2]

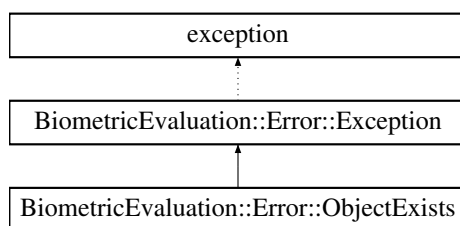
BiometricEvaluation::Error::ObjectDoesNotExist::ObjectDoesNotExist (   
const std::string & info )  
Construct a **ObjectDoesNotExist** (p. 457) object with an information string appended to the default information string.

## G.94 BiometricEvaluation::Error::ObjectExists Class Reference

The named object exists and will not be replaced.

```
#include <be_error_exception.h>
```

Inheritance diagram for BiometricEvaluation::Error::ObjectExists:



## Public Member Functions

- **ObjectExists** ()
- **ObjectExists** (const std::string &info)

### G.94.1 Detailed Description

The named object exists and will not be replaced.

### G.94.2 Constructor & Destructor Documentation

**G.94.2.1 ObjectExists() [1/2]**

```
BiometricEvaluation::Error::ObjectExists::ObjectExists ()
```

Construct a **ObjectExists** (p. 458) object with the default information string.

**G.94.2.2 ObjectExists() [2/2]**

```
BiometricEvaluation::Error::ObjectExists::ObjectExists (
 const std::string & info)
```

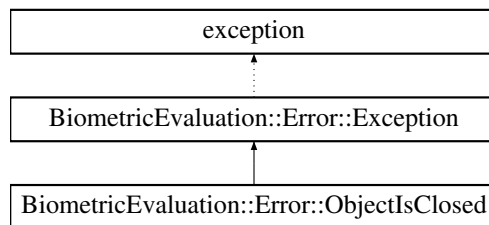
Construct a **ObjectExists** (p. 458) object with an information string appended to the default information string.

**G.95 BiometricEvaluation::Error::ObjectIsClosed Class Reference**

The object is closed.

```
#include <be_error_exception.h>
```

Inheritance diagram for BiometricEvaluation::Error::ObjectIsClosed:

**Public Member Functions**

- **ObjectIsClosed** ( )
- **ObjectIsClosed** (const std::string &info)

**G.95.1 Detailed Description**

The object is closed.

**G.95.2 Constructor & Destructor Documentation****G.95.2.1 ObjectIsClosed() [1/2]**

```
BiometricEvaluation::Error::ObjectIsClosed::ObjectIsClosed ()
```

Construct a **ObjectIsClosed** (p. 459) object with the default information string.

**G.95.2.2 ObjectIsClosed() [2/2]**

```
BiometricEvaluation::Error::ObjectIsClosed::ObjectIsClosed (
 const std::string & info)
```

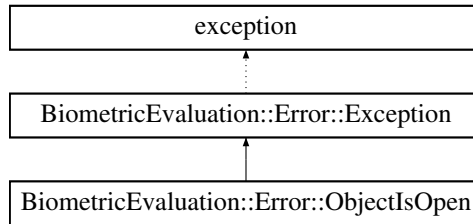
Construct a **ObjectIsClosed** (p. 459) object with an information string appended to the default information string.

## G.96 BiometricEvaluation::Error::ObjectIsOpen Class Reference

The object is already opened.

```
#include <be_error_exception.h>
```

Inheritance diagram for BiometricEvaluation::Error::ObjectIsOpen:



### Public Member Functions

- **ObjectIsOpen** ()
- **ObjectIsOpen** (const std::string &info)

### G.96.1 Detailed Description

The object is already opened.

### G.96.2 Constructor & Destructor Documentation

#### G.96.2.1 ObjectIsOpen() [1/2]

```
BiometricEvaluation::Error::ObjectIsOpen::ObjectIsOpen ()
```

Construct a **ObjectIsOpen** (p. 460) object with the default information string.

#### G.96.2.2 ObjectIsOpen() [2/2]

```
BiometricEvaluation::Error::ObjectIsOpen::ObjectIsOpen (
 const std::string & info)
```

Construct a **ObjectIsOpen** (p. 460) object with an information string appended to the default information string.

## G.97 BiometricEvaluation::Memory::OrderedMap< Key, T > Class Template Reference

```
#include <be_memory_orderedmap.h>
```

### Public Types

- using **container** = typename std::unordered\_map< Key, T >
- using **iterator** = **OrderedMapIterator**< Key, T >
- using **const\_iterator** = **OrderedMapConstIterator**< Key, T >
- using **size\_type** = typename container::size\_type

- using **value\_type** = typename container::value\_type
- using **key\_type** = Key
- using **mapped\_type** = T
- using **key\_equal** = typename container::key\_equal

## Public Member Functions

- **OrderedMap** ()
- bool **push\_back** (const value\_type &value)  
*Insert an element at the end of the collection.*
- void **erase** ( iterator pos)  
*Remove an element from the collection.*
- void **erase** (const Key &key)  
*Remove an element from the collection.*
- **iterator** **begin** ()
- **const\_iterator** **begin** () const
- **const\_iterator** **cbegin** () const
- **iterator** **end** ()
- **const\_iterator** **end** () const
- **const\_iterator** **cend** () const
- size\_type **size** () const
- bool **keyExists** (const Key &key) const  
*Determine if a value exists in the container.*
- const **OrderedMapIterator**< Key, T > **find** (const Key &key) const  
*Obtain an iterator to a particular key.*
- std::shared\_ptr< value\_type > **find\_quick** (const Key &key) const
- T & **operator** [] (const Key &key)  
*Subscripting operator.*
- key\_equal **key\_eq** () const
- ~**OrderedMap** ()

## Friends

- class **OrderedMapIterator**< Key, T >
- class **OrderedMapConstIterator**< Key, T >

### G.97.1 Detailed Description

template<class Key, class T>

class BiometricEvaluation::Memory::OrderedMap< Key, T >

A map where insertion order is preserved and elements are unique.

### G.97.2 Constructor & Destructor Documentation

### G.97.2.1 OrderedMap()

```
template<class Key , class T >
BiometricEvaluation::Memory::OrderedMap< Key, T >:: OrderedMap ()
 Constructor.
```

### G.97.2.2 ~OrderedMap()

```
template<class Key , class T >
BiometricEvaluation::Memory::OrderedMap< Key, T >::~ ~ OrderedMap ()
 Destructor
```

## G.97.3 Member Function Documentation

### G.97.3.1 begin() [1/2]

```
template<class Key , class T >
BiometricEvaluation::Memory::OrderedMap< Key, T >:: iterator BiometricEvaluation::Memory::OrderedMap< Key, T >::begin ()
```

Returns

Iterator at the first element of the collection.

### G.97.3.2 begin() [2/2]

```
template<class Key , class T >
BiometricEvaluation::Memory::OrderedMap< Key, T >:: const_iterator BiometricEvaluation::Memory::OrderedMap< Key, T >::begin () const
```

Returns

Iterator at the first element of the collection.

### G.97.3.3 cbegin()

```
template<class Key , class T >
BiometricEvaluation::Memory::OrderedMap< Key, T >:: const_iterator BiometricEvaluation::Memory::OrderedMap< Key, T >::cbegin () const
```

Returns

Iterator at the first element of the collection.



**G.97.3.4** `cend()`

```
template<class Key , class T >
BiometricEvaluation::Memory::OrderedMap< Key, T >:: const_iterator BiometricEvaluation::Memory::OrderedMap< Key, T >::cend () const
```

Returns

Iterator beyond the last element of the collection.

**G.97.3.5** `end()` [1/2]

```
template<class Key , class T >
BiometricEvaluation::Memory::OrderedMap< Key, T >:: iterator BiometricEvaluation::Memory::OrderedMap< Key, T >::end ()
```

Returns

Iterator beyond the last element of the collection.

**G.97.3.6** `end()` [2/2]

```
template<class Key , class T >
BiometricEvaluation::Memory::OrderedMap< Key, T >:: const_iterator BiometricEvaluation::Memory::OrderedMap< Key, T >::end () const
```

Returns

Iterator beyond the last element of the collection.

**G.97.3.7** `erase()` [1/2]

```
template<class Key , class T >
void BiometricEvaluation::Memory::OrderedMap< Key, T >::erase (
 iterator pos)
```

Remove an element from the collection.

Parameters

|            |                                                              |
|------------|--------------------------------------------------------------|
| <i>pos</i> | Iterator to element at the position which should be removed. |
|------------|--------------------------------------------------------------|

Note

Complexity: Average case: O(1), worst case O(size()).

**G.97.3.8** `erase()` [2/2]

```
template<class Key , class T >
```

```
void BiometricEvaluation::Memory::OrderedMap< Key, T >::erase (
 const Key & key)
```

Remove an element from the collection.

Parameters

|            |                               |
|------------|-------------------------------|
| <i>key</i> | Key of the element to remove. |
|------------|-------------------------------|

### G.97.3.9 find()

```
template<class Key , class T >
const BiometricEvaluation::Memory::OrderedMapIterator< Key, T > BiometricEvaluation::Memory::
::OrderedMap< Key, T >::find (
 const Key & key) const
```

Obtain an iterator to a particular key.

Note

Complexity is O(n).

### G.97.3.10 key\_eq()

```
template<class Key , class T >
BiometricEvaluation::Memory::OrderedMap< Key, T >::key_equal BiometricEvaluation::Memory::
OrderedMap< Key, T >::key_eq () const
```

Returns

Function that compares keys for equality.

### G.97.3.11 keyExists()

```
template<class Key , class T >
bool BiometricEvaluation::Memory::OrderedMap< Key, T >::keyExists (
 const Key & key) const
```

Determine if a value exists in the container.

Parameters

|            |                                  |
|------------|----------------------------------|
| <i>key</i> | Key to search the container for. |
|------------|----------------------------------|

Returns

Whether or not key exists in this container.

Note

Complexity is O(1).

### G.97.3.12 operator[]()

```
template<class Key , class T >
T & BiometricEvaluation::Memory::OrderedMap< Key, T >::operator[] (
 const Key & key)
 Subscripting operator.
```

Parameters

|            |                                 |
|------------|---------------------------------|
| <i>key</i> | Key used to index into the map. |
|------------|---------------------------------|

Returns

Value for key, which may be a new value.

### G.97.3.13 push\_back()

```
template<class Key , class T >
bool BiometricEvaluation::Memory::OrderedMap< Key, T >::push_back (
 const value_type & value)
 Insert an element at the end of the collection.
```

Parameters

|              |                  |
|--------------|------------------|
| <i>value</i> | Value to insert. |
|--------------|------------------|

Returns

Whether or not the object was inserted.

Note

Complexity: Average case: O(1), worst case O(size()).

### G.97.3.14 size()

```
template<class Key , class T >
BiometricEvaluation::Memory::OrderedMap< Key, T >::size_type BiometricEvaluation::Memory::OrderedMap< Key, T >::size () const
```

Returns

Number of elements in the collection.

## G.98 BiometricEvaluation::Memory::OrderedMapConstIterator< Key, T > Class Template Reference

```
#include <be_memory_orderedmap.h>
```

## Public Types

- using **iterator\_category** = std::bidirectional\_iterator\_tag
- using **value\_type** = std::pair< Key, T >
- using **difference\_type** = std::ptrdiff\_t
- using **pointer** = const **value\_type** \*
- using **reference** = const **value\_type** &

## Public Member Functions

- **OrderedMapConstIterator** ()
- **OrderedMapConstIterator** (const **OrderedMapIterator**< Key, T > &iterator)
- **~OrderedMapConstIterator** ()
- **reference operator\*** () const
- **pointer operator->** () const
- **OrderedMapConstIterator & operator++** ()
- **OrderedMapConstIterator operator++** (int dummy)
- **OrderedMapConstIterator & operator--** ()
- **OrderedMapConstIterator operator--** (int dummy)
- bool **operator==** (const **OrderedMapConstIterator** &rhs) const  
*Test for iterator equality.*
- bool **operator!=** (const **OrderedMapConstIterator** &rhs) const  
*Test for iterator equality.*

## Friends

- class **OrderedMap**< Key, T >

## G.98.1 Detailed Description

```
template<class Key, class T>
class BiometricEvaluation::Memory::OrderedMapConstIterator< Key, T >
```

Const Iterator for OrderedMaps.

## G.98.2 Member Typedef Documentation

### G.98.2.1 difference\_type

```
template<class Key, class T>
using BiometricEvaluation::Memory::OrderedMapConstIterator< Key, T >:: difference_type = std::
::ptrdiff_t
```

Type used to measure distance between iterators

### G.98.2.2 iterator\_category

```
template<class Key, class T>
using BiometricEvaluation::Memory::OrderedMapConstIterator< Key, T >:: iterator_category =
std::bidirectional_iterator_tag
```

Type of iterator

### G.98.2.3 pointer

```
template<class Key, class T>
using BiometricEvaluation::Memory::OrderedMapConstIterator< Key, T >:: pointer = const value←
.type*
```

Pointer to the type iterated over

### G.98.2.4 reference

```
template<class Key, class T>
using BiometricEvaluation::Memory::OrderedMapConstIterator< Key, T >:: reference = const value←
.type&
```

Reference to the type iterated over

### G.98.2.5 value\_type

```
template<class Key, class T>
using BiometricEvaluation::Memory::OrderedMapConstIterator< Key, T >:: value_type = std::pair<Key,
T>
```

Type when dereferencing iterators

## G.98.3 Constructor & Destructor Documentation

### G.98.3.1 OrderedMapConstIterator() [1/2]

```
template<class Key , class T >
BiometricEvaluation::Memory::OrderedMapConstIterator< Key, T >:: OrderedMapConstIterator ()
 Constructor
```

### G.98.3.2 OrderedMapConstIterator() [2/2]

```
template<class Key , class T >
BiometricEvaluation::Memory::OrderedMapConstIterator< Key, T >:: OrderedMapConstIterator (
 const OrderedMapIterator< Key, T > & iterator)
```

Iterator to ConstIterator converter

### G.98.3.3 ~OrderedMapConstIterator()

```
template<class Key , class T >
BiometricEvaluation::Memory::OrderedMapConstIterator< Key, T >::~~ OrderedMapConstIterator (
)
```

Destructor

## G.98.4 Member Function Documentation

**G.98.4.1 operator”!=()**

```
template<class Key , class T >
bool BiometricEvaluation::Memory::OrderedMapConstIterator< Key, T >::operator!= (
 const OrderedMapConstIterator< Key, T > & rhs) const
 Test for iterator equality.
```

Parameters

|            |                                                  |
|------------|--------------------------------------------------|
| <i>rhs</i> | Object on the right-hand side of the expression. |
|------------|--------------------------------------------------|

Returns

Whether or not this iterator is not equivalent to rhs.

**G.98.4.2 operator\*()**

```
template<class Key , class T >
BiometricEvaluation::Memory::OrderedMapConstIterator< Key, T >:: reference BiometricEvaluation↵
::Memory::OrderedMapConstIterator< Key, T >::operator* () const
```

Returns

Reference to the current iterated pair.

**G.98.4.3 operator++() [1/2]**

```
template<class Key , class T >
BiometricEvaluation::Memory::OrderedMapConstIterator< Key, T > & BiometricEvaluation::Memory↵
::OrderedMapConstIterator< Key, T >::operator++ ()
 Move to the next pair
```

**G.98.4.4 operator++() [2/2]**

```
template<class Key , class T >
BiometricEvaluation::Memory::OrderedMapConstIterator< Key, T > BiometricEvaluation::Memory↵
::OrderedMapConstIterator< Key, T >::operator++ (
 int dummy)
 Move to the next pair
```

**G.98.4.5 operator--() [1/2]**

```
template<class Key , class T >
BiometricEvaluation::Memory::OrderedMapConstIterator< Key, T > & BiometricEvaluation::Memory↵
::OrderedMapConstIterator< Key, T >::operator-- ()
 Move to the previous pair.
```

**G.98.4.6 operator--() [2/2]**

```
template<class Key , class T >
BiometricEvaluation::Memory::OrderedMapConstIterator< Key, T > BiometricEvaluation::Memory::OrderedMapConstIterator< Key, T >::operator-- (
 int dummy)
```

Move to the previous pair.

**G.98.4.7 operator->()**

```
template<class Key , class T >
BiometricEvaluation::Memory::OrderedMapConstIterator< Key, T >:: pointer BiometricEvaluation::Memory::OrderedMapConstIterator< Key, T >::operator-> () const
```

Returns

Pointer to the current iterated pair.

**G.98.4.8 operator==( )**

```
template<class Key , class T >
bool BiometricEvaluation::Memory::OrderedMapConstIterator< Key, T >::operator== (
 const OrderedMapConstIterator< Key, T > & rhs) const
```

Test for iterator equality.

Parameters

|            |                                                  |
|------------|--------------------------------------------------|
| <i>rhs</i> | Object on the right-hand side of the expression. |
|------------|--------------------------------------------------|

Returns

Whether or not this iterator is equivalent to rhs.

## G.99 BiometricEvaluation::Memory::OrderedMapIterator< Key, T > Class Template Reference

```
#include <be_memory_orderedmap.h>
```

**Public Types**

- using **iterator\_category** = std::bidirectional\_iterator\_tag
- using **value\_type** = std::pair< Key, T >
- using **difference\_type** = std::ptrdiff\_t
- using **pointer** = value\_type \*
- using **reference** = value\_type &

**Public Member Functions**

- **OrderedMapIterator** ( )
- **~OrderedMapIterator** ( )

- **reference** **operator\*** () const
- **pointer** **operator->** () const
- **OrderedMapIterator** & **operator++** ()
- **OrderedMapIterator** **operator++** (int dummy)
- **OrderedMapIterator** & **operator--** ()
- **OrderedMapIterator** **operator--** (int dummy)
- **bool** **operator==** (const **OrderedMapIterator** &rhs) const  
*Test for iterator equality.*
- **bool** **operator!=** (const **OrderedMapIterator** &rhs) const  
*Test for iterator equality.*

## Friends

- class **OrderedMap**< **Key**, **T** >
- class **OrderedMapConstIterator**< **Key**, **T** >

### G.99.1 Detailed Description

template<class Key, class T>

class **BiometricEvaluation::Memory::OrderedMapIterator**< **Key**, **T** >

Iterator for OrderedMaps.

### G.99.2 Member Typedef Documentation

#### G.99.2.1 difference\_type

template<class Key, class T>

using **BiometricEvaluation::Memory::OrderedMapIterator**< **Key**, **T** >:: **difference\_type** = std::ptrdiff\_t

Type used to measure distance between iterators

#### G.99.2.2 iterator\_category

template<class Key, class T>

using **BiometricEvaluation::Memory::OrderedMapIterator**< **Key**, **T** >:: **iterator\_category** = std::bidirectional\_iterator\_tag

Type of iterator

#### G.99.2.3 pointer

template<class Key, class T>

using **BiometricEvaluation::Memory::OrderedMapIterator**< **Key**, **T** >:: **pointer** = **value\_type\***

Pointer to the type iterated over

#### G.99.2.4 reference

template<class Key, class T>

using **BiometricEvaluation::Memory::OrderedMapIterator**< **Key**, **T** >:: **reference** = **value\_type&**

Reference to the type iterated over



### G.99.2.5 value\_type

```
template<class Key, class T>
using BiometricEvaluation::Memory::OrderedMapIterator< Key, T >:: value_type = std::pair<Key,
T>
 Type when dereferencing iterators
```

## G.99.3 Constructor & Destructor Documentation

### G.99.3.1 OrderedMapIterator()

```
template<class Key , class T >
BiometricEvaluation::Memory::OrderedMapIterator< Key, T >:: OrderedMapIterator ()
 Constructor
```

### G.99.3.2 ~OrderedMapIterator()

```
template<class Key , class T >
BiometricEvaluation::Memory::OrderedMapIterator< Key, T >::~~ OrderedMapIterator ()
 Destructor
```

## G.99.4 Member Function Documentation

### G.99.4.1 operator"!==( )

```
template<class Key , class T >
bool BiometricEvaluation::Memory::OrderedMapIterator< Key, T >::operator!= (
 const OrderedMapIterator< Key, T > & rhs) const
 Test for iterator equality.
```

Parameters

|            |                                                  |
|------------|--------------------------------------------------|
| <i>rhs</i> | Object on the right-hand side of the expression. |
|------------|--------------------------------------------------|

Returns

Whether or not this iterator is not equivalent to rhs.

### G.99.4.2 operator\*()

```
template<class Key , class T >
BiometricEvaluation::Memory::OrderedMapIterator< Key, T >:: reference BiometricEvaluation↵
::Memory::OrderedMapIterator< Key, T >::operator* () const
```

Returns

Reference to the current iterated pair.

**G.99.4.3 operator++()** [1/2]

```
template<class Key , class T >
BiometricEvaluation::Memory::OrderedMapIterator< Key, T > & BiometricEvaluation::Memory::OrderedMapIterator< Key, T >::operator++ ()
 Move to the next pair
```

**G.99.4.4 operator++()** [2/2]

```
template<class Key , class T >
BiometricEvaluation::Memory::OrderedMapIterator< Key, T > BiometricEvaluation::Memory::OrderedMapIterator< Key, T >::operator++ (
 int dummy)
 Move to the next pair
```

**G.99.4.5 operator--()** [1/2]

```
template<class Key , class T >
BiometricEvaluation::Memory::OrderedMapIterator< Key, T > & BiometricEvaluation::Memory::OrderedMapIterator< Key, T >::operator-- ()
 Move to the previous pair.
```

**G.99.4.6 operator--()** [2/2]

```
template<class Key , class T >
BiometricEvaluation::Memory::OrderedMapIterator< Key, T > BiometricEvaluation::Memory::OrderedMapIterator< Key, T >::operator-- (
 int dummy)
 Move to the previous pair.
```

**G.99.4.7 operator->()**

```
template<class Key , class T >
BiometricEvaluation::Memory::OrderedMapIterator< Key, T >:: pointer BiometricEvaluation::Memory::OrderedMapIterator< Key, T >::operator-> () const
```

Returns

Pointer to the current iterated pair.

**G.99.4.8 operator==( )**

```
template<class Key , class T >
bool BiometricEvaluation::Memory::OrderedMapIterator< Key, T >::operator==(
 const OrderedMapIterator< Key, T > & rhs) const
 Test for iterator equality.
```

Parameters

|            |                                                  |
|------------|--------------------------------------------------|
| <i>rhs</i> | Object on the right-hand side of the expression. |
|------------|--------------------------------------------------|

Returns

Whether or not this iterator is equivalent to rhs.

## G.100 BiometricEvaluation::Feature::AN2K11EFS::Orientation Struct Reference

Representation of orientation (deviation from upright) and its uncertainty.

```
#include <be_feature_an2k11efs.h>
```

### Public Attributes

- bool **is\_default**
- int **eod**
- bool **has\_euc**
- int **euc**

### Static Public Attributes

- static const int **EODDefault** = 0
- static const int **EUCDefault** = 15

### G.100.1 Detailed Description

Representation of orientation (deviation from upright) and its uncertainty.

### G.100.2 Member Data Documentation

#### G.100.2.1 eod

```
int BiometricEvaluation::Feature::AN2K11EFS::Orientation::eod
 Direction
```

#### G.100.2.2 EODDefault

```
const int BiometricEvaluation::Feature::AN2K11EFS::Orientation::EODDefault = 0 [static]
 ANSI/NIST default direction
```

#### G.100.2.3 euc

```
int BiometricEvaluation::Feature::AN2K11EFS::Orientation::euc
 Uncertainty
```

#### G.100.2.4 EUCDefault

```
const int BiometricEvaluation::Feature::AN2K11EFS::Orientation::EUCDefault = 15 [static]
 ANSI/NIST default uncertainty
```

### G.100.2.5 is\_default

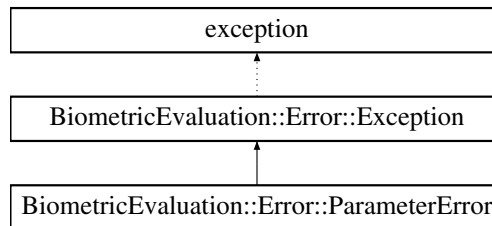
```
bool BiometricEvaluation::Feature::AN2K11EFS::Orientation::is_default
 Whether the values are the defaults
```

## G.101 BiometricEvaluation::Error::ParameterError Class Reference

An invalid parameter was passed to a constructor or method.

```
#include <be_error_exception.h>
```

Inheritance diagram for BiometricEvaluation::Error::ParameterError:



### Public Member Functions

- **ParameterError** ()
- **ParameterError** (const std::string &info)

### G.101.1 Detailed Description

An invalid parameter was passed to a constructor or method.

### G.101.2 Constructor & Destructor Documentation

#### G.101.2.1 ParameterError() [1/2]

```
BiometricEvaluation::Error::ParameterError::ParameterError ()
```

Construct a **ParameterError** (p. 474) object with the default information string.

#### G.101.2.2 ParameterError() [2/2]

```
BiometricEvaluation::Error::ParameterError::ParameterError (
 const std::string & info)
```

Construct a **ParameterError** (p. 474) object with an information string appended to the default information string.

## G.102 BiometricEvaluation::Feature::AN2K7Minutiae::PatternClassification Class Reference

Pattern classification codes.

```
#include <be_feature_an2k7minutiae.h>
```

## Classes

- struct **Entry**

## Public Types

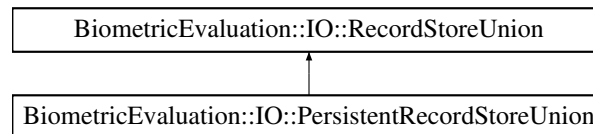
- using **Entry** = struct **Entry**

### G.102.1 Detailed Description

Pattern classification codes.

## G.103 BiometricEvaluation::IO::PersistentRecordStoreUnion Class Reference

Inheritance diagram for BiometricEvaluation::IO::PersistentRecordStoreUnion:



## Public Member Functions

- **PersistentRecordStoreUnion** (const std::string &path)  
*Open an existing **PersistentRecordStoreUnion** (p. 475).*
- **PersistentRecordStoreUnion** (const std::string &path, const std::map< const std::string, const std::string > &recordStores)  
*Create a new **PersistentRecordStoreUnion** (p. 475).*
- **PersistentRecordStoreUnion** (const std::string &path, std::initializer\_list< std::pair< const std::string, const std::string >> &recordStores)  
*Create a new **PersistentRecordStoreUnion** (p. 475).*
- ~**PersistentRecordStoreUnion** ()=default

## Additional Inherited Members

### G.103.1 Constructor & Destructor Documentation

#### G.103.1.1 PersistentRecordStoreUnion() [1/3]

```
BiometricEvaluation::IO::PersistentRecordStoreUnion::PersistentRecordStoreUnion (
 const std::string & path)
```

Open an existing **PersistentRecordStoreUnion** (p. 475).

Parameters

|             |                                                               |
|-------------|---------------------------------------------------------------|
| <i>path</i> | Path at which <b>RecordStoreUnion</b> (p. 526) was persisted. |
|-------------|---------------------------------------------------------------|

**G.103.1.2 PersistentRecordStoreUnion() [2/3]**

```
BiometricEvaluation::IO::PersistentRecordStoreUnion::PersistentRecordStoreUnion (
 const std::string & path,
 const std::map< const std::string, const std::string > & recordStores)
```

Create a new **PersistentRecordStoreUnion** (p. 475).

Parameters

|                     |                                                                   |
|---------------------|-------------------------------------------------------------------|
| <i>path</i>         | Path at which <b>RecordStoreUnion</b> (p. 526) will be persisted. |
| <i>recordStores</i> | Initial RecordStores members of the union.                        |

**G.103.1.3 PersistentRecordStoreUnion() [3/3]**

```
BiometricEvaluation::IO::PersistentRecordStoreUnion::PersistentRecordStoreUnion (
 const std::string & path,
 std::initializer_list< std::pair< const std::string, const std::string >> & recordStores)
```

Create a new **PersistentRecordStoreUnion** (p. 475).

Parameters

|                     |                                                                   |
|---------------------|-------------------------------------------------------------------|
| <i>path</i>         | Path at which <b>RecordStoreUnion</b> (p. 526) will be persisted. |
| <i>mode</i>         | Mode in which to open RecordStores in the union.                  |
| <i>recordStores</i> | Initial RecordStores members of the union.                        |

**G.103.1.4 ~PersistentRecordStoreUnion()**

```
BiometricEvaluation::IO::PersistentRecordStoreUnion::~~PersistentRecordStoreUnion () [default]
```

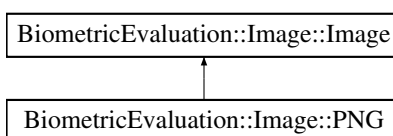
Destructor

**G.104 BiometricEvaluation::Image::PNG Class Reference**

A PNG-encoded image.

```
#include <be_image_png.h>
```

Inheritance diagram for BiometricEvaluation::Image::PNG:



## Public Member Functions

- **PNG** (const uint8\_t \*data, const uint64\_t size)
- **PNG** (const **Memory::uint8Array** &data)
- **Memory::uint8Array** **getRawData** () const  
*Accessor for the raw image data. The data returned should not be compressed or encoded.*
- **Memory::uint8Array** **getRawGrayscaleData** (uint8\_t depth) const  
*Accessor for decompressed data in grayscale.*

## Static Public Member Functions

- static bool **isPNG** (const uint8\_t \*data, uint64\_t size)

## Additional Inherited Members

### G.104.1 Detailed Description

A PNG-encoded image.

### G.104.2 Member Function Documentation

#### G.104.2.1 getRawData()

**Memory::uint8Array** **BiometricEvaluation::Image::PNG::getRawData** ( ) const [virtual]  
 Accessor for the raw image data. The data returned should not be compressed or encoded.

Returns

AutoArray holding raw image data.

Exceptions

|                                  |                                                 |
|----------------------------------|-------------------------------------------------|
| <b>Error::DataError</b> (p. 295) | <b>Error</b> (p. 108) decompressing image data. |
|----------------------------------|-------------------------------------------------|

Implements **BiometricEvaluation::Image::Image** (p. 361).

#### G.104.2.2 getRawGrayscaleData()

**Memory::uint8Array** **BiometricEvaluation::Image::PNG::getRawGrayscaleData** (  
     uint8\_t depth ) const [virtual]  
 Accessor for decompressed data in grayscale.

Parameters

|              |                                                                                         |
|--------------|-----------------------------------------------------------------------------------------|
| <i>depth</i> | The desired bit depth of the resulting raw image. This value may either be 16, 8, or 1. |
|--------------|-----------------------------------------------------------------------------------------|

## Returns

AutoArray holding raw grayscale image data.

## Exceptions

|                                       |                                                     |
|---------------------------------------|-----------------------------------------------------|
| <b>Error::DataError</b> (p. 295)      | <b>Error</b> (p. 108) decompressing image data.     |
| <b>Error::NotImplemented</b> (p. 456) | Unsupported conversion based on source color depth. |
| <b>Error::ParameterError</b> (p. 474) | Invalid value for depth.                            |

## Note

This method does not save a cached copy of the decompressed image because the bit depth of the image can be changed between calls.

When depth is 1, this method returns an image that uses 8 bits to represent a single pixel. The depth parameter is used to adjust the number of gray levels. When depth is 1, there are only 2 gray levels (black and white), despite using 8 bits to represent each pixel.

Alpha channels are completely ignored when converting to grayscale.

Implements **BiometricEvaluation::Image::Image** (p. 362).

**G.104.2.3 isPNG()**

```
static bool BiometricEvaluation::Image::PNG::isPNG (
 const uint8_t * data,
 uint64_t size) [static]
```

Whether or not data is a **PNG** (p. 476) image.

## Parameters

|    |             |                      |
|----|-------------|----------------------|
| in | <i>data</i> | The buffer to check. |
| in | <i>size</i> | The size of data.    |

## Returns

true if data appears to be a **PNG** (p. 476) image, false otherwise

**G.105 BiometricEvaluation::Feature::Sort::Polar Class Reference**

**Sort** (p. 113) by increasing distance from center and angle (theta).

```
#include <be_feature_sort.h>
```

**Public Member Functions**

- **Polar** (const **BiometricEvaluation::Image::Coordinate** &center)  
*Polar* (p. 478) constructor.
- bool **operator()** (const **BiometricEvaluation::Feature::MinutiaPoint** &lhs, const **BiometricEvaluation::Feature::MinutiaPoint** &rhs) const



Static Public Member Functions

- static **BiometricEvaluation::Image::Coordinate** **centerOfMinutiaeMass** (const BiometricEvaluation←  
::Feature::MinutiaPointSet &mps)  
*Obtain the center of minutiae mass.*
- static **BiometricEvaluation::Image::Coordinate** **centerOfImage** (const **BiometricEvaluation::←  
Image::Size** &size)  
*Obtain the center point of an image.*

G.105.1 Detailed Description

Sort (p. 113) by increasing distance from center and angle (theta).

G.105.2 Constructor & Destructor Documentation

G.105.2.1 Polar()

BiometricEvaluation::Feature::Sort::Polar::Polar (  
const **BiometricEvaluation::Image::Coordinate** & center )  
**Polar** (p. 478) constructor.

Parameters

|               |                                        |
|---------------|----------------------------------------|
| <i>center</i> | Coordinate to use for center of image. |
|---------------|----------------------------------------|

centerOfMinutiaeMass centerOfImage

G.105.3 Member Function Documentation

G.105.3.1 centerOfImage()

static **BiometricEvaluation::Image::Coordinate** BiometricEvaluation::Feature::Sort::Polar::center←  
OfImage (  
const **BiometricEvaluation::Image::Size** & size ) [static]  
Obtain the center point of an image.

Parameters

|             |                   |
|-------------|-------------------|
| <i>size</i> | Size of an image. |
|-------------|-------------------|

Note

If dimensions are odd, integer division is applied.

G.105.3.2 centerOfMinutiaeMass()

static **BiometricEvaluation::Image::Coordinate** BiometricEvaluation::Feature::Sort::Polar::center←

```
OfMinutiaeMass (
 const BiometricEvaluation::Feature::MinutiaPointSet & mps) [static]
 Obtain the center of minutiae mass.
```

Parameters

|            |                               |
|------------|-------------------------------|
| <i>mps</i> | Collection of minutia points. |
|------------|-------------------------------|

Returns

Center of minutiae mass for mps.

Exceptions

|                                      |             |
|--------------------------------------|-------------|
| <i>Error::StrategyError</i> (p. 567) | No minutia. |
|--------------------------------------|-------------|

### G.105.3.3 operator()

```
bool BiometricEvaluation::Feature::Sort::Polar::operator() (
 const BiometricEvaluation::Feature::MinutiaPoint & lhs,
 const BiometricEvaluation::Feature::MinutiaPoint & rhs) const
 MinutiaPoint (p. 444) polar ascending comparator.
```

## G.106 BiometricEvaluation::Face::PoseAngle Struct Reference

Representation of pose angle and uncertainty.

```
#include <be_face.h>
```

### Public Attributes

- uint8\_t **yaw**
- uint8\_t **pitch**
- uint8\_t **roll**
- uint8\_t **yawUncertainty**
- uint8\_t **pitchUncertainty**
- uint8\_t **rollUncertainty**

### G.106.1 Detailed Description

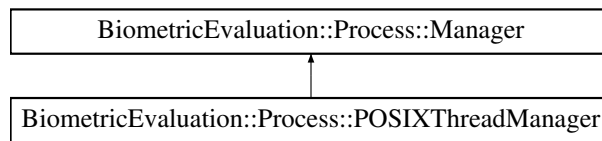
Representation of pose angle and uncertainty.

## G.107 BiometricEvaluation::Process::POSIXThreadManager Class Reference

**Manager** (p. 430) implementation that starts Workers in POSIX threads.

```
#include <be_process_posixthreadmanager.h>
```

Inheritance diagram for BiometricEvaluation::Process::POSIXThreadManager:



## Public Member Functions

- **POSIXThreadManager** ()
- `std::shared_ptr< WorkerController > addWorker (std::shared_ptr< Worker > worker)`  
*Adds a **Worker** (p. 595) to be managed by this **Manager** (p. 430).*
- `void startWorkers (bool wait=true, bool communicate=false)`  
*Begin **Worker** (p. 595)'s work.*
- `void startWorker (std::shared_ptr< WorkerController > worker, bool wait=true, bool communicate=false)`  
*Start a **Worker** (p. 595).*
- `void stopWorker (std::shared_ptr< WorkerController > workerController)`  
*Ask **Worker** (p. 595) to exit.*
- `void waitForWorkerExit ()`  
*Block until all Workers have exited.*
- **~POSIXThreadManager** ()  
*~POSIXThreadManager destructor.*

## Additional Inherited Members

### G.107.1 Detailed Description

**Manager** (p. 430) implementation that starts Workers in POSIX threads.

### G.107.2 Constructor & Destructor Documentation

#### G.107.2.1 POSIXThreadManager()

`BiometricEvaluation::Process::POSIXThreadManager::POSIXThreadManager ( )`  
**POSIXThreadManager** (p. 480) constructor.

### G.107.3 Member Function Documentation

#### G.107.3.1 addWorker()

```
std::shared_ptr< WorkerController> BiometricEvaluation::Process::POSIXThreadManager::addWorker
(
 std::shared_ptr< Worker > worker) [virtual]
 Adds a Worker (p. 595) to be managed by this Manager (p. 430).
```

## Parameters

|               |                                           |
|---------------|-------------------------------------------|
| <i>worker</i> | A <b>Worker</b> (p. 595) instance to run. |
|---------------|-------------------------------------------|

## Returns

shared\_ptr to worker.

Implements **BiometricEvaluation::Process::Manager** (p. 431).

**G.107.3.2 startWorker()**

```
void BiometricEvaluation::Process::POSIXThreadManager::startWorker (
 std::shared_ptr< WorkerController > worker,
 bool wait = true,
 bool communicate = false) [virtual]
```

Start a **Worker** (p. 595).

## Parameters

|                    |                                                                                                               |
|--------------------|---------------------------------------------------------------------------------------------------------------|
| <i>worker</i>      | Pointer to a <b>WorkerController</b> (p. 601) that is being managed by this <b>Manager</b> (p. 430) instance. |
| <i>wait</i>        | Whether or not to wait for this <b>Worker</b> (p. 595) to exit before returning control to the caller.        |
| <i>communicate</i> | Whether or not to enable communication among the Workers and Managers.                                        |

## Exceptions

|                                      |                                                                 |
|--------------------------------------|-----------------------------------------------------------------|
| <b>Error::ObjectExists</b> (p. 458)  | worker is already working.                                      |
| <b>Error::StrategyError</b> (p. 567) | worker is not managed by this <b>Manager</b> (p. 430) instance. |

Implements **BiometricEvaluation::Process::Manager** (p. 433).

**G.107.3.3 startWorkers()**

```
void BiometricEvaluation::Process::POSIXThreadManager::startWorkers (
 bool wait = true,
 bool communicate = false) [virtual]
```

Begin **Worker** (p. 595)'s work.

## Parameters

|    |                    |                                                                        |
|----|--------------------|------------------------------------------------------------------------|
| in | <i>wait</i>        | Whether or not to wait for all Workers to return before returning.     |
| in | <i>communicate</i> | Whether or not to enable communication among the Workers and Managers. |

## Exceptions

|                                      |                                                         |
|--------------------------------------|---------------------------------------------------------|
| <b>Error::ObjectExists</b> (p. 458)  | At least one <b>Worker</b> (p. 595) is already working. |
| <b>Error::StrategyError</b> (p. 567) | Problem starting the Workers.                           |

Implements **BiometricEvaluation::Process::Manager** (p. 434).

## G.107.3.4 stopWorker()

```
void BiometricEvaluation::Process::POSIXThreadManager::stopWorker (
 std::shared_ptr< WorkerController > workerController) [virtual]
 Ask Worker (p. 595) to exit.
```

## Parameters

|                         |                                                                         |
|-------------------------|-------------------------------------------------------------------------|
| <i>workerController</i> | Pointer to the <b>WorkerController</b> (p. 601) that should be stopped. |
|-------------------------|-------------------------------------------------------------------------|

## Exceptions

|                                           |                             |
|-------------------------------------------|-----------------------------|
| <b>Error::ObjectDoesNotExist</b> (p. 457) | worker is not working.      |
| <b>Error::StrategyError</b> (p. 567)      | Problem sending the signal. |

Implements **BiometricEvaluation::Process::Manager** (p. 434).

## G.107.3.5 waitForWorkerExit()

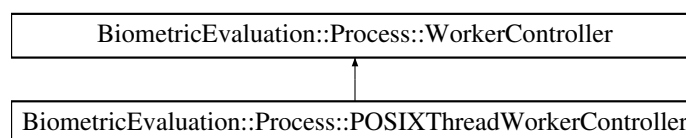
```
void BiometricEvaluation::Process::POSIXThreadManager::waitForWorkerExit () [virtual]
 Block until all Workers have exited.
 Use this method if wait=false was set during a call to startWorker(s) but now wait=true is desired.
 Implements BiometricEvaluation::Process::Manager (p. 435).
```

## G.108 BiometricEvaluation::Process::POSIXThreadWorkerController Class Reference

Decorated **Worker** (p. 595) returned from a **Process::POSIXThreadManager** (p. 480).

```
#include <be_process_posixthreadmanager.h>
```

Inheritance diagram for BiometricEvaluation::Process::POSIXThreadWorkerController:



## Public Member Functions

- void **reset** ()  
*Reuse the **Worker** (p. 595).*
- bool **isWorking** () const  
*Obtain whether or not **Worker** (p. 595) is working.*
- bool **everWorked** () const  
*Obtain whether or not this **Worker** (p. 595) has ever worked.*
- **~POSIXThreadWorkerController** ()  
*POSIXThreadWorkerController (p. 483) destructor.*

## Friends

- class **POSIXThreadManager**

## Additional Inherited Members

### G.108.1 Detailed Description

Decorated **Worker** (p. 595) returned from a **Process::POSIXThreadManager** (p. 480).

### G.108.2 Member Function Documentation

#### G.108.2.1 everWorked()

```
bool BiometricEvaluation::Process::POSIXThreadWorkerController::everWorked () const [virtual]
```

Obtain whether or not this **Worker** (p. 595) has ever worked.

Returns

true the **Worker** (p. 595) has ever or is currently working, false otherwise.

Note

**reset**() (p. 484) will change the result of this method.

Implements **BiometricEvaluation::Process::WorkerController** (p. 602).

#### G.108.2.2 isWorking()

```
bool BiometricEvaluation::Process::POSIXThreadWorkerController::isWorking () const [virtual]
```

Obtain whether or not **Worker** (p. 595) is working.

Returns

Whether or not the **Worker** (p. 595) is working.

Implements **BiometricEvaluation::Process::WorkerController** (p. 603).

### G.108.2.3 reset()

```
void BiometricEvaluation::Process::POSIXThreadWorkerController::reset () [virtual]
```

Reuse the **Worker** (p. [595](#)).

Exceptions

|                                     |                                                                 |
|-------------------------------------|-----------------------------------------------------------------|
| <b>Error::ObjectExists</b> (p. 458) | The previously started <b>Worker</b> (p. 595) is still running. |
|-------------------------------------|-----------------------------------------------------------------|

Reimplemented from **BiometricEvaluation::Process::WorkerController** (p. 603).

## G.109 BiometricEvaluation::View::AN2KViewVariableResolution::PrintPositionCoordinate Struct Reference

Offsets to the bounding boxes for the EJI, full finger views, or EJI segments.

```
#include <be_view_an2kview_varres.h>
```

### Public Attributes

- **Finger::FingerImageCode** **fingerView**
- **Finger::FingerImageCode** **segment**
- **Image::CoordinateSet** **coordinates**

### G.109.1 Detailed Description

Offsets to the bounding boxes for the EJI, full finger views, or EJI segments.

### G.109.2 Member Data Documentation

#### G.109.2.1 coordinates

```
Image::CoordinateSet BiometricEvaluation::View::AN2KViewVariableResolution::PrintPositionCoordinate↔
::coordinates
```

Two coordinates forming bounding box

#### G.109.2.2 fingerView

```
Finger::FingerImageCode BiometricEvaluation::View::AN2KViewVariableResolution::PrintPosition↔
Coordinate::fingerView
```

Full finger view being bounded

#### G.109.2.3 segment

```
Finger::FingerImageCode BiometricEvaluation::View::AN2KViewVariableResolution::PrintPosition↔
Coordinate::segment
```

Segment within full finger view bound

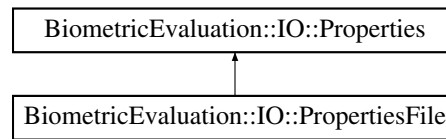
## G.110 BiometricEvaluation::IO::Properties Class Reference

Maintain key/value pairs of strings, with each property matched to one value.

```
#include <be_io_properties.h>
```

Inheritance diagram for **BiometricEvaluation::IO::Properties**:





## Public Member Functions

- **Properties** (**IO::Mode** mode= **IO::Mode::ReadWrite**, const std::map< std::string, std::string > &defaults={})  
Construct a new *Properties* (p. 486) object.
- **Properties** (const uint8\_t \*buffer, const size\_t size, **IO::Mode** mode= **IO::Mode::ReadWrite**, const std::map< std::string, std::string > &defaults={})  
Construct a new *Properties* (p. 486) object from the contents of a buffer.
- virtual void **setProperty** (const std::string &property, const std::string &value)  
Set a property with a value.
- virtual void **setPropertyFromInteger** (const std::string &property, int64\_t value)  
Set a property with an integer value.
- virtual void **setPropertyFromDouble** (const std::string &property, double value)  
Set a property with a double value.
- virtual void **setPropertyFromBoolean** (const std::string &property, bool value)  
Set a property with a boolean value.
- virtual void **removeProperty** (const std::string &property)  
Remove a property.
- virtual std::string **getProperty** (const std::string &property) const  
Retrieve a property value as a string object.
- virtual int64\_t **getPropertyAsInteger** (const std::string &property) const  
Retrieve a property value as an integer value.
- virtual double **getPropertyAsDouble** (const std::string &property) const  
Retrieve a property value as a double value.
- virtual bool **getPropertyAsBoolean** (const std::string &property) const
- std::vector< std::string > **getPropertyKeys** () const  
Retrieve a set of all property keys.
- virtual ~**Properties** ()

## Protected Member Functions

- **BiometricEvaluation::IO::Mode** **getMode** () const  
Obtain the mode of the *Properties* (p. 486) object.
- void **initWithBuffer** (const **Memory::uint8Array** &buffer, const std::map< std::string, std::string > &defaults)  
Initialize the *PropertiesMap* with the contents of a properly formatted buffer.
- void **initWithBuffer** (const uint8\_t \*const buffer, size\_t size, const std::map< std::string, std::string > &defaults)  
Initialize the *PropertiesMap* with the contents of a properly formatted buffer.

## G.110.1 Detailed Description

Maintain key/value pairs of strings, with each property matched to one value.

## G.110.2 Constructor & Destructor Documentation

### G.110.2.1 Properties() [1/2]

```
BiometricEvaluation::IO::Properties::Properties (
 IO::Mode mode = IO::Mode::ReadWrite,
 const std::map< std::string, std::string > & defaults = {})
```

Construct a new **Properties** (p. 486) object.

Parameters

|    |                 |                                         |
|----|-----------------|-----------------------------------------|
| in | <i>mode</i>     | The read/write mode of the object.      |
| in | <i>defaults</i> | Default property/value pairs to insert. |

### G.110.2.2 Properties() [2/2]

```
BiometricEvaluation::IO::Properties::Properties (
 const uint8_t * buffer,
 const size_t size,
 IO::Mode mode = IO::Mode::ReadWrite,
 const std::map< std::string, std::string > & defaults = {})
```

Construct a new **Properties** (p. 486) object from the contents of a buffer.  
The format of the buffer can be seen in **PropertiesFile** (p. 493).

Parameters

|    |                 |                                                         |
|----|-----------------|---------------------------------------------------------|
| in | <i>buffer</i>   | A buffer that contains the contents of a Property file. |
| in | <i>size</i>     | The size of buffer.                                     |
| in | <i>mode</i>     | The read/write mode of the object.                      |
| in | <i>defaults</i> | Default property/value pairs to insert.                 |

Exceptions

|                                      |                                             |
|--------------------------------------|---------------------------------------------|
| <b>Error::StrategyError</b> (p. 567) | A line in the properties file is malformed. |
|--------------------------------------|---------------------------------------------|

### G.110.2.3 ~Properties()

```
virtual BiometricEvaluation::IO::Properties::~~Properties () [virtual]
```

Destructor

G.110.3 Member Function Documentation

G.110.3.1 `getMode()`

`BiometricEvaluation::IO::Mode` `BiometricEvaluation::IO::Properties::getMode ( ) const` [protected]  
Obtain the mode of the **Properties** (p. 486) object.

Returns

Mode (`Mode::ReadOnly` (p. 127) or `Mode::ReadWrite` (p. 127))

G.110.3.2 `getProperty()`

`virtual std::string` `BiometricEvaluation::IO::Properties::getProperty (`  
    `const std::string & property ) const` [virtual]  
Retrieve a property value as a string object.

Parameters

|    |                 |                                  |
|----|-----------------|----------------------------------|
| in | <i>property</i> | The name of the property to get. |
|----|-----------------|----------------------------------|

Exceptions

|                                           |                                    |
|-------------------------------------------|------------------------------------|
| <i>Error::ObjectDoesNotExist</i> (p. 457) | The named property does not exist. |
|-------------------------------------------|------------------------------------|

G.110.3.3 `getPropertyAsDouble()`

`virtual double` `BiometricEvaluation::IO::Properties::getPropertyAsDouble (`  
    `const std::string & property ) const` [virtual]  
Retrieve a property value as a double value.

Parameters

|    |                 |                                  |
|----|-----------------|----------------------------------|
| in | <i>property</i> | The name of the property to get. |
|----|-----------------|----------------------------------|

Exceptions

|                                           |                                                                                                                        |
|-------------------------------------------|------------------------------------------------------------------------------------------------------------------------|
| <i>Error::ObjectDoesNotExist</i> (p. 457) | The named property does not exist.                                                                                     |
| <i>Error::ConversionError</i> (p. 283)    | The property value cannot be converted, due to non-numeric characters in the string, or the value is the empty string. |

**G.110.3.4 getPropertyAsInteger()**

```
virtual int64_t BiometricEvaluation::IO::Properties::getPropertyAsInteger (
 const std::string & property) const [virtual]
```

Retrieve a property value as an integer value.

Integer value strings for properties can represent either decimal or hexadecimal values, which must be preceded with either "0x" or "0X".

Parameters

|    |                 |                                  |
|----|-----------------|----------------------------------|
| in | <i>property</i> | The name of the property to get. |
|----|-----------------|----------------------------------|

Exceptions

|                                                                   |                                                                                                                        |
|-------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------|
| <b><i>Error::ObjectDoesNotExist</i></b> (p. <a href="#">457</a> ) | The named property does not exist.                                                                                     |
| <b><i>Error::ConversionError</i></b> (p. <a href="#">283</a> )    | The property value cannot be converted, due to non-numeric characters in the string, or the value is the empty string. |

**G.110.3.5 getPropertyKeys()**

```
std::vector<std::string> BiometricEvaluation::IO::Properties::getPropertyKeys () const
```

Retrieve a set of all property keys.

Returns

A vector of property key strings.

**G.110.3.6 initWithBuffer() [1/2]**

```
void BiometricEvaluation::IO::Properties::initWithBuffer (
 const Memory::uint8Array & buffer,
 const std::map< std::string, std::string > & defaults) [protected]
```

Initialize the PropertiesMap with the contents of a properly formatted buffer.

This method ensures that the PropertiesMap contains only the properties found within the buffer.

Parameters

|                 |                                |
|-----------------|--------------------------------|
| <i>buffer</i>   | Contents of a properties file. |
| <i>defaults</i> | Default property/value pairs.  |

Exceptions

|                                                              |                                    |
|--------------------------------------------------------------|------------------------------------|
| <b><i>Error::StrategyError</i></b> (p. <a href="#">567</a> ) | A line of the buffer is malformed. |
|--------------------------------------------------------------|------------------------------------|

**G.110.3.7 initWithBuffer() [2/2]**

```
void BiometricEvaluation::IO::Properties::initWithBuffer (
 const uint8_t *const buffer,
 size_t size,
 const std::map< std::string, std::string > & defaults) [protected]
```

Initialize the PropertiesMap with the contents of a properly formatted buffer.

This method ensures that the PropertiesMap contains only the properties found within the buffer.

Parameters

|                 |                                |
|-----------------|--------------------------------|
| <i>buffer</i>   | Contents of a properties file. |
| <i>size</i>     | Size of the buffer.            |
| <i>defaults</i> | Default property/value pairs.  |

Exceptions

|                                                       |                                    |
|-------------------------------------------------------|------------------------------------|
| <b>Error::StrategyError</b> (p. <a href="#">567</a> ) | A line of the buffer is malformed. |
|-------------------------------------------------------|------------------------------------|

**G.110.3.8 removeProperty()**

```
virtual void BiometricEvaluation::IO::Properties::removeProperty (
 const std::string & property) [virtual]
```

Remove a property.

Parameters

|    |                 |                                  |
|----|-----------------|----------------------------------|
| in | <i>property</i> | The name of the property to set. |
|----|-----------------|----------------------------------|

Exceptions

|                                                            |                                                                      |
|------------------------------------------------------------|----------------------------------------------------------------------|
| <b>Error::ObjectDoesNotExist</b> (p. <a href="#">457</a> ) | The named property does not exist.                                   |
| <b>Error::StrategyError</b> (p. <a href="#">567</a> )      | The <b>Properties</b> (p. <a href="#">486</a> ) object is read-only. |

**G.110.3.9 setProperty()**

```
virtual void BiometricEvaluation::IO::Properties::setProperty (
 const std::string & property,
 const std::string & value) [virtual]
```

Set a property with a value.

Both the property and value will have leading and trailing whitespace removed. If the property already exists in the set, its value will be replaced with the new value; otherwise, the property will be created.

## Parameters

|    |                 |                                         |
|----|-----------------|-----------------------------------------|
| in | <i>property</i> | The name of the property to set.        |
| in | <i>value</i>    | The value associated with the property. |

## Exceptions

|                                                              |                                                                      |
|--------------------------------------------------------------|----------------------------------------------------------------------|
| <b><i>Error::StrategyError</i></b> (p. <a href="#">567</a> ) | The <b>Properties</b> (p. <a href="#">486</a> ) object is read-only. |
|--------------------------------------------------------------|----------------------------------------------------------------------|

**G.110.3.10 setPropertyFromBoolean()**

```
virtual void BiometricEvaluation::IO::Properties::setPropertyFromBoolean (
 const std::string & property,
 bool value) [virtual]
```

Set a property with a boolean value.

The actual value to be written is implementation- defined and may not actually be preserved, but the boolean value is guaranteed to remain valid when read with `getPropertyAsBoolean()`.

## Parameters

|    |                 |                                         |
|----|-----------------|-----------------------------------------|
| in | <i>property</i> | The name of the property to set.        |
| in | <i>value</i>    | The value associated with the property. |

## Exceptions

|                                                              |                                                                      |
|--------------------------------------------------------------|----------------------------------------------------------------------|
| <b><i>Error::StrategyError</i></b> (p. <a href="#">567</a> ) | The <b>Properties</b> (p. <a href="#">486</a> ) object is read-only. |
|--------------------------------------------------------------|----------------------------------------------------------------------|

**G.110.3.11 setPropertyFromDouble()**

```
virtual void BiometricEvaluation::IO::Properties::setPropertyFromDouble (
 const std::string & property,
 double value) [virtual]
```

Set a property with a double value.

The property will have leading and trailing whitespace removed. If the property already exists in the set, its value will be replaced with the new value; otherwise the property will be created.

## Parameters

|    |                 |                                         |
|----|-----------------|-----------------------------------------|
| in | <i>property</i> | The name of the property to set.        |
| in | <i>value</i>    | The value associated with the property. |

## Exceptions

|                                      |                                                     |
|--------------------------------------|-----------------------------------------------------|
| <b>Error::StrategyError</b> (p. 567) | The <b>Properties</b> (p. 486) object is read-only. |
|--------------------------------------|-----------------------------------------------------|

**G.110.3.12 setPropertyFromInteger()**

```
virtual void BiometricEvaluation::IO::Properties::setPropertyFromInteger (
 const std::string & property,
 int64_t value) [virtual]
```

Set a property with an integer value.

The property will have leading and trailing whitespace removed. If the property already exists in the set, its value will be replaced with the new value; otherwise the property will be created.

## Parameters

|    |                 |                                         |
|----|-----------------|-----------------------------------------|
| in | <i>property</i> | The name of the property to set.        |
| in | <i>value</i>    | The value associated with the property. |

## Exceptions

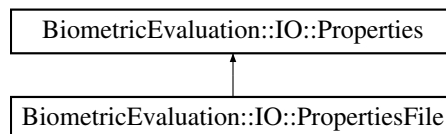
|                                      |                                                     |
|--------------------------------------|-----------------------------------------------------|
| <b>Error::StrategyError</b> (p. 567) | The <b>Properties</b> (p. 486) object is read-only. |
|--------------------------------------|-----------------------------------------------------|

**G.111 BiometricEvaluation::IO::PropertiesFile Class Reference**

A **Properties** (p. 486) object persisted in an file on disk.

```
#include <be_io_propertiesfile.h>
```

Inheritance diagram for BiometricEvaluation::IO::PropertiesFile:

**Public Member Functions**

- **PropertiesFile** (const std::string &pathname, **IO::Mode** mode= **IO::Mode::ReadOnly**, const std::map< std::string, std::string > &defaults={})  
*Construct a new **Properties** (p. 486) object from an existing or to be created properties file. The constructor will create the file when it does not exist.*
- void **sync** ()  
*Write the properties to the underlying file, synchronizing the in-memory and on-disk versions.*
- void **changeName** (const std::string &pathname)

Change the name of the **Properties** (p. 486), which means changing the name of the underlying file that stores the properties.

- **~PropertiesFile** ()
- **PropertiesFile** (const **PropertiesFile** &other)=delete  
*Copy constructor (disabled).*
- **PropertiesFile & operator=** (const **PropertiesFile** &other)=delete  
*Assignment operator (disabled).*

## Additional Inherited Members

### G.111.1 Detailed Description

A **Properties** (p. 486) object persisted in an file on disk.  
An example file might look like this:

```
* Name = John Smith
* Age = 32
* Favorite Hex Number = 0xffff
*
```

For property keys and values, leading and trailing whitespace is removed, therefore the call

```
props->setProperty(" My property ", " A Value ");
```

results in an entry in the property file as

```
* My property = A value
*
```

Therefore, the property names "Foo", " Foo", "Foo " are equivalent.

### G.111.2 Constructor & Destructor Documentation

#### G.111.2.1 PropertiesFile() [1/2]

```
BiometricEvaluation::IO::PropertiesFile::PropertiesFile (
 const std::string & pathname,
 IO::Mode mode = IO::Mode::ReadOnly,
 const std::map< std::string, std::string > & defaults = {})
```

Construct a new **Properties** (p. 486) object from an existing or to be created properties file. The constructor will create the file when it does not exist.

Parameters

|    |                 |                                               |
|----|-----------------|-----------------------------------------------|
| in | <i>pathname</i> | The path to the file to store the properties. |
| in | <i>mode</i>     | The read/write mode of the object.            |
| in | <i>defaults</i> | Default property/value pairs to insert.       |

Exceptions

|                                      |                                                             |
|--------------------------------------|-------------------------------------------------------------|
| <b>Error::StrategyError</b> (p. 567) | A line in the properties file is malformed.                 |
| <b>Error::FileError</b> (p. 316)     | An error occurred when using the underlying storage system. |



**G.111.2.2 ~PropertiesFile()**

BiometricEvaluation::IO::PropertiesFile::~~PropertiesFile ( )  
Destructor

**G.111.2.3 PropertiesFile() [2/2]**

BiometricEvaluation::IO::PropertiesFile::PropertiesFile (   
const **PropertiesFile** & *other* ) [delete]

Copy constructor (disabled).

Disabled because this object could represent a file on disk.

Parameters

|              |                                                |
|--------------|------------------------------------------------|
| <i>other</i> | <b>PropertiesFile</b> (p. 493) object to copy. |
|--------------|------------------------------------------------|

**G.111.3 Member Function Documentation****G.111.3.1 changeName()**

void BiometricEvaluation::IO::PropertiesFile::changeName (   
const std::string & *pathname* )

Change the name of the **Properties** (p. 486), which means changing the name of the underlying file that stores the properties.

Note

No check is made that the file is writeable at this time.

Parameters

|    |                 |                                                  |
|----|-----------------|--------------------------------------------------|
| in | <i>pathname</i> | The path to the <b>Properties</b> (p. 486) file. |
|----|-----------------|--------------------------------------------------|

Exceptions

|                                      |                                    |
|--------------------------------------|------------------------------------|
| <b>Error::StrategyError</b> (p. 567) | The object is read-only.           |
| <b>Error::ObjectExists</b> (p. 458)  | A file at pathname already exists. |

**G.111.3.2 operator=()**

**PropertiesFile**& BiometricEvaluation::IO::PropertiesFile::operator= (   
const **PropertiesFile** & *other* ) [delete]

Assignment operator (disabled).

Disabled because this object could represent a file on disk.

Parameters

|              |                                                  |
|--------------|--------------------------------------------------|
| <i>other</i> | <b>PropertiesFile</b> (p. 493) object to assign; |
|--------------|--------------------------------------------------|

Returns

This **PropertiesFile** (p. 493) object, now containing the contents of other.

### G.111.3.3 sync()

```
void BiometricEvaluation::IO::PropertiesFile::sync ()
```

Write the properties to the underlying file, synchronizing the in-memory and on-disk versions.

Exceptions

|                                      |                                                                            |
|--------------------------------------|----------------------------------------------------------------------------|
| <b>Error::FileError</b> (p. 316)     | An error occurred when using the underlying storage system.                |
| <b>Error::StrategyError</b> (p. 567) | The object was constructed with nullptr as the file name, or is read-only. |

## G.112 BiometricEvaluation::Feature::Sort::Quality Class Reference

```
#include <be_feature_sort.h>
```

### Public Member Functions

- **bool operator()** (const **BiometricEvaluation::Feature::MinutiaPoint** &lhs, const **BiometricEvaluation::Feature::MinutiaPoint** &rhs) const  
*MinutiaPoint* (p. 444) quality ascending comparator.

### G.112.1 Detailed Description

**Sort** (p. 113) by increasing minutiae quality

## G.113 BiometricEvaluation::Iris::INCITSView::QualitySubBlock Struct Reference

Representation of an iris quality block.

```
#include <be_iris_incitsview.h>
```

### Public Attributes

- **uint8\_t score**
- **uint16\_t vendorID**
- **uint16\_t algorithmID**

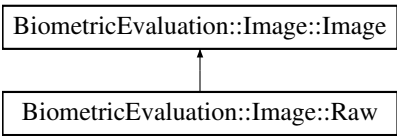
G.113.1 Detailed Description

Representation of an iris quality block.

G.114 BiometricEvaluation::Image::Raw Class Reference

An image with no encoding or compression.

```
#include <be_image_raw.h>
Inheritance diagram for BiometricEvaluation::Image::Raw:
```



Public Member Functions

- **Raw** (const uint8\_t \*data, const uint64\_t size, const **Size** dimensions, const uint32\_t colorDepth, const uint16\_t bitDepth, const **Resolution** resolution, const bool **hasAlphaChannel**)
- **Raw** (const **BiometricEvaluation::Memory::uint8Array** &data, const **Size** dimensions, const uint32\_t colorDepth, const uint16\_t bitDepth, const **Resolution** resolution, const bool **hasAlphaChannel**)
- **Memory::uint8Array** **getRawData** () const  
*Accessor for the raw image data. The data returned should not be compressed or encoded.*
- **Memory::uint8Array** **getRawGrayscaleData** (uint8\_t depth) const  
*Accessor for decompressed data in grayscale.*

Additional Inherited Members

G.114.1 Detailed Description

An image with no encoding or compression.

G.114.2 Member Function Documentation

G.114.2.1 getRawData()

```
Memory::uint8Array BiometricEvaluation::Image::Raw::getRawData () const [virtual]
Accessor for the raw image data. The data returned should not be compressed or encoded.
```

Returns

AutoArray holding raw image data.

Exceptions

|                                  |                                                 |
|----------------------------------|-------------------------------------------------|
| <b>Error::DataError</b> (p. 295) | <b>Error</b> (p. 108) decompressing image data. |
|----------------------------------|-------------------------------------------------|

Implements **BiometricEvaluation::Image::Image** (p. 361).

### G.114.2.2 getRawGrayscaleData()

**Memory::uint8Array** BiometricEvaluation::Image::Raw::getRawGrayscaleData (   
 uint8\_t depth ) const [virtual]

Accessor for decompressed data in grayscale.

Parameters

|              |                                                                                         |
|--------------|-----------------------------------------------------------------------------------------|
| <i>depth</i> | The desired bit depth of the resulting raw image. This value may either be 16, 8, or 1. |
|--------------|-----------------------------------------------------------------------------------------|

Returns

AutoArray holding raw grayscale image data.

Exceptions

|                                       |                                                     |
|---------------------------------------|-----------------------------------------------------|
| <b>Error::DataError</b> (p. 295)      | <b>Error</b> (p. 108) decompressing image data.     |
| <b>Error::NotImplemented</b> (p. 456) | Unsupported conversion based on source color depth. |
| <b>Error::ParameterError</b> (p. 474) | Invalid value for depth.                            |

Note

This method does not save a cached copy of the decompressed image because the bit depth of the image can be changed between calls.

When depth is 1, this method returns an image that uses 8 bits to represent a single pixel. The depth parameter is used to adjust the number of gray levels. When depth is 1, there are only 2 gray levels (black and white), despite using 8 bits to represent each pixel.

Alpha channels are completely ignored when converting to grayscale.

Implements **BiometricEvaluation::Image::Image** (p. 362).

## G.115 BiometricEvaluation::MPI::Receiver Class Reference

A class to represent an **MPI** (p. 145) task that receives WorkPackages containers from the **Distributor** (p. 305).

```
#include <be_mpi_receiver.h>
```

### Public Member Functions

- **Receiver** (const std::string &propertiesFileName, const std::shared\_ptr< **BiometricEvaluation::MPI::WorkPackageProcessor** > &workPackageProcessor)

*Construct a new work package receiver.*

- void **start** ()

*Start the receiving task.*

## G.115.1 Detailed Description

A class to represent an **MPI** (p. 145) task that receives WorkPackages containers from the **Distributor** (p. 305).

A receiver object depends on a set of properties contained in a file. The properties specify **MPI** (p. 145) settings, and other items. Subclasses of the class can add new properties.

Each receiver object is responsible for 1..n worker processes that are started when **Receiver::start()** (p. 499) is called. The receiver will start workers only when the distributor indicates that it has started successfully. Otherwise, the **Receiver** (p. 498) transitions to the shutdown state.

One of the optional properties is a Uniform Resource Locator (URL) for the Logsheet. If this property does not exist, no logging takes place (although applications can create their own Logsheet). If the URL is present, the framework will log at various points of processing. In the case of a FileLogsheet the framework will create more than one log file, each named after the ID of the **MPI** (p. 145) task created by the **MPI** (p. 145) runtime, and the child process created by **Receiver** (p. 498).

See also

**IO::Properties** (p. 486)

**IO::Logsheet** (p. 420)

**MPI::Distributor** (p. 305)

**Process::Worker** (p. 595)

## G.115.2 Constructor & Destructor Documentation

### G.115.2.1 Receiver()

```
BiometricEvaluation::MPI::Receiver::Receiver (
 const std::string & propertiesFileName,
 const std::shared_ptr< BiometricEvaluation::MPI::WorkPackageProcessor > & workPackageProcessor)
```

Construct a new work package receiver.

Parameters

|    |                             |                                                                             |
|----|-----------------------------|-----------------------------------------------------------------------------|
| in | <i>propertiesFileName</i>   | The name of the file containing the properties used by the receiver object. |
| in | <i>workPackageProcessor</i> | The object that will process the work received by this object.              |

Exceptions

|                                  |                                                  |
|----------------------------------|--------------------------------------------------|
| <b>Error::Exception</b> (p. 310) | An error occurred when constructing this object. |
|----------------------------------|--------------------------------------------------|

## G.115.3 Member Function Documentation

### G.115.3.1 start()

```
void BiometricEvaluation::MPI::Receiver::start ()
```

Start the receiving task.

Upon starting, the **Receiver** (p. 498) object will begin communicating with the **Distributor** (p. 305) using **MPI** (p. 145) messages. This **Receiver** (p. 498) object will send a status message back to the **Distributor** (p. 305) indicating success or failure to initialize. Success includes the startup of at least one worker process.

## G.116 BiometricEvaluation::IO::RecordStore::Record Struct Reference

### Public Member Functions

- **Record** ()
- **Record** (const std::string &key, const **Memory::uint8Array** &data)

Create a **Record** (p. 500) from the key and data.

### Public Attributes

- std::string **key**
- **Memory::uint8Array** **data**

### G.116.1 Constructor & Destructor Documentation

#### G.116.1.1 Record() [1/2]

BiometricEvaluation::IO::RecordStore::Record::Record ( )  
Default constructor.

#### G.116.1.2 Record() [2/2]

BiometricEvaluation::IO::RecordStore::Record::Record (   
const std::string & key,  
const **Memory::uint8Array** & data )

Create a **Record** (p. 500) from the key and data.

Parameters

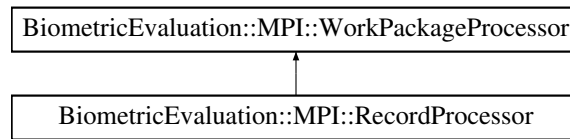
|    |             |                            |
|----|-------------|----------------------------|
| in | <i>key</i>  | The record's key.          |
| in | <i>data</i> | The record's data (value). |

## G.117 BiometricEvaluation::MPI::RecordProcessor Class Reference

An implementation of a work package processor that will extract record store keys, and optionally, values, from a **WorkPackage** (p. 606).

```
#include <be_mpi_recordprocessor.h>
```

Inheritance diagram for BiometricEvaluation::MPI::RecordProcessor:



## Public Member Functions

- **RecordProcessor** (const std::string &propertiesFileName)  
*Construct a work package processor with the given properties.*
- virtual void **processRecord** (const std::string &key)=0  
*Method implemented by child classes to perform an action using each record from the Record Store.*
- virtual void **processRecord** (const std::string &key, const **Memory::uint8Array** &value)=0  
*Method implemented by child classes to perform an action using each record from the Record Store.*
- virtual std::shared\_ptr< **WorkPackageProcessor** > **newProcessor** (std::shared\_ptr< **IO::Logsheet** > &logsheet)=0  
*Obtain an object that will process work packages. This method is part of the factory personality.*
- virtual void **performInitialization** (std::shared\_ptr< **IO::Logsheet** > &logsheet)=0  
*Initialization function to be called before work is distributed to the work package processor.*
- void **processWorkPackage** ( **MPI::WorkPackage** &workPackage)  
*Process (p. 150) the data contents of the work package. This method is part of the worker personality.*

## Protected Member Functions

- std::shared\_ptr< **MPI::RecordStoreResources** > **getResources** ()

### G.117.1 Detailed Description

An implementation of a work package processor that will extract record store keys, and optionally, values, from a **WorkPackage** (p. 606).

Subclasses of this abstract class must implement the method to process the records associated with the keys.

### G.117.2 Constructor & Destructor Documentation

#### G.117.2.1 RecordProcessor()

```
BiometricEvaluation::MPI::RecordProcessor::RecordProcessor (
 const std::string & propertiesFileName)
```

Construct a work package processor with the given properties.

A record processor uses a named record store to retrieve the data to be processed when only the key is delivered as part of a work package. When both key and value are part of the work package, there is no need to have access to the source record store.

Note

The size of a single value item is limited to  $2^{32}$  octets. If the size of the value item is larger, behavior is undefined.

## Parameters

|           |                           |                                                                 |
|-----------|---------------------------|-----------------------------------------------------------------|
| <i>in</i> | <i>propertiesFileName</i> | The name of the file containing the properties for this object. |
|-----------|---------------------------|-----------------------------------------------------------------|

## Exceptions

|                                  |                                                                    |
|----------------------------------|--------------------------------------------------------------------|
| <b>Error::Exception</b> (p. 310) | An error occurred, usually due to missing or incorrect properties. |
|----------------------------------|--------------------------------------------------------------------|

## G.117.3 Member Function Documentation

### G.117.3.1 newProcessor()

```
virtual std::shared_ptr< WorkPackageProcessor> BiometricEvaluation::MPI::RecordProcessor::new←
Processor (
```

```
 std::shared_ptr< IO::Logsheet > & logsheet) [pure virtual]
```

Obtain an object that will process work packages. This method is part of the factory personality.

## Parameters

|                 |                                                                                                                 |
|-----------------|-----------------------------------------------------------------------------------------------------------------|
| <i>logsheet</i> | A shared pointer to the <b>IO::Logsheet</b> (p. 420) that may be used to save messages generated by the object. |
|-----------------|-----------------------------------------------------------------------------------------------------------------|

## Returns

A shared pointer to the work package processor.

## Note

This method should always create a non-null **WorkPackageProcessor** (p. 608). If an error occurs during construction, throw a **Error::Exception** (p. 310) with a message to be caught and logged.

Implements **BiometricEvaluation::MPI::WorkPackageProcessor** (p. 609).

### G.117.3.2 performInitialization()

```
virtual void BiometricEvaluation::MPI::RecordProcessor::performInitialization (
 std::shared_ptr< IO::Logsheet > & logsheet) [pure virtual]
```

Initialization function to be called before work is distributed to the work package processor.

Implementations of this class can use this function to do any processing necessary before work is given to the processor, pre-forking.

This method is part of the factory personality. All state that is to be common across all package processor objects can be initialized in this method.

## Parameters

|                 |                                                                                                                 |
|-----------------|-----------------------------------------------------------------------------------------------------------------|
| <i>logsheet</i> | A shared pointer to the <b>IO::Logsheet</b> (p. 420) that may be used to save messages generated by the object. |
|-----------------|-----------------------------------------------------------------------------------------------------------------|



## Exceptions

|                                  |                                                                                                                  |
|----------------------------------|------------------------------------------------------------------------------------------------------------------|
| <b>Error::Exception</b> (p. 310) | An implementation specific error occurred. The exception string will be logged by the <b>Framework</b> (p. 117). |
|----------------------------------|------------------------------------------------------------------------------------------------------------------|

Implements **BiometricEvaluation::MPI::WorkPackageProcessor** (p. 609).

**G.117.3.3 processRecord()** [1/2]

```
virtual void BiometricEvaluation::MPI::RecordProcessor::processRecord (
 const std::string & key) [pure virtual]
```

Method implemented by child classes to perform an action using each record from the Record Store.

The source RecordStore must be accessible to the the implementation as the value for each key is not included.

## Parameters

|    |     |                                                             |
|----|-----|-------------------------------------------------------------|
| in | key | The key associated with the record that is to be processed. |
|----|-----|-------------------------------------------------------------|

## Exceptions

|                                  |                                                                                                    |
|----------------------------------|----------------------------------------------------------------------------------------------------|
| <b>Error::Exception</b> (p. 310) | An error occurred processing the record: Missing record, input/output error, or memory allocation. |
|----------------------------------|----------------------------------------------------------------------------------------------------|

**G.117.3.4 processRecord()** [2/2]

```
virtual void BiometricEvaluation::MPI::RecordProcessor::processRecord (
 const std::string & key,
 const Memory::uint8Array & value) [pure virtual]
```

Method implemented by child classes to perform an action using each record from the Record Store.

## Parameters

|    |       |                                                             |
|----|-------|-------------------------------------------------------------|
| in | key   | The key associated with the record that is to be processed. |
| in | value | The data from the record that is to be processed.           |

## Exceptions

|                                  |                                                                                                                        |
|----------------------------------|------------------------------------------------------------------------------------------------------------------------|
| <b>Error::Exception</b> (p. 310) | An fatal error occurred when processing the work package; the processing responsible for this object should shut down. |
|----------------------------------|------------------------------------------------------------------------------------------------------------------------|

### G.117.3.5 processWorkPackage()

```
void BiometricEvaluation::MPI::RecordProcessor::processWorkPackage (
 MPI::WorkPackage & workPackage) [virtual]
```

**Process** (p. 150) the data contents of the work package. This method is part of the worker personality.

Parameters

|    |                    |                   |
|----|--------------------|-------------------|
| in | <i>workPackage</i> | The work package. |
|----|--------------------|-------------------|

Exceptions

|                                  |                                                                                                                        |
|----------------------------------|------------------------------------------------------------------------------------------------------------------------|
| <b>Error::Exception</b> (p. 310) | An fatal error occurred when processing the work package; the processing responsible for this object should shut down. |
|----------------------------------|------------------------------------------------------------------------------------------------------------------------|

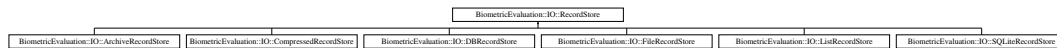
Implements **BiometricEvaluation::MPI::WorkPackageProcessor** (p. 610).

## G.118 BiometricEvaluation::IO::RecordStore Class Reference

A class to represent a data storage mechanism.

```
#include <be_io_recordstore.h>
```

Inheritance diagram for BiometricEvaluation::IO::RecordStore:



### Classes

- struct **Record**

### Public Types

- enum **Kind** {  
**Kind::BerkeleyDB**, **Kind::Archive**, **Kind::File**, **Kind::SQLite**,  
**Kind::Compressed**, **Kind::List**, **Kind::Default** = BerkeleyDB }
- using **Record** = struct **Record**
- using **iterator** = **IO::RecordStoreIterator**

### Public Member Functions

- virtual std::string **getDescription** () const =0
- virtual unsigned int **getCount** () const =0
- virtual std::string **getPathname** () const =0
- virtual void **move** (const std::string &pathname)=0  
*Move the **RecordStore** (p. 504).*
- virtual void **changeDescription** (const std::string &description)=0
- virtual uint64\_t **getSpaceUsed** () const =0  
*Obtain real storage utilization.*

- virtual void **sync** () const =0
  - virtual void **insert** (const std::string &key, const **Memory::uint8Array** &data)
  - virtual void **insert** (const std::string &key, const void \*const data, const uint64\_t size)=0
  - virtual void **remove** (const std::string &key)=0
  - virtual **Memory::uint8Array** **read** (const std::string &key) const =0
- Read a complete record from a store.*
- virtual void **replace** (const std::string &key, const **Memory::uint8Array** &data)
  - virtual void **replace** (const std::string &key, const void \*const data, const uint64\_t size)
  - virtual uint64\_t **length** (const std::string &key) const =0
  - virtual void **flush** (const std::string &key) const =0
  - virtual **RecordStore::Record** **sequence** (int cursor= **BE\_RECSTORE\_SEQ\_NEXT**)=0
- Sequence through a **RecordStore** (p. 504), returning the key/data pairs.*
- virtual std::string **sequenceKey** (int cursor= **BE\_RECSTORE\_SEQ\_NEXT**)=0
- Sequence through a **RecordStore** (p. 504), returning the key.*
- virtual void **setCursorAtKey** (const std::string &key)=0
  - virtual bool **containsKey** (const std::string &key) const
- Determines whether the **RecordStore** (p. 504) contains an element with the specified key.*
- virtual **iterator** **begin** () noexcept
  - virtual **iterator** **end** () noexcept

## Static Public Member Functions

- static std::shared\_ptr< **RecordStore** > **openRecordStore** (const std::string &pathname, **IO::Mode** mode= **Mode::ReadOnly**)
- Open an existing **RecordStore** (p. 504) and return a managed pointer to the the object representing that store.*
- static std::shared\_ptr< **RecordStore** > **createRecordStore** (const std::string &pathname, const std::string &description, const **IO::RecordStore::Kind** &kind)
- Create a new **RecordStore** (p. 504) and return a managed pointer to the the object representing that store.*
- static void **removeRecordStore** (const std::string &pathname)
  - static void **mergeRecordStores** (const std::string &mergePathname, const std::string &description, const **IO::RecordStore::Kind** &kind, const std::vector< std::string > &pathnames)
- Create a new **RecordStore** (p. 504) that contains the contents of several other **RecordStores**.*

## Static Public Attributes

- static const std::string **INVALIDKEYCHARS**
- static const int **BE\_RECSTORE\_SEQ\_START** = 1
- static const int **BE\_RECSTORE\_SEQ\_NEXT** = 2

### G.118.1 Detailed Description

A class to represent a data storage mechanism.

A **RecordStore** (p. 504) is an abstraction that associates keys with a specific data item. Implementations of this abstraction can store the records in any format supported by the operating system, such as files or databases, rooted in the file system.

Certain characters are prohibited in the key string. See **IO::RecordStore::INVALIDKEYCHARS** (p. 517). A key string cannot begin with the space character.

See also

**IO::ArchiveRecordStore** (p. 223), **IO::DBRecordStore** (p. 295), **IO::FileRecordStore** (p. 325).

## G.118.2 Member Enumeration Documentation

### G.118.2.1 Kind

enum **BiometricEvaluation::IO::RecordStore::Kind** [strong]  
Possible types of **RecordStore** (p. 504)

Enumerator

|            |                                            |
|------------|--------------------------------------------|
| BerkeleyDB | <b>DBRecordStore</b> (p. 295)              |
| Archive    | <b>ArchiveRecordStore</b> (p. 223)         |
| File       | <b>FileRecordStore</b> (p. 325)            |
| SQLite     | <b>SQLiteRecordStore</b> (p. 554)          |
| Compressed | <b>CompressedRecordStore</b> (p. 261)      |
| List       | <b>ListRecordStore</b> (p. 413)            |
| Default    | ”Default” <b>RecordStore</b> (p. 504) kind |

## G.118.3 Member Function Documentation

### G.118.3.1 begin()

virtual **iterator** **BiometricEvaluation::IO::RecordStore::begin** ( ) [virtual], [noexcept]

Returns

Iterator to the first record.

### G.118.3.2 changeDescription()

virtual void **BiometricEvaluation::IO::RecordStore::changeDescription** (   
const std::string & *description* ) [pure virtual]

Change the description of the **RecordStore** (p. 504).

Parameters

|    |                    |                      |
|----|--------------------|----------------------|
| in | <i>description</i> | The new description. |
|----|--------------------|----------------------|

Exceptions

|                                      |                                                             |
|--------------------------------------|-------------------------------------------------------------|
| <b>Error::StrategyError</b> (p. 567) | An error occurred when using the underlying storage system. |
|--------------------------------------|-------------------------------------------------------------|

Implemented in **BiometricEvaluation::IO::ArchiveRecordStore** (p. 226), **BiometricEvaluation::IO::ListRecordStore** (p. 414), **BiometricEvaluation::IO::FileRecordStore** (p. 327), **BiometricEvaluation::IO::DBRecordStore** (p. 297), **BiometricEvaluation::IO::CompressedRecordStore** (p. 264), and **BiometricEvaluation::IO::SQLiteRecordStore** (p. 555).

### G.118.3.3 containsKey()

```
virtual bool BiometricEvaluation::IO::RecordStore::containsKey (
 const std::string & key) const [virtual]
```

Determines whether the **RecordStore** (p. 504) contains an element with the specified key.

Parameters

|            |                    |
|------------|--------------------|
| <i>key</i> | The key to locate. |
|------------|--------------------|

Returns

True if the **RecordStore** (p. 504) contains an element with the key, false otherwise.

### G.118.3.4 createRecordStore()

```
static std::shared_ptr< RecordStore> BiometricEvaluation::IO::RecordStore::createRecordStore (
 const std::string & pathname,
 const std::string & description,
 const IO::RecordStore::Kind & kind) [static]
```

Create a new **RecordStore** (p. 504) and return a managed pointer to the the object representing that store.

The allocated object will be automatically freed when the returned pointer goes out of scope. Applications should not delete the object.

Parameters

|    |                    |                                                        |
|----|--------------------|--------------------------------------------------------|
| in | <i>pathname</i>    | The directory of the store to be created.              |
| in | <i>description</i> | The description of the store to be created.            |
| in | <i>kind</i>        | The kind of <b>RecordStore</b> (p. 504) to be created. |

Returns

An managed pointer to the object representing the created store.

Exceptions

|                                           |                                                             |
|-------------------------------------------|-------------------------------------------------------------|
| <b>Error::ObjectDoesNotExist</b> (p. 457) | The <b>RecordStore</b> (p. 504) does not exist.             |
| <b>Error::StrategyError</b> (p. 567)      | An error occurred when using the underlying storage system. |

**G.118.3.5 end()**

```
virtual iterator BiometricEvaluation::IO::RecordStore::end () [virtual], [noexcept]
```

Returns

Iterator past the last record.

**G.118.3.6 flush()**

```
virtual void BiometricEvaluation::IO::RecordStore::flush (
 const std::string & key) const [pure virtual]
```

Commit the record's data to storage.

Parameters

|    |     |                                      |
|----|-----|--------------------------------------|
| in | key | The key of the record to be flushed. |
|----|-----|--------------------------------------|

Exceptions

|                                           |                                                             |
|-------------------------------------------|-------------------------------------------------------------|
| <b>Error::ObjectDoesNotExist</b> (p. 457) | A record for the key does not exist.                        |
| <b>Error::StrategyError</b> (p. 567)      | An error occurred when using the underlying storage system. |

Implemented in **BiometricEvaluation::IO::CompressedRecordStore** (p. 264), **BiometricEvaluation::IO::ArchiveRecordStore** (p. 226), **BiometricEvaluation::IO::FileRecordStore** (p. 328), **BiometricEvaluation::IO::DBRecordStore** (p. 298), **BiometricEvaluation::IO::ListRecordStore** (p. 415), and **BiometricEvaluation::IO::SQLiteRecordStore** (p. 555).

**G.118.3.7 getCount()**

```
virtual unsigned int BiometricEvaluation::IO::RecordStore::getCount () const [pure virtual]
```

Obtain the number of items in the **RecordStore** (p. 504).

Returns

The number of items in the **RecordStore** (p. 504).

Implemented in **BiometricEvaluation::IO::ArchiveRecordStore** (p. 227), **BiometricEvaluation::IO::ListRecordStore** (p. 415), **BiometricEvaluation::IO::FileRecordStore** (p. 328), **BiometricEvaluation::IO::DBRecordStore** (p. 298), **BiometricEvaluation::IO::CompressedRecordStore** (p. 265), and **BiometricEvaluation::IO::SQLiteRecordStore** (p. 555).

**G.118.3.8 getDescription()**

```
virtual std::string BiometricEvaluation::IO::RecordStore::getDescription () const [pure virtual]
```

Obtain a textual description of the **RecordStore** (p. 504).

## Returns

The **RecordStore** (p. 504)'s description.

Implemented in **BiometricEvaluation::IO::ArchiveRecordStore** (p. 227), **BiometricEvaluation::IO::ListRecordStore** (p. 415), **BiometricEvaluation::IO::FileRecordStore** (p. 328), **BiometricEvaluation::IO::DBRecordStore** (p. 298), **BiometricEvaluation::IO::CompressedRecordStore** (p. 265), and **BiometricEvaluation::IO::SQLiteRecordStore** (p. 556).

## G.118.3.9 getPathname()

```
virtual std::string BiometricEvaluation::IO::RecordStore::getPathname () const [pure virtual]
```

Return the path name of the **RecordStore** (p. 504).

## Returns

Where in the file system the **RecordStore** (p. 504) is located.

Implemented in **BiometricEvaluation::IO::ArchiveRecordStore** (p. 227), **BiometricEvaluation::IO::ListRecordStore** (p. 415), **BiometricEvaluation::IO::FileRecordStore** (p. 328), **BiometricEvaluation::IO::DBRecordStore** (p. 298), **BiometricEvaluation::IO::CompressedRecordStore** (p. 265), and **BiometricEvaluation::IO::SQLiteRecordStore** (p. 556).

## G.118.3.10 getSpaceUsed()

```
virtual uint64_t BiometricEvaluation::IO::RecordStore::getSpaceUsed () const [pure virtual]
```

Obtain real storage utilization.

The amount of disk space used, for example. This is the actual space allocated by the underlying storage mechanism, in bytes.

## Returns

The amount of backing storage used by the **RecordStore** (p. 504).

## Exceptions

|                                      |                                                             |
|--------------------------------------|-------------------------------------------------------------|
| <b>Error::StrategyError</b> (p. 567) | An error occurred when using the underlying storage system. |
|--------------------------------------|-------------------------------------------------------------|

Implemented in **BiometricEvaluation::IO::ArchiveRecordStore** (p. 227), **BiometricEvaluation::IO::FileRecordStore** (p. 328), **BiometricEvaluation::IO::ListRecordStore** (p. 415), **BiometricEvaluation::IO::DBRecordStore** (p. 298), **BiometricEvaluation::IO::CompressedRecordStore** (p. 265), and **BiometricEvaluation::IO::SQLiteRecordStore** (p. 556).

## G.118.3.11 insert() [1/2]

```
virtual void BiometricEvaluation::IO::RecordStore::insert (
 const std::string & key,
 const Memory::uint8Array & data) [virtual]
```

Insert a record into the store.

## Parameters

|    |             |                                       |
|----|-------------|---------------------------------------|
| in | <i>key</i>  | The key of the record to be inserted. |
| in | <i>data</i> | The data for the record.              |

## Exceptions

|                                                       |                                                                                                                                      |
|-------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------|
| <b>Error::ObjectExists</b> (p. <a href="#">458</a> )  | A record with the given key is already present.                                                                                      |
| <b>Error::StrategyError</b> (p. <a href="#">567</a> ) | The <b>RecordStore</b> (p. <a href="#">504</a> ) is opened read-only, or an error occurred when using the underlying storage system. |

**G.118.3.12 insert()** [2/2]

```
virtual void BiometricEvaluation::IO::RecordStore::insert (
 const std::string & key,
 const void *const data,
 const uint64_t size) [pure virtual]
```

Insert a record into the store.

## Parameters

|    |             |                                       |
|----|-------------|---------------------------------------|
| in | <i>key</i>  | The key of the record to be inserted. |
| in | <i>data</i> | The data for the record.              |
| in | <i>size</i> | The size of the record, in bytes.     |

## Exceptions

|                                                       |                                                                                                                                      |
|-------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------|
| <b>Error::ObjectExists</b> (p. <a href="#">458</a> )  | A record with the given key is already present.                                                                                      |
| <b>Error::StrategyError</b> (p. <a href="#">567</a> ) | The <b>RecordStore</b> (p. <a href="#">504</a> ) is opened read-only, or an error occurred when using the underlying storage system. |

Implemented in **BiometricEvaluation::IO::CompressedRecordStore** (p. [266](#)), **BiometricEvaluation::IO::ArchiveRecordStore** (p. [228](#)), **BiometricEvaluation::IO::DBRecordStore** (p. [299](#)), **BiometricEvaluation::IO::FileRecordStore** (p. [329](#)), **BiometricEvaluation::IO::ListRecordStore** (p. [416](#)), and **BiometricEvaluation::IO::SQLiteRecordStore** (p. [556](#)).

**G.118.3.13 length()**

```
virtual uint64_t BiometricEvaluation::IO::RecordStore::length (
 const std::string & key) const [pure virtual]
```

Return the length of a record.

## Parameters

|    |            |                        |
|----|------------|------------------------|
| in | <i>key</i> | The key of the record. |
|----|------------|------------------------|



## Returns

The record length.

## Exceptions

|                                           |                                                             |
|-------------------------------------------|-------------------------------------------------------------|
| <b>Error::ObjectDoesNotExist</b> (p. 457) | A record for the key does not exist.                        |
| <b>Error::StrategyError</b> (p. 567)      | An error occurred when using the underlying storage system. |

Implemented in **BiometricEvaluation::IO::CompressedRecordStore** (p. 266), **BiometricEvaluation::IO::ArchiveRecordStore** (p. 228), **BiometricEvaluation::IO::FileRecordStore** (p. 329), **BiometricEvaluation::IO::DBRecordStore** (p. 299), **BiometricEvaluation::IO::ListRecordStore** (p. 416), and **BiometricEvaluation::IO::SQLiteRecordStore** (p. 557).

## G.118.3.14 mergeRecordStores()

```
static void BiometricEvaluation::IO::RecordStore::mergeRecordStores (
 const std::string & mergePathname,
 const std::string & description,
 const IO::RecordStore::Kind & kind,
 const std::vector< std::string > & pathnames) [static]
```

Create a new **RecordStore** (p. 504) that contains the contents of several other RecordStores.

## Parameters

|    |                      |                                                                                                                                             |
|----|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------|
| in | <i>mergePathname</i> | The path name of the new <b>RecordStore</b> (p. 504) that will be created.                                                                  |
| in | <i>description</i>   | The text used to describe the new <b>RecordStore</b> (p. 504).                                                                              |
| in | <i>kind</i>          | The kind of the new, merged <b>RecordStore</b> (p. 504).                                                                                    |
| in | <i>pathnames</i>     | Vector of path names to RecordStores to open. These are the RecordStores that will be merged to create the new <b>RecordStore</b> (p. 504). |

## Exceptions

|                                      |                                                                |
|--------------------------------------|----------------------------------------------------------------|
| <b>Error::ObjectExists</b> (p. 458)  | A <b>RecordStore</b> (p. 504) at mergePathname already exists. |
| <b>Error::StrategyError</b> (p. 567) | An error occurred when using the underlying storage system.    |

## G.118.3.15 move()

```
virtual void BiometricEvaluation::IO::RecordStore::move (
 const std::string & pathname) [pure virtual]
```

Move the **RecordStore** (p. 504).

The **RecordStore** (p. 504) can be moved to a new path in the file system.

## Parameters

|    |                 |                                                  |
|----|-----------------|--------------------------------------------------|
| in | <i>pathname</i> | The new path of the <b>RecordStore</b> (p. 504). |
|----|-----------------|--------------------------------------------------|

## Exceptions

|                                      |                                                             |
|--------------------------------------|-------------------------------------------------------------|
| <b>Error::StrategyError</b> (p. 567) | An error occurred when using the underlying storage system. |
|--------------------------------------|-------------------------------------------------------------|

Implemented in **BiometricEvaluation::IO::CompressedRecordStore** (p. 267), **BiometricEvaluation::IO::ArchiveRecordStore** (p. 229), **BiometricEvaluation::IO::FileRecordStore** (p. 330), **BiometricEvaluation::IO::ListRecordStore** (p. 417), **BiometricEvaluation::IO::DBRecordStore** (p. 300), and **BiometricEvaluation::IO::SQLiteRecordStore** (p. 557).

**G.118.3.16 openRecordStore()**

```
static std::shared_ptr< RecordStore> BiometricEvaluation::IO::RecordStore::openRecordStore (
 const std::string & pathname,
 IO::Mode mode = Mode::ReadOnly) [static]
```

Open an existing **RecordStore** (p. 504) and return a managed pointer to the the object representing that store.

Applications can open existing record stores without the need to know what type of **RecordStore** (p. 504) it is.

The allocated object will be automatically freed when the returned pointer goes out of scope. Applications should not delete the object.

## Parameters

|    |                 |                                                                      |
|----|-----------------|----------------------------------------------------------------------|
| in | <i>pathname</i> | The path name of the store to be opened.                             |
| in | <i>mode</i>     | The type of access a client of this <b>RecordStore</b> (p. 504) has. |

## Returns

An object representing the existing store.

## Exceptions

|                                           |                                                             |
|-------------------------------------------|-------------------------------------------------------------|
| <b>Error::ObjectDoesNotExist</b> (p. 457) | The <b>RecordStore</b> (p. 504) does not exist.             |
| <b>Error::StrategyError</b> (p. 567)      | An error occurred when using the underlying storage system. |

**G.118.3.17 read()**

```
virtual Memory::uint8Array BiometricEvaluation::IO::RecordStore::read (
 const std::string & key) const [pure virtual]
```

Read a complete record from a store.

The AutoArray will be resized to match the size of the data.

## Parameters

|    |            |                                   |
|----|------------|-----------------------------------|
| in | <i>key</i> | The key of the record to be read. |
|----|------------|-----------------------------------|

## Returns

The record associated with the key.

## Exceptions

|                                           |                                                             |
|-------------------------------------------|-------------------------------------------------------------|
| <b>Error::ObjectDoesNotExist</b> (p. 457) | A record for the key does not exist.                        |
| <b>Error::StrategyError</b> (p. 567)      | An error occurred when using the underlying storage system. |

Implemented in **BiometricEvaluation::IO::CompressedRecordStore** (p. 267), **BiometricEvaluation::IO::ArchiveRecordStore** (p. 230), **BiometricEvaluation::IO::FileRecordStore** (p. 330), **BiometricEvaluation::IO::DBRecordStore** (p. 300), **BiometricEvaluation::IO::ListRecordStore** (p. 417), and **BiometricEvaluation::IO::SQLiteRecordStore** (p. 558).

## G.118.3.18 remove()

```
virtual void BiometricEvaluation::IO::RecordStore::remove (
 const std::string & key) [pure virtual]
```

Remove a record from the store.

## Parameters

|    |     |                                      |
|----|-----|--------------------------------------|
| in | key | The key of the record to be removed. |
|----|-----|--------------------------------------|

## Exceptions

|                                           |                                                             |
|-------------------------------------------|-------------------------------------------------------------|
| <b>Error::ObjectDoesNotExist</b> (p. 457) | A record for the key does not exist.                        |
| <b>Error::StrategyError</b> (p. 567)      | An error occurred when using the underlying storage system. |

Implemented in **BiometricEvaluation::IO::CompressedRecordStore** (p. 268), **BiometricEvaluation::IO::ArchiveRecordStore** (p. 230), **BiometricEvaluation::IO::DBRecordStore** (p. 301), **BiometricEvaluation::IO::FileRecordStore** (p. 331), **BiometricEvaluation::IO::ListRecordStore** (p. 418), and **BiometricEvaluation::IO::SQLiteRecordStore** (p. 558).

## G.118.3.19 removeRecordStore()

```
static void BiometricEvaluation::IO::RecordStore::removeRecordStore (
 const std::string & pathname) [static]
```

Remove a **RecordStore** (p. 504) by deleting all persistent data associated with the store.

## Parameters

|    |          |                                                       |
|----|----------|-------------------------------------------------------|
| in | pathname | The name of the existing <b>RecordStore</b> (p. 504). |
|----|----------|-------------------------------------------------------|

## Exceptions

|                                                            |                                                             |
|------------------------------------------------------------|-------------------------------------------------------------|
| <b>Error::ObjectDoesNotExist</b> (p. <a href="#">457</a> ) | A record with the given key does not exist.                 |
| <b>Error::StrategyError</b> (p. <a href="#">567</a> )      | An error occurred when using the underlying storage system. |

**G.118.3.20** `replace()` [1/2]

```
virtual void BiometricEvaluation::IO::RecordStore::replace (
 const std::string & key,
 const Memory::uint8Array & data) [virtual]
```

Replace a complete record in a **RecordStore** (p. [504](#)).

## Parameters

|    |             |                                       |
|----|-------------|---------------------------------------|
| in | <i>key</i>  | The key of the record to be replaced. |
| in | <i>data</i> | The data for the record.              |

## Exceptions

|                                                            |                                                                                                                                      |
|------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------|
| <b>Error::ObjectDoesNotExist</b> (p. <a href="#">457</a> ) | A record for the key does not exist.                                                                                                 |
| <b>Error::StrategyError</b> (p. <a href="#">567</a> )      | The <b>RecordStore</b> (p. <a href="#">504</a> ) is opened read-only, or an error occurred when using the underlying storage system. |

**G.118.3.21** `replace()` [2/2]

```
virtual void BiometricEvaluation::IO::RecordStore::replace (
 const std::string & key,
 const void *const data,
 const uint64_t size) [virtual]
```

Replace a complete record in a **RecordStore** (p. [504](#)).

## Parameters

|    |             |                                       |
|----|-------------|---------------------------------------|
| in | <i>key</i>  | The key of the record to be replaced. |
| in | <i>data</i> | The data for the record.              |
| in | <i>size</i> | The size of the record, in bytes.     |

## Exceptions

|                                                            |                                                                                                                                      |
|------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------|
| <b>Error::ObjectDoesNotExist</b> (p. <a href="#">457</a> ) | A record for the key does not exist.                                                                                                 |
| <b>Error::StrategyError</b> (p. <a href="#">567</a> )      | The <b>RecordStore</b> (p. <a href="#">504</a> ) is opened read-only, or an error occurred when using the underlying storage system. |

Reimplemented in **BiometricEvaluation::IO::FileRecordStore** (p. 331), and **BiometricEvaluation::IO::ListRecordStore** (p. 418).

### G.118.3.22 sequence()

```
virtual RecordStore::Record BiometricEvaluation::IO::RecordStore::sequence (
 int cursor = BE_RECSTORE_SEQ_NEXT) [pure virtual]
```

Sequence through a **RecordStore** (p. 504), returning the key/data pairs.

Sequencing means to start at some point in the store and return the record, then repeatedly calling the function to return the next record. The starting point is typically the first record, and is set to that when the **RecordStore** (p. 504) object is created. The starting point can be reset by calling this method with the cursor parameter set to BE\_RECSTORE\_SEQ\_START.

Parameters

|    |               |                                                                  |
|----|---------------|------------------------------------------------------------------|
| in | <i>cursor</i> | The location within the sequence of the key/data pair to return. |
|----|---------------|------------------------------------------------------------------|

Returns

The record that is currently in sequence.

Exceptions

|                                           |                                                             |
|-------------------------------------------|-------------------------------------------------------------|
| <b>Error::ObjectDoesNotExist</b> (p. 457) | End of sequencing.                                          |
| <b>Error::StrategyError</b> (p. 567)      | An error occurred when using the underlying storage system. |

Implemented in **BiometricEvaluation::IO::CompressedRecordStore** (p. 268), **BiometricEvaluation::IO::ArchiveRecordStore** (p. 231), **BiometricEvaluation::IO::FileRecordStore** (p. 331), **BiometricEvaluation::IO::ListRecordStore** (p. 418), **BiometricEvaluation::IO::DBRecordStore** (p. 301), and **BiometricEvaluation::IO::SQLiteRecordStore** (p. 559).

### G.118.3.23 sequenceKey()

```
virtual std::string BiometricEvaluation::IO::RecordStore::sequenceKey (
 int cursor = BE_RECSTORE_SEQ_NEXT) [pure virtual]
```

Sequence through a **RecordStore** (p. 504), returning the key.

Sequencing means to start at some point in the store and return the key, then repeatedly calling the function to return the next key. The starting point is typically the first record, and is set to that when the **RecordStore** (p. 504) object is created. The starting point can be reset by calling this method with the cursor parameter set to BE\_RECSTORE\_SEQ\_START.

Parameters

|    |               |                                                                  |
|----|---------------|------------------------------------------------------------------|
| in | <i>cursor</i> | The location within the sequence of the key/data pair to return. |
|----|---------------|------------------------------------------------------------------|

## Returns

The key of the currently sequenced record.

## Exceptions

|                                           |                                                             |
|-------------------------------------------|-------------------------------------------------------------|
| <b>Error::ObjectDoesNotExist</b> (p. 457) | End of sequencing.                                          |
| <b>Error::StrategyError</b> (p. 567)      | An error occurred when using the underlying storage system. |

Implemented in **BiometricEvaluation::IO::CompressedRecordStore** (p. 269), **BiometricEvaluation::IO::ArchiveRecordStore** (p. 231), **BiometricEvaluation::IO::FileRecordStore** (p. 332), **BiometricEvaluation::IO::ListRecordStore** (p. 419), **BiometricEvaluation::IO::DBRecordStore** (p. 301), and **BiometricEvaluation::IO::SQLiteRecordStore** (p. 559).

## G.118.3.24 setCursorAtKey()

```
virtual void BiometricEvaluation::IO::RecordStore::setCursorAtKey (
 const std::string & key) [pure virtual]
```

Set the sequence cursor to an arbitrary position within the **RecordStore** (p. 504), starting at key. Key will be the first record returned from the next call to **sequence()** (p. 515).

## Parameters

|    |     |                                                                                                          |
|----|-----|----------------------------------------------------------------------------------------------------------|
| in | key | The key of the record which will be returned by the first subsequent call to <b>sequence()</b> (p. 515). |
|----|-----|----------------------------------------------------------------------------------------------------------|

## Exceptions

|                                           |                                                             |
|-------------------------------------------|-------------------------------------------------------------|
| <b>Error::ObjectDoesNotExist</b> (p. 457) | A record for the key does not exist.                        |
| <b>Error::StrategyError</b> (p. 567)      | An error occurred when using the underlying storage system. |

Implemented in **BiometricEvaluation::IO::CompressedRecordStore** (p. 269), **BiometricEvaluation::IO::ArchiveRecordStore** (p. 232), **BiometricEvaluation::IO::FileRecordStore** (p. 332), **BiometricEvaluation::IO::ListRecordStore** (p. 419), **BiometricEvaluation::IO::DBRecordStore** (p. 302), and **BiometricEvaluation::IO::SQLiteRecordStore** (p. 560).

## G.118.3.25 sync()

```
virtual void BiometricEvaluation::IO::RecordStore::sync () const [pure virtual]
```

Synchronize the entire record store to persistent storage.

## Exceptions

|                                      |                                                             |
|--------------------------------------|-------------------------------------------------------------|
| <b>Error::StrategyError</b> (p. 567) | An error occurred when using the underlying storage system. |
|--------------------------------------|-------------------------------------------------------------|

Implemented in **BiometricEvaluation::IO::FileRecordStore** (p. 333), **BiometricEvaluation::IO::DBRecordStore** (p. 302), **BiometricEvaluation::IO::CompressedRecordStore** (p. 270), **BiometricEvaluation::IO::ArchiveRecordStore** (p. 232), **BiometricEvaluation::IO::ListRecordStore** (p. 420), and

**BiometricEvaluation::IO::SQLiteRecordStore** (p. 560).

## G.118.4 Member Data Documentation

### G.118.4.1 BE\_RECSTORE\_SEQ\_NEXT

```
const int BiometricEvaluation::IO::RecordStore::BE_RECSTORE_SEQ_NEXT = 2 [static]
```

Tell sequence to sequence from current position

### G.118.4.2 BE\_RECSTORE\_SEQ\_START

```
const int BiometricEvaluation::IO::RecordStore::BE_RECSTORE_SEQ_START = 1 [static]
```

Tell **sequence()** (p. 515) to sequence from beginning

### G.118.4.3 INVALIDKEYCHARS

```
const std::string BiometricEvaluation::IO::RecordStore::INVALIDKEYCHARS [static]
```

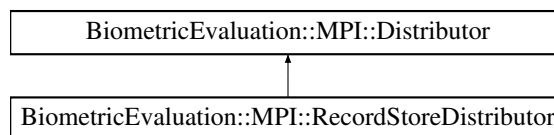
The set of prohibited characters in a key: '/', '\', '\*', '&'

## G.119 BiometricEvaluation::MPI::RecordStoreDistributor Class Reference

An implementation of the MPI::Distributor abstraction that uses a record store for input to create the work packages.

```
#include <be_mpi_recordstoredistributor.h>
```

Inheritance diagram for BiometricEvaluation::MPI::RecordStoreDistributor:



## Public Member Functions

- **RecordStoreDistributor** (const std::string &propertiesFileName, const bool includeValues)

*Construct a distributor using the named properties.*

## Static Public Attributes

- static const std::string **CHECKPOINTLASTKEY**
- static const std::string **CHECKPOINTNUMKEYS**

## Protected Member Functions

- void **createWorkPackage** ( **MPI::WorkPackage** &workPackage)  
*Create a work package for distribution.*
- void **checkpointSave** (const std::string &reason)  
*Create a checkpoint state.*
- void **checkpointRestore** ()  
*Restore from a checkpoint state.*

### G.119.1 Detailed Description

An implementation of the **MPI::Distributor** abstraction that uses a record store for input to create the work packages.

This class supports checkpointing when an early exit is requested, allowing all workers to complete their current work package.

See **MPI::Distributor** (p. 305)

### G.119.2 Constructor & Destructor Documentation

#### G.119.2.1 RecordStoreDistributor()

```
BiometricEvaluation::MPI::RecordStoreDistributor::RecordStoreDistributor (
 const std::string & propertiesFileName,
 const bool includeValues)
```

Construct a distributor using the named properties.

The distributor object is based on the properties given in the file. The name of the input record store must be one of the properties.

The work package sent to Receivers can contain either RecordStore keys, or key/value pairs.

#### Note

The size of a single value item is limited to  $2^{32}$  octets. If the size of the value item is larger, behavior is undefined.

#### Parameters

|    |                           |                                                                                         |
|----|---------------------------|-----------------------------------------------------------------------------------------|
| in | <i>propertiesFileName</i> | The file containing the properties.                                                     |
| in | <i>includeValues</i>      | true if both the key and value items are included in the work package, false otherwise. |

#### Exceptions

|                                  |                                                                    |
|----------------------------------|--------------------------------------------------------------------|
| <b>Error::Exception</b> (p. 310) | An error occurred, typically due to missing or invalid properties. |
|----------------------------------|--------------------------------------------------------------------|



See also

**MPI::Distributor** (p. 305)

**MPI::RecordProcessor** (p. 500)

**MPI::RecordStoreResources** (p. 524)

## G.119.3 Member Function Documentation

### G.119.3.1 checkpointRestore()

```
void BiometricEvaluation::MPI::RecordStoreDistributor::checkpointRestore () [protected], [virtual]
```

Restore from a checkpoint state.

Implementations of this class use a checkpoint state to move the data sequence cursor to a point past data that has been previously distributed. The **MPI** (p. 145) **Framework** (p. 117) calls this method prior to the start of distributing work packages.

Implements **BiometricEvaluation::MPI::Distributor** (p. 306).

### G.119.3.2 checkpointSave()

```
void BiometricEvaluation::MPI::RecordStoreDistributor::checkpointSave (
 const std::string & reason) [protected], [virtual]
```

Create a checkpoint state.

Implementations of this class create a checkpoint state that captures enough information to allow the implementation to move the data sequence cursor to a point past data that has been previously distributed. The **MPI** (p. 145) **Framework** (p. 117) calls this method when a premature shutdown is requested.

Parameters

|               |                                                            |
|---------------|------------------------------------------------------------|
| <i>reason</i> | A string giving the reason for the checkpoint to be saved. |
|---------------|------------------------------------------------------------|

Implements **BiometricEvaluation::MPI::Distributor** (p. 307).

### G.119.3.3 createWorkPackage()

```
void BiometricEvaluation::MPI::RecordStoreDistributor::createWorkPackage (
 MPI::WorkPackage & workPackage) [protected], [virtual]
```

Create a work package for distribution.

Implementations of this class create a work package to encapsulate the specific data type that is to be distributed.

Implements **BiometricEvaluation::MPI::Distributor** (p. 307).

## G.119.4 Member Data Documentation

### G.119.4.1 CHECKPOINTLASTKEY

```
const std::string BiometricEvaluation::MPI::RecordStoreDistributor::CHECKPOINTLASTKEY [static]
```

The last key that was distributed, "Last Key".

### G.119.4.2 CHECKPOINTNUMKEYS

```
const std::string BiometricEvaluation::MPI::RecordStoreDistributor::CHECKPOINTNUMKEYS [static]
 The number of keys that were distributed, "Num Keys".
```

## G.120 BiometricEvaluation::IO::RecordStoreIterator Class Reference

Generic ForwardIterator for all RecordStores.

```
#include <be_io_recordstore.h>
```

### Public Types

- using **iterator\_category** = std::forward\_iterator\_tag
- using **value\_type** = **RecordStore::Record**
- using **difference\_type** = std::ptrdiff\_t
- using **pointer** = **value\_type** \*
- using **reference** = **value\_type** &

### Public Member Functions

- **RecordStoreIterator** ()=default  
*Default constructor.*
- **RecordStoreIterator** ( **IO::RecordStore** \*recordStore, bool atEnd)  
*Constructor.*
- **RecordStoreIterator** (const **RecordStoreIterator** &rhs)=default
- **RecordStoreIterator** ( **RecordStoreIterator** &&rvalue)=default
- **~RecordStoreIterator** ()=default
- **reference** **operator\*** ()
- **pointer** **operator->** ()
- **RecordStoreIterator** & **operator++** ()
- **RecordStoreIterator** **operator++** (int postfix)
- **RecordStoreIterator** **operator+=** ( **difference\_type** rhs)  
*Advance a variable number of arguments.*
- **RecordStoreIterator** **operator+** ( **difference\_type** rhs)  
*Advance a variable number of arguments.*
- bool **operator==** (const **RecordStoreIterator** &rhs)  
*Equivalence operator.*
- bool **operator!=** (const **RecordStoreIterator** &rhs)  
*Non-equivalence operator.*
- **RecordStoreIterator** & **operator=** ( **RecordStoreIterator** &rhs)=default
- **RecordStoreIterator** & **operator=** ( **RecordStoreIterator** &&rhs)=default

### G.120.1 Detailed Description

Generic ForwardIterator for all RecordStores.

Note

Dereferencing an iterator returns a copy of the value. Modifying a non-const iterator does not manipulate the underlying **RecordStore** (p. 504).

This generic iterator provides no optimization over **RecordStore::sequence()** (p. 515).

G.120.2 Member Typedef Documentation

G.120.2.1 difference\_type

using BiometricEvaluation::IO::RecordStoreIterator::difference\_type = std::ptrdiff\_t  
Type used to measure distance between iterators

G.120.2.2 iterator\_category

using BiometricEvaluation::IO::RecordStoreIterator::iterator\_category = std::forward\_iterator<-tag  
Type of iterator

G.120.2.3 pointer

using BiometricEvaluation::IO::RecordStoreIterator::pointer = value\_type\*  
Pointer to the type iterated over

G.120.2.4 reference

using BiometricEvaluation::IO::RecordStoreIterator::reference = value\_type&  
Reference to the type iterated over

G.120.2.5 value\_type

using BiometricEvaluation::IO::RecordStoreIterator::value\_type = RecordStore::Record  
Type when dereferencing iterators

G.120.3 Constructor & Destructor Documentation

G.120.3.1 RecordStoreIterator() [1/4]

BiometricEvaluation::IO::RecordStoreIterator::RecordStoreIterator ( ) [default]  
Default constructor.  
Creates "end" iterator.

Note

Satisfies DefaultConstructible requirement.

G.120.3.2 RecordStoreIterator() [2/4]

BiometricEvaluation::IO::RecordStoreIterator::RecordStoreIterator (   
IO::RecordStore \* recordStore,  
bool atEnd )  
Constructor.

Parameters

|             |                                                                      |
|-------------|----------------------------------------------------------------------|
| recordStore | Pointer to a <b>RecordStore</b> (p. 504) that will be iterated over. |
| atEnd       | Whether or not to start at the "end" iterator.                       |

## Note

Iterator defaults to starting at the beginning of the **RecordStore** (p. 504).  
**RecordStoreIterator** (p. 520) does not retain any ownership of recordStore.

**G.120.3.3 RecordStoreIterator()** [3/4]

```
BiometricEvaluation::IO::RecordStoreIterator::RecordStoreIterator (
 const RecordStoreIterator & rhs) [default]
```

Default copy constructor

**G.120.3.4 RecordStoreIterator()** [4/4]

```
BiometricEvaluation::IO::RecordStoreIterator::RecordStoreIterator (
 RecordStoreIterator && rvalue) [default]
```

Default move constructor

**G.120.3.5 ~RecordStoreIterator()**

```
BiometricEvaluation::IO::RecordStoreIterator::~~RecordStoreIterator () [default]
```

Default destructor

**G.120.4 Member Function Documentation****G.120.4.1 operator"!="()**

```
bool BiometricEvaluation::IO::RecordStoreIterator::operator!= (
 const RecordStoreIterator & rhs) [inline]
```

Non-equivalence operator.

## Parameters

|            |                                                                  |
|------------|------------------------------------------------------------------|
| <i>rhs</i> | Reference to <b>RecordStoreIterator</b> (p. 520) being compared. |
|------------|------------------------------------------------------------------|

## Returns

Whether or not this is not equivalent to rhs.

## Note

Satisfies "i != j" is equivalent to "!(i == j)" condition of InputIterator.

**G.120.4.2 operator\*()**

```
reference BiometricEvaluation::IO::RecordStoreIterator::operator* ()
```

## Returns

Reference to a Record.

**G.120.4.3 operator+()**

**RecordStoreIterator** BiometricEvaluation::IO::RecordStoreIterator::operator+ (   
 **difference\_type** *rhs* )

Advance a variable number of arguments.

Parameters

|            |                                           |
|------------|-------------------------------------------|
| <i>rhs</i> | Number of objects to advance (1 or more). |
|------------|-------------------------------------------|

Returns

Self after advancing *rhs* objects.

**G.120.4.4 operator++() [1/2]**

**RecordStoreIterator**& BiometricEvaluation::IO::RecordStoreIterator::operator++ ( )

Returns

Self after advancing.

**G.120.4.5 operator++() [2/2]**

**RecordStoreIterator** BiometricEvaluation::IO::RecordStoreIterator::operator++ (   
 int *postfix* )

Returns

Copy of self before advancing.

**G.120.4.6 operator+=()**

**RecordStoreIterator** BiometricEvaluation::IO::RecordStoreIterator::operator+= (   
 **difference\_type** *rhs* )

Advance a variable number of arguments.

Parameters

|            |                                           |
|------------|-------------------------------------------|
| <i>rhs</i> | Number of objects to advance (1 or more). |
|------------|-------------------------------------------|

Returns

Self after advancing *rhs* objects.

**G.120.4.7 operator->()**

**pointer** BiometricEvaluation::IO::RecordStoreIterator::operator-> ( )

Returns

A dereferenced Record.

#### G.120.4.8 operator=()

```
RecordStoreIterator& BiometricEvaluation::IO::RecordStoreIterator::operator= (
 RecordStoreIterator && rhs) [default]
```

Default move assignment operator

#### G.120.4.9 operator==()

```
bool BiometricEvaluation::IO::RecordStoreIterator::operator== (
 const RecordStoreIterator & rhs)
```

Equivalence operator.

Parameters

|            |                                                                  |
|------------|------------------------------------------------------------------|
| <i>rhs</i> | Reference to <b>RecordStoreIterator</b> (p. 520) being compared. |
|------------|------------------------------------------------------------------|

Returns

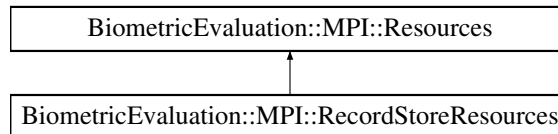
Whether or not this is equivalent to rhs.

## G.121 BiometricEvaluation::MPI::RecordStoreResources Class Reference

A class to represent a set of resources needed by an **MPI** (p. 145) program using a RecordStore for input.

```
#include <be_mpi_recordstoreresources.h>
```

Inheritance diagram for BiometricEvaluation::MPI::RecordStoreResources:



### Public Member Functions

- **RecordStoreResources** (const std::string &propertiesFileName)  
*Constructor taking the name of the properties file with the resource names.*
- uint32\_t **getChunkSize** () const
- bool **haveRecordStore** () const  
*Indicator that a record store has been opened.*
- std::shared\_ptr< **IO::RecordStore** > **getRecordStore** () const  
*Return the RecordStore named in the property set.*

## Static Public Member Functions

- static std::vector< std::string > **getRequiredProperties** ()  
*Obtain the required properties as strings.*
- static std::vector< std::string > **getOptionalProperties** ()  
*Obtain the list of optional properties.*

## Static Public Attributes

- static const std::string **INPUTRSPROPERTY**  
*The property string "Input Record Store"; required.*
- static const std::string **CHUNKSIZEPROPERTY**  
*The property string "Chunk Size"; required.*

### G.121.1 Detailed Description

A class to represent a set of resources needed by an **MPI** (p. 145) program using a RecordStore for input.

**Resources** (p. 533) are opened based on the property when appropriate. The input record store need not be accessible. Applications should call **haveRecordStore**() (p. 526) to check whether the record store has been opened.

### G.121.2 Constructor & Destructor Documentation

#### G.121.2.1 RecordStoreResources()

```
BiometricEvaluation::MPI::RecordStoreResources::RecordStoreResources (
 const std::string & propertiesFileName)
```

Constructor taking the name of the properties file with the resource names.

Exceptions

|                                           |                                       |
|-------------------------------------------|---------------------------------------|
| <b>Error::FileError</b> (p. 316)          | The resources file could not be read. |
| <b>Error::ObjectDoesNotExist</b> (p. 457) | A required property does not exist.   |
| <b>Error::Exception</b> (p. 310)          | Some other error occurred.            |

### G.121.3 Member Function Documentation

#### G.121.3.1 getOptionalProperties()

```
static std::vector<std::string> BiometricEvaluation::MPI::RecordStoreResources::getOptional←
Properties () [static]
```

Obtain the list of optional properties.

Returns

A set of optional property strings.

**G.121.3.2 getRecordStore()**

```
std::shared_ptr< IO::RecordStore> BiometricEvaluation::MPI::RecordStoreResources::getRecordStore () const
```

Return the RecordStore named in the property set.

Returns

A shared pointer to the record store.

**G.121.3.3 getRequiredProperties()**

```
static std::vector<std::string> BiometricEvaluation::MPI::RecordStoreResources::getRequiredProperties () [static]
```

Obtain the required properties as strings.

Returns

The set of required properties.

**G.121.3.4 haveRecordStore()**

```
bool BiometricEvaluation::MPI::RecordStoreResources::haveRecordStore () const
```

Indicator that a record store has been opened.

Returns

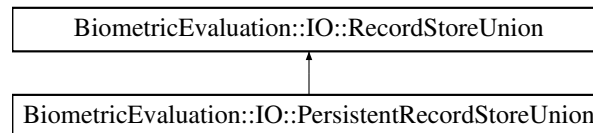
true if input record store is opened, false otherwise.

**G.122 BiometricEvaluation::IO::RecordStoreUnion Class Reference**

A collection of N related read-only RecordStores, operated on simultaneously.

```
#include <be_io_recordstoreunion.h>
```

Inheritance diagram for BiometricEvaluation::IO::RecordStoreUnion:

**Public Member Functions**

- **RecordStoreUnion** (const std::map< const std::string, const std::string > &recordStores)
- **RecordStoreUnion** (std::map< const std::string, const std::string >::iterator first, std::map< const std::string, const std::string >::iterator last)
- **RecordStoreUnion** (std::initializer\_list< std::pair< const std::string, const std::string >> recordStores)
- **RecordStoreUnion** (const std::map< const std::string, const std::shared\_ptr< **BiometricEvaluation::IO::RecordStore** >> &recordStores)



- **RecordStoreUnion** (std::map< const std::string, const std::shared\_ptr< **BiometricEvaluation::IO::RecordStore** >>::iterator first, std::map< const std::string, const std::shared\_ptr< **BiometricEvaluation::IO::RecordStore** >>::iterator last)
- **RecordStoreUnion** (std::initializer\_list< std::pair< const std::string, const std::shared\_ptr< **BiometricEvaluation::IO::RecordStore** >> > recordStores)
- std::shared\_ptr< **BiometricEvaluation::IO::RecordStore** > **getRecordStore** (const std::string &name) const  
*Obtain a pointer to an open **RecordStore** (p. 504).*
- std::vector< std::string > **getNames** () const  
*Obtain the names of RecordStores set during construction.*
- std::map< const std::string, **BiometricEvaluation::Memory::uint8Array** > **read** (const std::string &key) const  
*Read a key from all member RecordStores.*
- std::map< const std::string, uint64\_t > **length** (const std::string &key) const  
*Retrieve the length of a key from all member RecordStores.*
- **RecordStoreUnion** (const **RecordStoreUnion** &)=delete
- **RecordStoreUnion** & operator= (const **RecordStoreUnion** &)=delete
- ~**RecordStoreUnion** ()

## Protected Member Functions

- **RecordStoreUnion** ()  
*Empty constructor for children.*
- void **setImpl** (const std::shared\_ptr< RecordStoreUnion::Impl > &pimpl)  
*Change the implementation of this object.*

## G.122.1 Detailed Description

A collection of N related read-only RecordStores, operated on simultaneously.

A **RecordStoreUnion** (p. 526) object is not copyable due to the fact that most **RecordStore** (p. 504) objects are not copyable.

## G.122.2 Constructor & Destructor Documentation

### G.122.2.1 RecordStoreUnion() [1/7]

```
BiometricEvaluation::IO::RecordStoreUnion::RecordStoreUnion (
 const std::map< const std::string, const std::string > & recordStores)
RecordStoreUnion (p. 526) constructor.
```

Parameters

|                     |                                                                            |
|---------------------|----------------------------------------------------------------------------|
| <i>recordStores</i> | Map of developer-provided names to paths to a <b>RecordStore</b> (p. 504). |
|---------------------|----------------------------------------------------------------------------|

**G.122.2.2 RecordStoreUnion()** [2/7]

```
BiometricEvaluation::IO::RecordStoreUnion::RecordStoreUnion (
 std::map< const std::string, const std::string >::iterator first,
 std::map< const std::string, const std::string >::iterator last)
 RecordStoreUnion (p. 526) constructor.
```

Parameters

|              |                                                                                                       |
|--------------|-------------------------------------------------------------------------------------------------------|
| <i>first</i> | Iterator to the start of a map of developer-provided names to paths to a <b>RecordStore</b> (p. 504). |
| <i>last</i>  | Iterator to the end of a map of developer-provided names to paths to a <b>RecordStore</b> (p. 504).   |

**G.122.2.3 RecordStoreUnion()** [3/7]

```
BiometricEvaluation::IO::RecordStoreUnion::RecordStoreUnion (
 std::initializer_list< std::pair< const std::string, const std::string >> recordStores)
 RecordStoreUnion (p. 526) constructor.
```

Parameters

|                     |                                                                                      |
|---------------------|--------------------------------------------------------------------------------------|
| <i>recordStores</i> | List of pairs of developer-provided name and paths to a <b>RecordStore</b> (p. 504). |
|---------------------|--------------------------------------------------------------------------------------|

**G.122.2.4 RecordStoreUnion()** [4/7]

```
BiometricEvaluation::IO::RecordStoreUnion::RecordStoreUnion (
 const std::map< const std::string, const std::shared_ptr< BiometricEvaluation
::IO::RecordStore >> & recordStores)
 RecordStoreUnion (p. 526) constructor.
```

Parameters

|                     |                                                                               |
|---------------------|-------------------------------------------------------------------------------|
| <i>recordStores</i> | Map of developer-provided names and open <b>RecordStore</b> (p. 504) objects. |
|---------------------|-------------------------------------------------------------------------------|

Note

Behavior when providing a **RecordStore** (p. 504) that has been opened read/write is undefined.

**G.122.2.5 RecordStoreUnion()** [5/7]

```
BiometricEvaluation::IO::RecordStoreUnion::RecordStoreUnion (
 std::map< const std::string, const std::shared_ptr< BiometricEvaluation::IO::
RecordStore >>::iterator first,
 std::map< const std::string, const std::shared_ptr< BiometricEvaluation::IO::
RecordStore >>::iterator last)
 RecordStoreUnion (p. 526) constructor.
```

Parameters

|              |                                                                                                          |
|--------------|----------------------------------------------------------------------------------------------------------|
| <i>first</i> | Iterator to the start of a map of developer-provided names and open <b>RecordStore</b> (p. 504) objects. |
| <i>last</i>  | Iterator to the end of a map of developer-provided names and open <b>RecordStore</b> (p. 504) objects.   |

Note

Behavior when providing a **RecordStore** (p. 504) that has been opened read/write is undefined.

G.122.2.6 RecordStoreUnion() [6/7]

```
BiometricEvaluation::IO::RecordStoreUnion::RecordStoreUnion (
 std::initializer_list< std::pair< const std::string, const std::shared_ptr< Biometric↔
Evaluation::IO::RecordStore >>> recordStores)
 RecordStoreUnion (p. 526) constructor.
```

Parameters

|                     |                                                                                        |
|---------------------|----------------------------------------------------------------------------------------|
| <i>recordStores</i> | List of pairs of developer-provided name and open <b>RecordStore</b> (p. 504) objects. |
|---------------------|----------------------------------------------------------------------------------------|

Note

Behavior when providing a **RecordStore** (p. 504) that has been opened read/write is undefined.

G.122.2.7 ~RecordStoreUnion()

```
BiometricEvaluation::IO::RecordStoreUnion::~~RecordStoreUnion ()
 Destructor.
```

G.122.2.8 RecordStoreUnion() [7/7]

```
BiometricEvaluation::IO::RecordStoreUnion::RecordStoreUnion () [protected]
 Empty constructor for children.
```

Note

Implementation is not set. Callers must also call **setImpl()** (p. 531) to provide functionality.

setImpl

G.122.3 Member Function Documentation

G.122.3.1 getNames()

```
std::vector<std::string> BiometricEvaluation::IO::RecordStoreUnion::getNames () const
 Obtain the names of RecordStores set during construction.
```

## Returns

Vector of names of RecordStores.

**G.122.3.2 getRecordStore()**

```
std::shared_ptr< BiometricEvaluation::IO::RecordStore> BiometricEvaluation::IO::RecordStore←
Union::getRecordStore (
```

```
 const std::string & name) const
```

Obtain a pointer to an open **RecordStore** (p. 504).

## Parameters

|             |                                                                   |
|-------------|-------------------------------------------------------------------|
| <i>name</i> | Name provided to <b>RecordStore</b> (p. 504) during construction. |
|-------------|-------------------------------------------------------------------|

## Exceptions

|                           |                         |
|---------------------------|-------------------------|
| <i>ObjectDoesNotExist</i> | name is not recognized. |
|---------------------------|-------------------------|

**G.122.3.3 length()**

```
std::map<const std::string, uint64_t> BiometricEvaluation::IO::RecordStoreUnion::length (
 const std::string & key) const
```

Retrieve the length of a key from all member RecordStores.

## Parameters

|            |                  |
|------------|------------------|
| <i>key</i> | The key to read. |
|------------|------------------|

## Returns

Map of **RecordStore** (p. 504) name to data length read from said **RecordStore** (p. 504).

## Exceptions

|                                           |                                                                                                   |
|-------------------------------------------|---------------------------------------------------------------------------------------------------|
| <i>Error::ObjectDoesNotExist</i> (p. 457) | key does not exist in any member RecordStores.                                                    |
| <i>Error::StrategyError</i> (p. 567)      | Exceptions propagated from <b>RecordStore</b> (p. 504), with the exception of ObjectDoesNotExist. |

## Note

Exceptions are thrown after **length()** (p. 530) has been called on all member RecordStores.

G.122.3.4 read()

```
std::map<const std::string, BiometricEvaluation::Memory::uint8Array> BiometricEvaluation::IO::RecordStoreUnion::read (
 const std::string & key) const
 Read a key from all member RecordStores.
```

Parameters

|            |                  |
|------------|------------------|
| <i>key</i> | The key to read. |
|------------|------------------|

Returns

Map of **RecordStore** (p. 504) name to data read from said **RecordStore** (p. 504).

Exceptions

|                                           |                                                                                                   |
|-------------------------------------------|---------------------------------------------------------------------------------------------------|
| <i>Error::ObjectDoesNotExist</i> (p. 457) | key does not exist in any member RecordStores.                                                    |
| <i>Error::StrategyError</i> (p. 567)      | Exceptions propagated from <b>RecordStore</b> (p. 504), with the exception of ObjectDoesNotExist. |

Note

Exceptions are thrown after **read()** (p. 530) has been called on all member RecordStores.

G.122.3.5 setImpl()

```
void BiometricEvaluation::IO::RecordStoreUnion::setImpl (
 const std::shared_ptr< RecordStoreUnion::Impl > & pimpl) [protected]
 Change the implementation of this object.
```

Parameters

|             |                                        |
|-------------|----------------------------------------|
| <i>impl</i> | Pointer to an implementation instance. |
|-------------|----------------------------------------|

G.123 BiometricEvaluation::Image::Resolution Struct Reference

A structure to represent the resolution of an image.

```
#include <be_image.h>
```

Public Types

- enum **Units** { **Units::NA** = 0, **Units::PPI** = 1, **Units::PPMM** = 2, **Units::PPCM** = 3 }

Possible representations of the units in a **Resolution** (p. 531) struct.

## Public Member Functions

- **Resolution** (const double **xRes**=0.0, const double **yRes**=0.0, const **Units** **units**= **Units::PPI**)  
Create a **Resolution** (p. 531) struct.
- **Resolution toUnits** (const **Units** & **units**) const  
Obtain alternate representations of this resolution.

## Public Attributes

- double **xRes**
- double **yRes**
- **Units** **units**

### G.123.1 Detailed Description

A structure to represent the resolution of an image.

### G.123.2 Member Enumeration Documentation

#### G.123.2.1 Units

enum **BiometricEvaluation::Image::Resolution::Units** [strong]  
Possible representations of the units in a **Resolution** (p. 531) struct.

Enumerator

|      |                                       |
|------|---------------------------------------|
| NA   | Not-applicable: unknown, or otherwise |
| PPI  | Pixels per inch                       |
| PPMM | Pixels per millimeter                 |
| PPCM | Pixels per centimeter                 |

### G.123.3 Constructor & Destructor Documentation

#### G.123.3.1 Resolution()

```
BiometricEvaluation::Image::Resolution::Resolution (
 const double xRes = 0.0,
 const double yRes = 0.0,
 const Units units = Units::PPI)
```

Create a **Resolution** (p. 531) struct.

Parameters

|    |              |                                              |
|----|--------------|----------------------------------------------|
| in | <i>xRes</i>  | <b>Resolution</b> (p. 531) along the X-axis  |
| in | <i>yRes</i>  | <b>Resolution</b> (p. 531) along the Y-axis  |
| in | <i>units</i> | Units in which xRes and yRes are represented |

## G.123.4 Member Function Documentation

### G.123.4.1 toUnits()

**Resolution** BiometricEvaluation::Image::Resolution::toUnits (   
     const **Units** & *units* ) const  
 Obtain alternate representations of this resolution.

Parameters

|              |                                                  |
|--------------|--------------------------------------------------|
| <i>units</i> | The units to which this resolution is converted. |
|--------------|--------------------------------------------------|

Returns

This resolution, in units units.

Exceptions

|                                 |                                                                        |
|---------------------------------|------------------------------------------------------------------------|
| <i>BE::Error::StrategyError</i> | Units are not defined for either the source or destination resolution. |
|---------------------------------|------------------------------------------------------------------------|

## G.123.5 Member Data Documentation

### G.123.5.1 units

**Units** BiometricEvaluation::Image::Resolution::units  
 Units in which xRes and yRes are represented

### G.123.5.2 xRes

double BiometricEvaluation::Image::Resolution::xRes  
**Resolution** (p. 531) along the X-axis

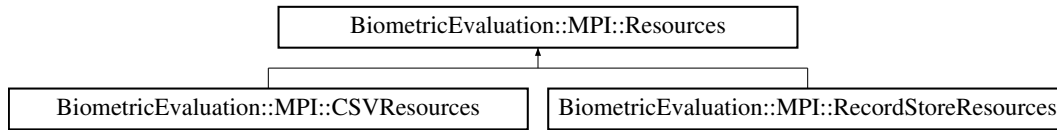
### G.123.5.3 yRes

double BiometricEvaluation::Image::Resolution::yRes  
**Resolution** (p. 531) along the Y-axis

## G.124 BiometricEvaluation::MPI::Resources Class Reference

```
#include <be_mpi_resources.h>
```

Inheritance diagram for BiometricEvaluation::MPI::Resources:



## Public Member Functions

- **Resources** (const std::string &propertiesFileName)  
*Constructor taking the name of the properties file describing the resources.*
- std::string **getPropertiesFileName** () const  
*Obtain the name of the file used to construct this object.*
- std::string **getLogsheetsURL** () const  
*Obtain the Uniform Resource Locator for the **IO** (p. 126);Logsheets object.*
- std::string **getCheckpointPath** () const  
*Obtain the Checkpoint Path name.*
- int **getRank** () const
- int **getNumTasks** () const
- int **getWorkersPerNode** () const

## Static Public Member Functions

- static std::vector< std::string > **getRequiredProperties** ()  
*Obtain the list of required properties.*
- static std::vector< std::string > **getOptionalProperties** ()  
*Obtain the list of optional properties.*

## Static Public Attributes

- static const std::string **WORKERSPERNODEPROPERTY**  
*The property string "Workers Per Node"; required.*
- static const std::string **NUMCPUS**  
*The "Workers Per Node" setting "NUMCPUS".*
- static const std::string **NUMCORES**  
*The "Workers Per Node" setting "NUMCORES".*
- static const std::string **NUMSOCKETS**  
*The "Workers Per Node" setting "NUMSOCKETS".*
- static const std::string **LOGSHEETURLPROPERTY**  
*The property string "Logsheets URL"; optional.*
- static const std::string **CHECKPOINTPATHPROPERTY**  
*The property string "Checkpoint Path"; required when checkpointing is enabled, optional otherwise.*

### G.124.1 Detailed Description

A class to represent a set of resources needed by an **MPI** (p. 145) program. The resources are based on a properties file as well as some dynamic information, such as **MPI** (p. 145) rank and process ID.



## G.124.2 Constructor & Destructor Documentation

### G.124.2.1 Resources()

```
BiometricEvaluation::MPI::Resources::Resources (
 const std::string & propertiesFileName)
```

Constructor taking the name of the properties file describing the resources.

Parameters

|    |                           |                                                 |
|----|---------------------------|-------------------------------------------------|
| in | <i>propertiesFileName</i> | The name of the file containing the Properties. |
|----|---------------------------|-------------------------------------------------|

Exceptions

|                                           |                                       |
|-------------------------------------------|---------------------------------------|
| <i>Error::FileError</i> (p. 316)          | The resources file could not be read. |
| <i>Error::ObjectDoesNotExist</i> (p. 457) | A required property does not exist.   |
| <i>Error::Exception</i> (p. 310)          | Some other error occurred.            |

## G.124.3 Member Function Documentation

### G.124.3.1 getCheckpointPath()

```
std::string BiometricEvaluation::MPI::Resources::getCheckpointPath () const
```

Obtain the Checkpoint Path name.

This string may be empty, indicating that there is no checkpoint path in the Properties file.

Returns

The Checkpoint Path.

### G.124.3.2 getLogsheetsURL()

```
std::string BiometricEvaluation::MPI::Resources::getLogsheetsURL () const
```

Obtain the Uniform Resource Locator for the **IO** (p. 126):Logsheets object.

This string may be empty, indicating that there is no Logsheets URL in the Properties file.

Returns

The Logsheets URL.

**G.124.3.3 getOptionalProperties()**

```
static std::vector<std::string> BiometricEvaluation::MPI::Resources::getOptionalProperties (
) [static]
```

Obtain the list of optional properties.

Returns

A set of optional property strings.

**G.124.3.4 getPropertiesFileName()**

```
std::string BiometricEvaluation::MPI::Resources::getPropertiesFileName () const
```

Obtain the name of the file used to construct this object.

Returns

The name of the properties file.

**G.124.3.5 getRequiredProperties()**

```
static std::vector<std::string> BiometricEvaluation::MPI::Resources::getRequiredProperties (
) [static]
```

Obtain the list of required properties.

Returns

A set of required property strings.

**G.124.4 Member Data Documentation****G.124.4.1 NUMCORES**

```
const std::string BiometricEvaluation::MPI::Resources::NUMCORES [static]
```

The "Workers Per Node" setting "NUMCORES".

This setting indicates the **MPI** (p. 145) **Framework** (p. 117) is to create one worker for each physical CPU core.

**G.124.4.2 NUMCPUS**

```
const std::string BiometricEvaluation::MPI::Resources::NUMCPUS [static]
```

The "Workers Per Node" setting "NUMCPUS".

This setting indicates the **MPI** (p. 145) **Framework** (p. 117) is to create one worker for each logical CPU.

**G.124.4.3 NUMSOCKETS**

```
const std::string BiometricEvaluation::MPI::Resources::NUMSOCKETS [static]
```

The "Workers Per Node" setting "NUMSOCKETS".

This setting indicates the **MPI** (p. 145) **Framework** (p. 117) is to create one worker for each physical CPU socket.

#### G.124.4.4 WORKERSPERNODEPROPERTY

```
const std::string BiometricEvaluation::MPI::Resources::WORKERSPERNODEPROPERTY [static]
```

The property string "Workers Per Node"; required.

This value shall be either an integer or one of the strings "NUMCPUS", "NUMCORES", "NUMSOCKETS".

## G.125 BiometricEvaluation::Framework::API< T >::Result Class Reference

```
#include <be_framework_api.h>
```

### Public Member Functions

- **Result ()**
- **bool operator! () const**  
*Logical negation operator overload.*
- **operator bool () const**  
*Boolean conversion operator.*

### Public Attributes

- **uint64\_t elapsed**
- **T status**  
*Value returned from operation.*
- **APICurrentState currentState**  
*Current state of operation.*

### G.125.1 Detailed Description

```
template<typename T>
class BiometricEvaluation::Framework::API< T >::Result
```

The result of an operation.

### G.125.2 Constructor & Destructor Documentation

#### G.125.2.1 Result()

```
template<typename T >
BiometricEvaluation::Framework::API< T >::Result::Result ()
 Constructor
```

### G.125.3 Member Function Documentation

**G.125.3.1 operator bool()**

```
template<typename T >
BiometricEvaluation::Framework::API< T >::Result::operator bool () const [inline], [explicit]
 Boolean conversion operator.
```

**Returns**

True if operation completed, false otherwise.

**G.125.3.2 operator"!()**

```
template<typename T >
bool BiometricEvaluation::Framework::API< T >::Result::operator! () const [inline]
 Logical negation operator overload.
```

**Returns**

True if operation failed to complete, false otherwise.

**G.125.4 Member Data Documentation****G.125.4.1 elapsed**

```
template<typename T >
uint64_t BiometricEvaluation::Framework::API< T >::Result::elapsed
 Time (p. 161) elapsed while calling operation.
```

**G.125.4.2 status**

```
template<typename T >
T BiometricEvaluation::Framework::API< T >::Result::status
 Value returned from operation.
```

**Note**

Only populated when currentState == **APICurrentState::Completed** (p. 118).

**G.126 BiometricEvaluation::Feature::RidgeCountItem Struct Reference**

Representation of ridge count data, which is the number of ridges between any two minutia data points, each represented by its index number.

```
#include <be_feature_minutiae.h>
```

**Public Member Functions**

- **RidgeCountItem** ( **RidgeCountExtractionMethod** extraction\_method, int index\_one, int index\_two, int count=0)

Create a *RidgeCountItem* (p. 538) struct.

## Public Attributes

- **RidgeCountExtractionMethod** `extraction_method`
- `int` **index\_one**
- `int` **index\_two**
- `int` **count**

### G.126.1 Detailed Description

Representation of ridge count data, which is the number of ridges between any two minutia data points, each represented by its index number.

## G.127 BiometricEvaluation::Image::ROI Struct Reference

A structure to represent a region of interest (**ROI** (p. 539)), which is a bounding box and a set of coordinates.

```
#include <be_image.h>
```

## Public Member Functions

- **ROI** ()
- **ROI** (const **Size** size, const uint32\_t horzOffset, const uint32\_t vertOffset, const CoordinateSet &path)

## Public Attributes

- **Size** `size`
- `uint32_t` **horzOffset**
- `uint32_t` **vertOffset**
- `CoordinateSet` **path**

### G.127.1 Detailed Description

A structure to represent a region of interest (**ROI** (p. 539)), which is a bounding box and a set of coordinates.

### G.127.2 Constructor & Destructor Documentation

#### G.127.2.1 ROI() [1/2]

```
BiometricEvaluation::Image::ROI::ROI ()
```

Create an empty **ROI** (p. 539) object.

#### G.127.2.2 ROI() [2/2]

```
BiometricEvaluation::Image::ROI::ROI (
 const Size size,
 const uint32_t horzOffset,
 const uint32_t vertOffset,
 const CoordinateSet & path)
```

Create a **ROI** (p. 539) object with the given parameters.

## Parameters

|    |                   |                                                  |
|----|-------------------|--------------------------------------------------|
| in | <i>size</i>       | The size of the region of interest.              |
| in | <i>horzOffset</i> | The horizontal offset of the region of interest. |
| in | <i>vertOffset</i> | The vertical offset of the region of interest.   |
| in | <i>path</i>       | The path offset of the region of interest.       |

## G.128 BiometricEvaluation::MPI::Runtime Class Reference

**Runtime** (p. 540) support for the startup/shutdown of **MPI** (p. 145) jobs.

```
#include <be_mpi_runtime.h>
```

### Public Member Functions

- **Runtime** (int &argc, char \*\*&argv, bool checkpointEnable=false)  
Construct the runtime environment for the processes making up the **MPI** (p. 145) job.
- void **start** ( **BiometricEvaluation::MPI::Distributor** &distributor, **BiometricEvaluation::MPI::↔Receiver** &receiver)  
Startup the runtime environment for the **MPI** (p. 145) job.
- void **shutdown** ()  
Shutdown the runtime environment for the **MPI** (p. 145) job.
- void **abort** (int errcode)  
Abort the runtime the **MPI** (p. 145) job.

### G.128.1 Detailed Description

**Runtime** (p. 540) support for the startup/shutdown of **MPI** (p. 145) jobs.

This class provides methods that are used by applications to start and shutdown the **MPI** (p. 145) job. Each job consists of a single distributor of work, and 1..n receivers of work which then distribute the work packages to child processes to take action on the work package.

### G.128.2 Constructor & Destructor Documentation

#### G.128.2.1 Runtime()

```
BiometricEvaluation::MPI::Runtime::Runtime (
 int & argc,
 char **& argv,
 bool checkpointEnable = false)
```

Construct the runtime environment for the processes making up the **MPI** (p. 145) job.

Whether to save a checkpoint on clean shutdown, and recover a checkpoint on startup, is optionally specified.

## Parameters

|    |             |                                                                   |
|----|-------------|-------------------------------------------------------------------|
| in | <i>argc</i> | The argument count, taken from the command line passed to main(). |
|----|-------------|-------------------------------------------------------------------|

## Parameters

|    |                         |                                                                                                                                                                                                                    |
|----|-------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| in | <i>argv</i>             | The argument vector, taken from the command line passed to main().                                                                                                                                                 |
| in | <i>checkpointEnable</i> | True indicates that a checkpoint should be saved on early shutdown and restored on startup, if the checkpoint data is present.. Checkpoints are implementation-defined by the <b>Distributor</b> (p. 305) classes. |

## G.128.3 Member Function Documentation

### G.128.3.1 abort()

```
void BiometricEvaluation::MPI::Runtime::abort (
 int errcode)
```

Abort the runtime the **MPI** (p. 145) job.

This method will cause the **MPI** (p. 145) job to terminate immediately. All processes will end without the opportunity to save.

## Parameters

|    |                |                                                              |
|----|----------------|--------------------------------------------------------------|
| in | <i>errcode</i> | The error code to return to the <b>MPI</b> (p. 145) runtime. |
|----|----------------|--------------------------------------------------------------|

### G.128.3.2 shutdown()

```
void BiometricEvaluation::MPI::Runtime::shutdown ()
```

Shutdown the runtime environment for the **MPI** (p. 145) job.

This method must be called in order for the **MPI** (p. 145) runtime to cleanly exit.

### G.128.3.3 start()

```
void BiometricEvaluation::MPI::Runtime::start (
 BiometricEvaluation::MPI::Distributor & distributor,
 BiometricEvaluation::MPI::Receiver & receiver)
```

Startup the runtime environment for the **MPI** (p. 145) job.

Exceptions thrown by the **Distributor** (p. 305) or Receiver are caught and logged.

## Parameters

|    |                    |                                                                                                        |
|----|--------------------|--------------------------------------------------------------------------------------------------------|
| in | <i>distributor</i> | The <b>Distributor</b> (p. 305) object that will form the basis of the first <b>MPI</b> (p. 145) task. |
| in | <i>receiver</i>    | The <b>Receiver</b> (p. 498) object which will form the basis of <b>MPI</b> (p. 145) tasks 1..n.       |

## G.129 BiometricEvaluation::Process::Semaphore Class Reference

Represent a semaphore that can be used for interprocess communication.

```
#include <be_process_semaphore.h>
```

## Public Member Functions

- **Semaphore** (const std::string &name, const mode\_t mode, const int value, const bool force=false)  
*Create a new named semaphore.*
- **Semaphore** (const std::string &name)  
*Open an existing named semaphore.*
- bool **wait** (const bool interruptible)  
*Wait indefinitely for the semaphore to unblock.*
- bool **trywait** (const bool interruptible)  
*Attempt to obtain the semaphore without blocking.*
- bool **timedwait** (const uint64\_t interval, const bool interruptible)  
*Attempt to obtain the semaphore while blocking for at most the specified time interval.*
- void **post** ()  
*Post (increment) to the semaphore.*
- std::string **getName** ()  
*Obtain the name of the **Semaphore** (p. 541).*

### G.129.1 Detailed Description

Represent a semaphore that can be used for interprocess communication.

Semaphores are shared counters with mutually exclusive modification properties. A counter value greater than zero means that a resource represented by the semaphore is available. A typical use is to grant exclusive access to a resource by allowing the counter to be valued at zero or one; this is known as a binary semaphore.

Note

The counter value is not exposed to clients of the object.

Because a **Semaphore** (p. 541) object wraps a system resource, the **Semaphore** (p. 541) can be passed to other functions, or inherited across a fork boundary.

### G.129.2 Constructor & Destructor Documentation

#### G.129.2.1 Semaphore() [1/2]

```
BiometricEvaluation::Process::Semaphore::Semaphore (
 const std::string & name,
 const mode_t mode,
 const int value,
 const bool force = false)
```

Create a new named semaphore.

Parameters

|    |              |                                                                                                                                                                                                                                                           |
|----|--------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| in | <i>name</i>  | The name of the semaphore, which must obey the syntax documented for the sem_open(2) call. If the semaphore already exists in the name space, construction will fail unless the force flag is true. In that case, the existing semaphore will be removed. |
| in | <i>mode</i>  | The permission mode of the semaphore.                                                                                                                                                                                                                     |
| in | <i>value</i> | The initial value of the semaphore.                                                                                                                                                                                                                       |
| in | <i>force</i> | The semaphore is created, disassociating an existing semaphore of the same name.                                                                                                                                                                          |



## Exceptions

|                                                       |                                                   |
|-------------------------------------------------------|---------------------------------------------------|
| <b>Error::ObjectExists</b> (p. <a href="#">458</a> )  | The semaphore already exists with the given name. |
| <b>Error::StrategyError</b> (p. <a href="#">567</a> ) | An error occurred when creating the semaphore.    |

**G.129.2.2 Semaphore()** [2/2]

```
BiometricEvaluation::Process::Semaphore::Semaphore (
 const std::string & name)
```

Open an existing named semaphore.

## Parameters

|    |      |                                                                                            |
|----|------|--------------------------------------------------------------------------------------------|
| in | name | The name of the semaphore, which must obey the syntax documented for the sem_open(2) call. |
|----|------|--------------------------------------------------------------------------------------------|

## Exceptions

|                                                            |                                                 |
|------------------------------------------------------------|-------------------------------------------------|
| <b>Error::ObjectDoesNotExist</b> (p. <a href="#">457</a> ) | A semaphore does not exist with the given name. |
| <b>Error::StrategyError</b> (p. <a href="#">567</a> )      | An error occurred when creating the semaphore.  |

**G.129.3 Member Function Documentation****G.129.3.1 getName()**

```
std::string BiometricEvaluation::Process::Semaphore::getName ()
```

Obtain the name of the **Semaphore** (p. [541](#)).

## Returns

The name of the Semaphore.

**G.129.3.2 post()**

```
void BiometricEvaluation::Process::Semaphore::post ()
```

Post (increment) to the semaphore.

## Exceptions

|                                                            |                                                                        |
|------------------------------------------------------------|------------------------------------------------------------------------|
| <b>Error::ObjectDoesNotExist</b> (p. <a href="#">457</a> ) | The semaphore is no longer valid.                                      |
| <b>Error::StrategyError</b> (p. <a href="#">567</a> )      | <b>System</b> (p. <a href="#">152</a> ) error obtaining the semaphore. |

**G.129.3.3   timedwait()**

```
bool BiometricEvaluation::Process::Semaphore::timedwait (
 const uint64_t interval,
 const bool interruptible)
```

Attempt to obtain the semaphore while blocking for at most the specified time interval.

## Parameters

|    |                      |                                                                                 |
|----|----------------------|---------------------------------------------------------------------------------|
| in | <i>interval</i>      | The max time to wait, in microseconds.                                          |
| in | <i>interruptible</i> | true if the function should return if waiting was interrupted, false otherwise. |

## Returns

true if the semaphore was obtained; false if not.

## Exceptions

|                                                                   |                                                                                                                                                                 |
|-------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b><i>Error::ObjectDoesNotExist</i></b> (p. <a href="#">457</a> ) | The semaphore is no longer valid.                                                                                                                               |
| <b><i>Error::NotImplemented</i></b> (p. <a href="#">456</a> )     | Function is not implemented on the system. Applications should then call <b>wait()</b> (p. <a href="#">544</a> ) or <b>trywait()</b> (p. <a href="#">544</a> ). |
| <b><i>Error::StrategyError</i></b> (p. <a href="#">567</a> )      | <b>System</b> (p. <a href="#">152</a> ) error obtaining the semaphore.                                                                                          |

**G.129.3.4   trywait()**

```
bool BiometricEvaluation::Process::Semaphore::trywait (
 const bool interruptible)
```

Attempt to obtain the semaphore without blocking.

## Parameters

|    |                      |                                                                                 |
|----|----------------------|---------------------------------------------------------------------------------|
| in | <i>interruptible</i> | true if the function should return if waiting was interrupted, false otherwise. |
|----|----------------------|---------------------------------------------------------------------------------|

## Returns

true if the semaphore was obtained; false if not.

## Exceptions

|                                                                   |                                                                        |
|-------------------------------------------------------------------|------------------------------------------------------------------------|
| <b><i>Error::ObjectDoesNotExist</i></b> (p. <a href="#">457</a> ) | The semaphore is no longer valid.                                      |
| <b><i>Error::StrategyError</i></b> (p. <a href="#">567</a> )      | <b>System</b> (p. <a href="#">152</a> ) error obtaining the semaphore. |

G.129.3.5 wait()

```
bool BiometricEvaluation::Process::Semaphore::wait (
 const bool interruptible)
```

Wait indefinitely for the semaphore to unblock.

Parameters

|    |               |                                                                                 |
|----|---------------|---------------------------------------------------------------------------------|
| in | interruptible | true if the function should return if waiting was interrupted, false otherwise. |
|----|---------------|---------------------------------------------------------------------------------|

Returns

true if the semaphore was obtained; false if not.

Exceptions

|                                    |                                                |
|------------------------------------|------------------------------------------------|
| Error::ObjectDoesNotExist (p. 457) | The semaphore is no longer valid.              |
| Error::StrategyError (p. 567)      | System (p. 152) error obtaining the semaphore. |

G.130 BiometricEvaluation::Error::SignalManager Class Reference

A **SignalManager** (p. 545) object is used to handle signals that come from the operating system.

```
#include <be_error_signal_manager.h>
```

Public Member Functions

- **SignalManager** ()
- **SignalManager** (const sigset\_t signalSet)
- void **setSignalSet** (const sigset\_t signalSet)
- void **clearSignalSet** ()
- void **setDefaultSignalSet** ()
- bool **sigHandled** ()
- void **start** ()
- void **stop** ()
- void **setSigHandled** ()
- void **clearSigHandled** ()

Static Public Attributes

- static bool **\_canSigJump**
- static sigjmp\_buf **\_sigJumpBuf**

G.130.1 Detailed Description

A **SignalManager** (p. 545) object is used to handle signals that come from the operating system.

Applications typically do not invoke most methods of a **SignalManager** (p. 545), except the **setSignalSet** (p. 547), **setDefaultSignalSet** (p. 547), and **sigHandled** (p. 547). An application wishing to just catch

memory errors can simply construct a **SignalManager** (p. 545) object, and invoke **sigHandled()** (p. 547) at the end of the signal block to detect whether a signal was handled.

The **BEGIN\_SIGNAL\_BLOCK** macro sets up the jump block and tells the **SignalManager** (p. 545) object to start handling signals. Applications can call either **setSignalSet()** (p. 547) or **setDefaultSignalSet()** (p. 547) before invoking these macros to indicate which signals are to be handled.

The **END\_SIGNAL\_BLOCK()** macro clears the signal set, so from that point forward application code signals will be handled in the system's default manner until another signal block is created.

The **ABORT\_SIGNAL\_MANAGER()** macro also disables the watchdog timer but does not create the code point destination for the jump point. This macro should be used to disable a **SignalManager** (p. 545) object when the application is no longer interested in the signal handling.

#### Attention

The **BEGIN\_SIGNAL\_BLOCK()** macro must be paired with either the **END\_SIGNAL\_BLOCK()** macro or **ABORT\_SIGNAL\_MANAGER()** macro. Failure to do so may result in undefined behavior as an active **SignalManager** (p. 545) may be invoked, forcing a jump into an incompletely initialized function.

A **SignalManager** (p. 545) is passive (i.e. no signal handlers are installed) until that **start()** (p. 548) method is called, and becomes passive when **stop()** (p. 548) is invoked. The signals that are to be handled by the object are maintained as state, and the set of signals can be changed at any time, but are not in effect until **start()** (p. 548) is called.

#### Attention

The **start()** (p. 548), **stop()** (p. 548), **setSigHandled()** (p. 547) and **clearSigHandled()** (p. 547) methods are not meant to be used directly by applications, which should use the **BEGIN\_SIGNAL\_BLOCK()/END\_SIGNAL\_BLOCK()** macro pair.

## G.130.2 Constructor & Destructor Documentation

### G.130.2.1 SignalManager() [1/2]

```
BiometricEvaluation::Error::SignalManager::SignalManager ()
```

Construct a new **SignalManager** (p. 545) object with the default signal handling: SIGSEGV and SIGBUS.

#### Exceptions

|                                      |                                        |
|--------------------------------------|----------------------------------------|
| <b>Error::StrategyError</b> (p. 567) | Could not register the signal handler. |
|--------------------------------------|----------------------------------------|

### G.130.2.2 SignalManager() [2/2]

```
BiometricEvaluation::Error::SignalManager::SignalManager (
 const sigset_t signalSet)
```

Construct a new **SignalManager** (p. 545) object with the specified signal handling, no defaults.

#### Parameters

|                  |                                                                                                |
|------------------|------------------------------------------------------------------------------------------------|
| <i>signalSet</i> | (in) The signal set; see <b>sigaction(2)</b> , <b>sigemptyset(3)</b> and <b>sigaddset(3)</b> . |
|------------------|------------------------------------------------------------------------------------------------|

Exceptions

|                                       |                                                                       |
|---------------------------------------|-----------------------------------------------------------------------|
| <b>Error::ParameterError</b> (p. 474) | One of the signals in signalSet cannot be handled (SIGKILL, SIGSTOP). |
|---------------------------------------|-----------------------------------------------------------------------|

G.130.3 Member Function Documentation

G.130.3.1 clearSigHandled()

void BiometricEvaluation::Error::SignalManager::clearSigHandled ( )  
Clear the indication that a signal was handled.

G.130.3.2 clearSignalSet()

void BiometricEvaluation::Error::SignalManager::clearSignalSet ( )  
Clear all signal handling.

G.130.3.3 setDefaultSignalSet()

void BiometricEvaluation::Error::SignalManager::setDefaultSignalSet ( )  
Set the default signals this object will manage: SIGSEGV and SIGBUS.

G.130.3.4 setSigHandled()

void BiometricEvaluation::Error::SignalManager::setSigHandled ( )  
Set a flag to indicate a signal was handled.

G.130.3.5 setSignalSet()

void BiometricEvaluation::Error::SignalManager::setSignalSet (   
const sigset\_t signalSet )  
Set the signals this object will manage.

Parameters

|                  |                                                                         |
|------------------|-------------------------------------------------------------------------|
| <i>signalSet</i> | (in) The signal set; see sigaction(2), sigemptyset(3) and sigaddset(3). |
|------------------|-------------------------------------------------------------------------|

Exceptions

|                                       |                                                                       |
|---------------------------------------|-----------------------------------------------------------------------|
| <b>Error::ParameterError</b> (p. 474) | One of the signals in signalSet cannot be handled (SIGKILL, SIGSTOP). |
|---------------------------------------|-----------------------------------------------------------------------|

G.130.3.6 sigHandled()

bool BiometricEvaluation::Error::SignalManager::sigHandled ( )  
Indicate whether a signal was handled.

## Returns

true if a signal was handled, false otherwise.

**G.130.3.7 start()**

```
void BiometricEvaluation::Error::SignalManager::start ()
```

Start handling signals of the current signal set.

## Exceptions

|                                                              |                                        |
|--------------------------------------------------------------|----------------------------------------|
| <i><b>Error::StrategyError</b></i> (p. <a href="#">567</a> ) | Could not register the signal handler. |
|--------------------------------------------------------------|----------------------------------------|

## Note

If an application invokes **start()** (p. [548](#)) without setting up a signal jump block, behavior is undefined, and can result in an infinite loop if further processing causes a signal to be raised.

**G.130.3.8 stop()**

```
void BiometricEvaluation::Error::SignalManager::stop ()
```

Stop handling signals of the current signal set.

## Exceptions

|                                                              |                                        |
|--------------------------------------------------------------|----------------------------------------|
| <i><b>Error::StrategyError</b></i> (p. <a href="#">567</a> ) | Could not register the signal handler. |
|--------------------------------------------------------------|----------------------------------------|

**G.130.4 Member Data Documentation****G.130.4.1 \_canSigJump**

```
bool BiometricEvaluation::Error::SignalManager::_canSigJump [static]
```

Flag indicating can jump after handling a signal.

## Note

Should not be directly used by applications.

**G.130.4.2 \_sigJumpBuf**

```
sigjmp_buf BiometricEvaluation::Error::SignalManager::_sigJumpBuf [static]
```

The jump buffer used by the signal handler.

## Note

Should not be directly used by applications.

## G.131 BiometricEvaluation::Image::Size Struct Reference

A structure to represent the size of an image, in pixels.

```
#include <be_image.h>
```

### Public Member Functions

- **Size** (const uint32\_t xSize=0, const uint32\_t ySize=0)  
*Create a **Size** (p. 549) struct.*

### Public Attributes

- uint32\_t xSize
- uint32\_t ySize

### G.131.1 Detailed Description

A structure to represent the size of an image, in pixels.

### G.131.2 Constructor & Destructor Documentation

#### G.131.2.1 Size()

```
BiometricEvaluation::Image::Size::Size (
 const uint32_t xSize = 0,
 const uint32_t ySize = 0)
Create a Size (p. 549) struct.
```

Parameters

|    |       |                                |
|----|-------|--------------------------------|
| in | xSize | Number of pixels on the X-axis |
| in | ySize | Number of pixels on the Y-axis |

### G.131.3 Member Data Documentation

#### G.131.3.1 xSize

```
uint32_t BiometricEvaluation::Image::Size::xSize
Number of pixels on the X-axis
```

#### G.131.3.2 ySize

```
uint32_t BiometricEvaluation::Image::Size::ySize
Number of pixels on the Y-axis
```

## G.132 BiometricEvaluation::Device::Smartcard Class Reference

```
#include <be_device_smartcard.h>
```

### Classes

- class **APDU**
- struct **APDUException**  
*Exception thrown when a command fails.*
- struct **APDUResponse**  
*The data and status words returned by the card in response to a command.*

### Public Member Functions

- **Smartcard** (unsigned int cardNum)  
*Connect to the Nth card in the system independent of any application installed on the card.*
- **Smartcard** (unsigned int cardNum, const **Memory::uint8Array** &appID)  
*Connect to the Nth card in the system and activate the application with the given identifier.*
- **Memory::uint8Array** **getDedicatedFileObject** (const **Memory::uint8Array** &objectID)
- **APDUResponse** **sendAPDU** ( **Device::Smartcard::APDU** &apdu)  
*Send an APDU (p. 216) to a card using the best transmission method available for the card.*
- **Memory::uint8Array** **getLastAPDU** () const
- **Memory::uint8Array** **getLastResponseData** () const
- std::string **getReaderID** () const  
*Obtain the identifier of the reader that the smartcard is plugged into.*
- void **setDryrun** (bool state)
- ~**Smartcard** ()
- **Smartcard** ( **Smartcard** &&other) noexcept  
*Move constructor.*
- **Smartcard** & **operator=** ( **Smartcard** &&other) noexcept  
*Move assignment.*

### G.132.1 Detailed Description

Representation of a single ISO 7816 smartcard in the system. A card can be associated with an application that is present on the card. Smartcards are accessed with a command/response protocol, and this class provides the capability to retrieve the response status and data whether the command succeeds or fails.

### G.132.2 Constructor & Destructor Documentation

#### G.132.2.1 Smartcard() [1/3]

```
BiometricEvaluation::Device::Smartcard::Smartcard (
 unsigned int cardNum)
```

Connect to the Nth card in the system independent of any application installed on the card.

Cards are numbered according to reader sequencing. Therefore, the first card (number 0) is expected to be in the first reader.



## Parameters

|    |                |                                      |
|----|----------------|--------------------------------------|
| in | <i>cardNum</i> | The number of the card to attach to. |
|----|----------------|--------------------------------------|

## Exceptions

|                                       |                                               |
|---------------------------------------|-----------------------------------------------|
| <b>Error::ParameterError</b> (p. 474) | No card exists for the given card number.     |
| <b>Error::StrategyError</b> (p. 567)  | Failed to access at least one of the readers. |

**G.132.2.2 Smartcard()** [2/3]

```
BiometricEvaluation::Device::Smartcard::Smartcard (
 unsigned int cardNum,
 const Memory::uint8Array & appID)
```

Connect to the Nth card in the system and activate the application with the given identifier.

Cards are numbered according to reader sequencing. Therefore, the first card (number 0) is expected to be in the first reader. The response data from application activation can be retrieved with the **getLastResponseData()** (p. 552) method.

## Parameters

|    |                |                                                    |
|----|----------------|----------------------------------------------------|
| in | <i>cardNum</i> | The number of the card to attach to.               |
| in | <i>appID</i>   | The ID of the application to activate on the card. |

## Exceptions

|                                       |                                                                                                                                                |
|---------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>APDUException</b> (p. 219)         | An error occurred activating the application. The status word fields on the exception's response object should be read to determine the error. |
| <b>Error::ParameterError</b> (p. 474) | No card exists for the given card number with the given application ID.                                                                        |
| <b>Error::StrategyError</b> (p. 567)  | Failed to access at least one of the readers.                                                                                                  |

**G.132.2.3 ~Smartcard()**

```
BiometricEvaluation::Device::Smartcard::~~Smartcard ()
```

Destructor.

**G.132.2.4 Smartcard()** [3/3]

```
BiometricEvaluation::Device::Smartcard::Smartcard (
 Smartcard && other) [noexcept]
```

Move constructor.

**Smartcard** (p. 550) objects are movable, maintaining the single instance of the access to the physical card. This allows the object to be placed in an STL container.

## G.132.3 Member Function Documentation

### G.132.3.1 getDedicatedFileObject()

**Memory::uint8Array** BiometricEvaluation::Device::Smartcard::getDedicatedFileObject (   
 const **Memory::uint8Array** & *objectID* )

Read a data object from the application dedicated file.

The objectID parameter must be a **TLV** (p. 581) octet string with the tag set to one of these values:

- 0x5C - A tag list data object.
- 0x5D - A header list data object.
- 0x4D - An extended header list data object.

#### Parameters

|    |                 |                                 |
|----|-----------------|---------------------------------|
| in | <i>objectID</i> | The ID of the requested object. |
|----|-----------------|---------------------------------|

#### Returns

The dedicated file object.

#### Exceptions

|                                       |                                                                                                                                                                                                                       |
|---------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>APDUException</i> (p. 219)         | An error occurred activating the application. The status word fields on the exception's response object should be read to determine the error. The data field of the response may contain partial data from the card. |
| <i>Error::StrategyError</i> (p. 567)  | An error occurred when communicating with the card.                                                                                                                                                                   |
| <i>Error::ParameterError</i> (p. 474) | The object ID is too large.                                                                                                                                                                                           |

### G.132.3.2 getLastAPDU()

**Memory::uint8Array** BiometricEvaluation::Device::Smartcard::getLastAPDU ( ) const

Obtain a copy of the last **APDU** (p. 216) sent to the card.

#### Returns

The last sent **APDU** (p. 216) as an array of octets.

### G.132.3.3 getLastResponseData()

**Memory::uint8Array** BiometricEvaluation::Device::Smartcard::getLastResponseData ( ) const

Obtain a copy of the last response data returned from the card.

Returns

The last response data as an array of octets. May be empty.

G.132.3.4 `getReaderID()`

```
std::string BiometricEvaluation::Device::Smartcard::getReaderID () const
```

Obtain the identifier of the reader that the smartcard is plugged into.

Returns

The string identifier of the reader.

G.132.3.5 `operator=()`

```
Smartcard& BiometricEvaluation::Device::Smartcard::operator= (
 Smartcard && other) [noexcept]
```

Move assignment.

**Smartcard** (p. 550) objects are movable, maintaining the single instance of the access to the physical card. This allows the object to be placed in an STL container.

G.132.3.6 `sendAPDU()`

```
APDUResponse BiometricEvaluation::Device::Smartcard::sendAPDU (
 Device::Smartcard::APDU & apdu)
```

Send an **APDU** (p. 216) to a card using the best transmission method available for the card.

Parameters

|                |             |                                                                                                                |
|----------------|-------------|----------------------------------------------------------------------------------------------------------------|
| <i>in, out</i> | <i>apdu</i> | The <b>APDU</b> (p. 216) to be sent. Fields may be modified by the function, specifically the length field(s). |
|----------------|-------------|----------------------------------------------------------------------------------------------------------------|

Exceptions

|                                      |                                                                                                                                                                                                                                                                  |
|--------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>APDUException</i> (p. 219)        | The status words from the command response are something other than 0x9000. The status word fields on the exception's response object should read to determine the result of the command. The data field of the response may contain partial data from the card. |
| <i>Error::StrategyError</i> (p. 567) | An error occurred when communicating with the card.                                                                                                                                                                                                              |

G.132.3.7 `setDryrun()`

```
void BiometricEvaluation::Device::Smartcard::setDryrun (
 bool state)
```

Set the 'dryrun' state.

## Parameters

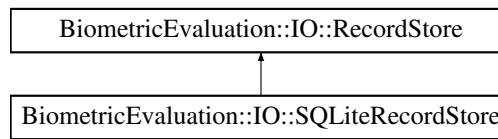
|    |       |                                                                                                               |
|----|-------|---------------------------------------------------------------------------------------------------------------|
| in | state | True when the <b>APDU</b> (p. 216) should be created, but not sent to the card. <b>getLastAPDU()</b> (p. 552) |
|----|-------|---------------------------------------------------------------------------------------------------------------|

## G.133 BiometricEvaluation::IO::SQLiteRecordStore Class Reference

A **RecordStore** (p. 504) implementation using a SQLite database as the underlying record storage system.

```
#include <be_io_sqliterecstore.h>
```

Inheritance diagram for BiometricEvaluation::IO::SQLiteRecordStore:



### Public Member Functions

- **SQLiteRecordStore** (const std::string &pathname, const std::string &description)
- **SQLiteRecordStore** (const std::string &pathname, **IO::Mode** mode= **Mode::ReadOnly**)
- void **move** (const std::string &pathname) override
 

*Move the **RecordStore** (p. 504).*
- void **sync** () const override
- unsigned int **getCount** () const override
- std::string **getPathname** () const override
- std::string **getDescription** () const override
- void **changeDescription** (const std::string &description) override
- uint64\_t **getSpaceUsed** () const override
 

*Obtain real storage utilization.*
- void **insert** (const std::string &key, const void \*const data, const uint64\_t size) override
- void **remove** (const std::string &key) override
- **Memory::uint8Array read** (const std::string &key) const override
 

*Read a complete record from a store.*
- uint64\_t **length** (const std::string &key) const override
- void **flush** (const std::string &key) const override
- **RecordStore::Record sequence** (int cursor= **BE\_RECSTORE\_SEQ\_NEXT**) override
 

*Sequence through a **RecordStore** (p. 504), returning the key/data pairs.*
- std::string **sequenceKey** (int cursor= **BE\_RECSTORE\_SEQ\_NEXT**) override
 

*Sequence through a **RecordStore** (p. 504), returning the key.*
- void **setCursorAtKey** (const std::string &key) override
- **SQLiteRecordStore** (const **SQLiteRecordStore** &)=delete
- **SQLiteRecordStore & operator=** (const **SQLiteRecordStore** &)=delete

Additional Inherited Members

G.133.1 Detailed Description

A **RecordStore** (p. 504) implementation using a SQLite database as the underlying record storage system.

G.133.2 Member Function Documentation

G.133.2.1 changeDescription()

void BiometricEvaluation::IO::SQLiteRecordStore::changeDescription (   
const std::string & *description* ) [override], [virtual]  
Change the description of the **RecordStore** (p. 504).

Parameters

|    |                    |                      |
|----|--------------------|----------------------|
| in | <i>description</i> | The new description. |
|----|--------------------|----------------------|

Exceptions

|                                      |                                                             |
|--------------------------------------|-------------------------------------------------------------|
| <b>Error::StrategyError</b> (p. 567) | An error occurred when using the underlying storage system. |
|--------------------------------------|-------------------------------------------------------------|

Implements **BiometricEvaluation::IO::RecordStore** (p. 506).

G.133.2.2 flush()

void BiometricEvaluation::IO::SQLiteRecordStore::flush (   
const std::string & *key* ) const [override], [virtual]  
Commit the record's data to storage.

Parameters

|    |            |                                      |
|----|------------|--------------------------------------|
| in | <i>key</i> | The key of the record to be flushed. |
|----|------------|--------------------------------------|

Exceptions

|                                           |                                                             |
|-------------------------------------------|-------------------------------------------------------------|
| <b>Error::ObjectDoesNotExist</b> (p. 457) | A record for the key does not exist.                        |
| <b>Error::StrategyError</b> (p. 567)      | An error occurred when using the underlying storage system. |

Implements **BiometricEvaluation::IO::RecordStore** (p. 508).

G.133.2.3 getCount()

unsigned int BiometricEvaluation::IO::SQLiteRecordStore::getCount ( ) const [override], [virtual]  
Obtain the number of items in the **RecordStore** (p. 504).

Returns

The number of items in the **RecordStore** (p. 504).

Implements **BiometricEvaluation::IO::RecordStore** (p. 508).

#### G.133.2.4 getDescription()

```
std::string BiometricEvaluation::IO::SQLiteRecordStore::getDescription () const [override], [virtual]
```

Obtain a textual description of the **RecordStore** (p. 504).

Returns

The **RecordStore** (p. 504)'s description.

Implements **BiometricEvaluation::IO::RecordStore** (p. 508).

#### G.133.2.5 getPathname()

```
std::string BiometricEvaluation::IO::SQLiteRecordStore::getPathname () const [override], [virtual]
```

Return the path name of the **RecordStore** (p. 504).

Returns

Where in the file system the **RecordStore** (p. 504) is located.

Implements **BiometricEvaluation::IO::RecordStore** (p. 509).

#### G.133.2.6 getSpaceUsed()

```
uint64_t BiometricEvaluation::IO::SQLiteRecordStore::getSpaceUsed () const [override], [virtual]
```

Obtain real storage utilization.

The amount of disk space used, for example. This is the actual space allocated by the underlying storage mechanism, in bytes.

Returns

The amount of backing storage used by the **RecordStore** (p. 504).

Exceptions

|                                      |                                                             |
|--------------------------------------|-------------------------------------------------------------|
| <b>Error::StrategyError</b> (p. 567) | An error occurred when using the underlying storage system. |
|--------------------------------------|-------------------------------------------------------------|

Implements **BiometricEvaluation::IO::RecordStore** (p. 509).

#### G.133.2.7 insert()

```
void BiometricEvaluation::IO::SQLiteRecordStore::insert (
 const std::string & key,
```

```
const void *const data,
const uint64_t size) [override], [virtual]
```

Insert a record into the store.

Parameters

|    |             |                                       |
|----|-------------|---------------------------------------|
| in | <i>key</i>  | The key of the record to be inserted. |
| in | <i>data</i> | The data for the record.              |
| in | <i>size</i> | The size of the record, in bytes.     |

Exceptions

|                                                       |                                                                                                                                      |
|-------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------|
| <b>Error::ObjectExists</b> (p. <a href="#">458</a> )  | A record with the given key is already present.                                                                                      |
| <b>Error::StrategyError</b> (p. <a href="#">567</a> ) | The <b>RecordStore</b> (p. <a href="#">504</a> ) is opened read-only, or an error occurred when using the underlying storage system. |

Implements **BiometricEvaluation::IO::RecordStore** (p. [510](#)).

### G.133.2.8 length()

```
uint64_t BiometricEvaluation::IO::SQLiteRecordStore::length (
const std::string & key) const [override], [virtual]
```

Return the length of a record.

Parameters

|    |            |                        |
|----|------------|------------------------|
| in | <i>key</i> | The key of the record. |
|----|------------|------------------------|

Returns

The record length.

Exceptions

|                                                            |                                                             |
|------------------------------------------------------------|-------------------------------------------------------------|
| <b>Error::ObjectDoesNotExist</b> (p. <a href="#">457</a> ) | A record for the key does not exist.                        |
| <b>Error::StrategyError</b> (p. <a href="#">567</a> )      | An error occurred when using the underlying storage system. |

Implements **BiometricEvaluation::IO::RecordStore** (p. [510](#)).

### G.133.2.9 move()

```
void BiometricEvaluation::IO::SQLiteRecordStore::move (
const std::string & pathname) [override], [virtual]
```

Move the **RecordStore** (p. [504](#)).

The **RecordStore** (p. [504](#)) can be moved to a new path in the file system.

## Parameters

|    |                 |                                                  |
|----|-----------------|--------------------------------------------------|
| in | <i>pathname</i> | The new path of the <b>RecordStore</b> (p. 504). |
|----|-----------------|--------------------------------------------------|

## Exceptions

|                                      |                                                             |
|--------------------------------------|-------------------------------------------------------------|
| <b>Error::StrategyError</b> (p. 567) | An error occurred when using the underlying storage system. |
|--------------------------------------|-------------------------------------------------------------|

Implements **BiometricEvaluation::IO::RecordStore** (p. 511).

**G.133.2.10 read()**

```
Memory::uint8Array BiometricEvaluation::IO::SQLiteRecordStore::read (
 const std::string & key) const [override], [virtual]
```

Read a complete record from a store.

The AutoArray will be resized to match the size of the data.

## Parameters

|    |            |                                   |
|----|------------|-----------------------------------|
| in | <i>key</i> | The key of the record to be read. |
|----|------------|-----------------------------------|

## Returns

The record associated with the key.

## Exceptions

|                                           |                                                             |
|-------------------------------------------|-------------------------------------------------------------|
| <b>Error::ObjectDoesNotExist</b> (p. 457) | A record for the key does not exist.                        |
| <b>Error::StrategyError</b> (p. 567)      | An error occurred when using the underlying storage system. |

Implements **BiometricEvaluation::IO::RecordStore** (p. 512).

**G.133.2.11 remove()**

```
void BiometricEvaluation::IO::SQLiteRecordStore::remove (
 const std::string & key) [override], [virtual]
```

Remove a record from the store.

## Parameters

|    |            |                                      |
|----|------------|--------------------------------------|
| in | <i>key</i> | The key of the record to be removed. |
|----|------------|--------------------------------------|

## Exceptions

|                                           |                                      |
|-------------------------------------------|--------------------------------------|
| <b>Error::ObjectDoesNotExist</b> (p. 457) | A record for the key does not exist. |
|-------------------------------------------|--------------------------------------|



## Exceptions

|                                      |                                                             |
|--------------------------------------|-------------------------------------------------------------|
| <b>Error::StrategyError</b> (p. 567) | An error occurred when using the underlying storage system. |
|--------------------------------------|-------------------------------------------------------------|

Implements **BiometricEvaluation::IO::RecordStore** (p. 513).

## G.133.2.12 sequence()

```
RecordStore::Record BiometricEvaluation::IO::SQLiteRecordStore::sequence (
 int cursor = BE_RECSTORE_SEQ_NEXT) [override], [virtual]
```

Sequence through a **RecordStore** (p. 504), returning the key/data pairs.

Sequencing means to start at some point in the store and return the record, then repeatedly calling the function to return the next record. The starting point is typically the first record, and is set to that when the **RecordStore** (p. 504) object is created. The starting point can be reset by calling this method with the cursor parameter set to **BE\_RECSTORE\_SEQ\_START**.

## Parameters

|    |        |                                                                  |
|----|--------|------------------------------------------------------------------|
| in | cursor | The location within the sequence of the key/data pair to return. |
|----|--------|------------------------------------------------------------------|

## Returns

The record that is currently in sequence.

## Exceptions

|                                           |                                                             |
|-------------------------------------------|-------------------------------------------------------------|
| <b>Error::ObjectDoesNotExist</b> (p. 457) | End of sequencing.                                          |
| <b>Error::StrategyError</b> (p. 567)      | An error occurred when using the underlying storage system. |

Implements **BiometricEvaluation::IO::RecordStore** (p. 515).

## G.133.2.13 sequenceKey()

```
std::string BiometricEvaluation::IO::SQLiteRecordStore::sequenceKey (
 int cursor = BE_RECSTORE_SEQ_NEXT) [override], [virtual]
```

Sequence through a **RecordStore** (p. 504), returning the key.

Sequencing means to start at some point in the store and return the key, then repeatedly calling the function to return the next key. The starting point is typically the first record, and is set to that when the **RecordStore** (p. 504) object is created. The starting point can be reset by calling this method with the cursor parameter set to **BE\_RECSTORE\_SEQ\_START**.

## Parameters

|    |        |                                                                  |
|----|--------|------------------------------------------------------------------|
| in | cursor | The location within the sequence of the key/data pair to return. |
|----|--------|------------------------------------------------------------------|

## Returns

The key of the currently sequenced record.

## Exceptions

|                                           |                                                             |
|-------------------------------------------|-------------------------------------------------------------|
| <b>Error::ObjectDoesNotExist</b> (p. 457) | End of sequencing.                                          |
| <b>Error::StrategyError</b> (p. 567)      | An error occurred when using the underlying storage system. |

Implements **BiometricEvaluation::IO::RecordStore** (p. 515).

**G.133.2.14 setCursorAtKey()**

```
void BiometricEvaluation::IO::SQLiteRecordStore::setCursorAtKey (
 const std::string & key) [override], [virtual]
```

Set the sequence cursor to an arbitrary position within the **RecordStore** (p. 504), starting at key. Key will be the first record returned from the next call to **sequence()** (p. 559).

## Parameters

|    |     |                                                                                                          |
|----|-----|----------------------------------------------------------------------------------------------------------|
| in | key | The key of the record which will be returned by the first subsequent call to <b>sequence()</b> (p. 559). |
|----|-----|----------------------------------------------------------------------------------------------------------|

## Exceptions

|                                           |                                                             |
|-------------------------------------------|-------------------------------------------------------------|
| <b>Error::ObjectDoesNotExist</b> (p. 457) | A record for the key does not exist.                        |
| <b>Error::StrategyError</b> (p. 567)      | An error occurred when using the underlying storage system. |

Implements **BiometricEvaluation::IO::RecordStore** (p. 516).

**G.133.2.15 sync()**

```
void BiometricEvaluation::IO::SQLiteRecordStore::sync () const [override], [virtual]
```

Synchronize the entire record store to persistent storage.

## Exceptions

|                                      |                                                             |
|--------------------------------------|-------------------------------------------------------------|
| <b>Error::StrategyError</b> (p. 567) | An error occurred when using the underlying storage system. |
|--------------------------------------|-------------------------------------------------------------|

Implements **BiometricEvaluation::IO::RecordStore** (p. 516).

**G.134 BiometricEvaluation::Process::Statistics Class Reference**

The **Statistics** (p. 560) class provides an interface for gathering process statistics, such as memory usage, system time, etc.

```
#include <be_process_statistics.h>
```

Public Member Functions

- **Statistics** ()
- **Statistics** ( **IO::FileLogCabinet** \*const logCabinet)
- **Statistics** (const std::shared\_ptr< **IO::Logsheet** > &logSheet)  
*Construct a Statistic object that logs to an existing Logsheet.*
- void **getCPUTimes** (uint64\_t \*usertime, uint64\_t \*systemtime)
- void **getMemorySizes** (uint64\_t \*vmrss, uint64\_t \*vmsize, uint64\_t \*vmpeak, uint64\_t \*vmdata, uint64\_t \*vmstack)
- uint32\_t **getNumThreads** ()
- void **logStats** ()  
*Create a snapshot of the current process statistics in the FileLogsheet created in the FileLogCabinet.*
- void **startAutoLogging** (uint64\_t interval)  
*Start logging process statistics automatically, in intervals of microseconds. The first log entry will occur soon after the call to this method as the delay interval is invoked after the first entry.*
- void **stopAutoLogging** ()  
*Stop the automatic logging of process statistics.*
- void **callStatistics\_logStats** ()

G.134.1 Detailed Description

The **Statistics** (p. 560) class provides an interface for gathering process statistics, such as memory usage, system time, etc.

The information gathered by objects of this class are for the current process, and can optionally be logged to a FileLogsheet object contained within the provided FileLogCabinet.

Note

The resolution of a returned value for many methods may not match the resolution allowed by the interface. For example, the operating system may allow for second resolution whereas the interface allows microsecond resolution.

G.134.2 Constructor & Destructor Documentation

G.134.2.1 Statistics() [1/3]

BiometricEvaluation::Process::Statistics::Statistics ( )  
Constructor with no parameters.

G.134.2.2 Statistics() [2/3]

BiometricEvaluation::Process::Statistics::Statistics ( **IO::FileLogCabinet** \*const logCabinet )  
Construct a **Statistics** (p. 560) object with the associated FileLogCabinet.

Parameters

|    |            |                                                                                                                              |
|----|------------|------------------------------------------------------------------------------------------------------------------------------|
| in | logCabinet | The FileLogCabinet object where this object will create a FileLogsheet to contain the statistic information for the process. |
|----|------------|------------------------------------------------------------------------------------------------------------------------------|

## Exceptions

|                                                               |                                                                                                                                     |
|---------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------|
| <b><i>Error::NotImplemented</i></b> (p. <a href="#">456</a> ) | Logging is not supported on this OS. This exception can be thrown when any portion of the statistics gathering cannot be completed. |
| <b><i>Error::ObjectExists</i></b> (p. <a href="#">458</a> )   | The FileLogsheet already exists. This exception should rarely, if ever, occur.                                                      |
| <b><i>Error::StrategyError</i></b> (p. <a href="#">567</a> )  | Failure to create the FileLogsheet in the cabinet.                                                                                  |

**G.134.2.3 Statistics()** [3/3]

```
BiometricEvaluation::Process::Statistics::Statistics (
 const std::shared_ptr< IO::Logsheet > & logSheet)
```

Construct a Statistic object that logs to an existing Logsheet.

## Parameters

|    |                 |                                          |
|----|-----------------|------------------------------------------|
| in | <i>logSheet</i> | Existing Logsheet that will be appended. |
|----|-----------------|------------------------------------------|

## Exceptions

|                                                               |                                                                                                                                     |
|---------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------|
| <b><i>Error::NotImplemented</i></b> (p. <a href="#">456</a> ) | Logging is not supported on this OS. This exception can be thrown when any portion of the statistics gathering cannot be completed. |
|---------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------|

**G.134.3 Member Function Documentation****G.134.3.1 callStatistics\_logStats()**

```
void BiometricEvaluation::Process::Statistics::callStatistics_logStats ()
```

Helper function in C++ space that has access to this object, and is called from C space by the logging thread. Applications should not call this function.

**G.134.3.2 getCPUTimes()**

```
void BiometricEvaluation::Process::Statistics::getCPUTimes (
 uint64_t * usertime,
 uint64_t * systemtime)
```

Obtain the total user and system times for the process, in microseconds. Any of the out parameters can be nullptr, indicating non-interest in that statistic.

## Note

This method may not be implemented in all operating systems.

## Parameters

|     |                   |                                               |
|-----|-------------------|-----------------------------------------------|
| out | <i>usertime</i>   | Pointer where to store the total user time.   |
| out | <i>systemtime</i> | Pointer where to store the total system time. |

## Exceptions

|                                              |                                                                                                                                                |
|----------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------|
| <b><i>Error::StrategyError</i></b> (p. 567)  | An error occurred when obtaining the process statistics from the operating system. The exception information string contains the error reason. |
| <b><i>Error::NotImplemented</i></b> (p. 456) | This method is not implemented on this OS.                                                                                                     |

## G.134.3.3 getMemorySizes()

```
void BiometricEvaluation::Process::Statistics::getMemorySizes (
 uint64_t * vmrss,
 uint64_t * vmsize,
 uint64_t * vmpeak,
 uint64_t * vmdata,
 uint64_t * vmstack)
```

Obtain the current memory set sizes for the process, in kilobytes. Any of the out parameters can be nullptr, indicating non-interest in that statistic.

## Note

This method may not be implemented in all operating systems.

## Parameters

|     |                |                                                                       |
|-----|----------------|-----------------------------------------------------------------------|
| out | <i>vmrss</i>   | Pointer where to store the current resident set size.                 |
| out | <i>vmsize</i>  | Pointer where to store the current total virtual memory size.         |
| out | <i>vmpeak</i>  | Pointer where to store the peak total virtual memory size.            |
| out | <i>vmdata</i>  | Pointer where to store the current virtual memory data segment size.  |
| out | <i>vmstack</i> | Pointer where to store the current virtual memory stack segment size. |

## Exceptions

|                                              |                                                                                                                                                |
|----------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------|
| <b><i>Error::StrategyError</i></b> (p. 567)  | An error occurred when obtaining the process statistics from the operating system. The exception information string contains the error reason. |
| <b><i>Error::NotImplemented</i></b> (p. 456) | This method is not implemented on this OS.                                                                                                     |

**G.134.3.4 getNumThreads()**

```
uint32_t BiometricEvaluation::Process::Statistics::getNumThreads ()
```

Obtain the number of threads composing this process.

**Note**

This method may not be implemented in all operating systems.

**Exceptions**

|                                                               |                                                                                                                                          |
|---------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------|
| <b><i>Error::StrategyError</i></b> (p. <a href="#">567</a> )  | An error occurred when obtaining the process info from the operating system. The exception information string contains the error reason. |
| <b><i>Error::NotImplemented</i></b> (p. <a href="#">456</a> ) | This method is not implemented on this OS.                                                                                               |

**G.134.3.5 logStats()**

```
void BiometricEvaluation::Process::Statistics::logStats ()
```

Create a snapshot of the current process statistics in the FileLogsheets created in the FileLogCabinet.

**Exceptions**

|                                                                   |                                                                                           |
|-------------------------------------------------------------------|-------------------------------------------------------------------------------------------|
| <b><i>Error::ObjectDoesNotExist</i></b> (p. <a href="#">457</a> ) | The FileLogsheets does not exist; this object was not created with FileLogCabinet object. |
| <b><i>Error::StrategyError</i></b> (p. <a href="#">567</a> )      | An error occurred when writing to the FileLogsheets.                                      |
| <b><i>Error::NotImplemented</i></b> (p. <a href="#">456</a> )     | The statistics gathering is not implemented for this operating system.                    |

**G.134.3.6 startAutoLogging()**

```
void BiometricEvaluation::Process::Statistics::startAutoLogging (
 uint64_t interval)
```

Start logging process statistics automatically, in intervals of microseconds. The first log entry will occur soon after the call to this method as the delay interval is invoked after the first entry.

**Note**

It is unrealistic to expect that log entries can be made at a rate of one per microsecond.

If **stopAutoLogging()** (p. [565](#)) is called very soon after the start, a log entry may not be made.

**Parameters**

|    |                 |                                                     |
|----|-----------------|-----------------------------------------------------|
| in | <i>interval</i> | The gap between logging snapshots, in microseconds. |
|----|-----------------|-----------------------------------------------------|

## Exceptions

|                                                            |                                                                                          |
|------------------------------------------------------------|------------------------------------------------------------------------------------------|
| <b>Error::ObjectDoesNotExist</b> (p. <a href="#">457</a> ) | The FileLogsheet does not exist; this object was not created with FileLogCabinet object. |
| <b>Error::ObjectExists</b> (p. <a href="#">458</a> )       | Autologging is currently invoked.                                                        |
| <b>Error::StrategyError</b> (p. <a href="#">567</a> )      | An error occurred when writing to the FileLogsheet.                                      |
| <b>Error::NotImplemented</b> (p. <a href="#">456</a> )     | The statistics gathering is not implemented for this operating system.                   |

## G.134.3.7 stopAutoLogging()

```
void BiometricEvaluation::Process::Statistics::stopAutoLogging ()
 Stop the automatic logging of process statistics.
```

## Exceptions

|                                                            |                                                                               |
|------------------------------------------------------------|-------------------------------------------------------------------------------|
| <b>Error::ObjectDoesNotExist</b> (p. <a href="#">457</a> ) | Not currently autologging.                                                    |
| <b>Error::StrategyError</b> (p. <a href="#">567</a> )      | An error occurred when stopping, most likely because the logging thread died. |

## G.135 BiometricEvaluation::Framework::Status Class Reference

```
#include <be_framework_status.h>
```

## Public Member Functions

- **Status** (int32\_t code= **OK**, const std::string &message="") noexcept  
*Status* (p. [565](#)) *constructor*.
- int32\_t **getCode** () const noexcept  
*Obtain the return code from this Status* (p. [565](#)).
- std::string **getMessage** () const noexcept  
*Obtain the explanatory message from this Status* (p. [565](#)).

## Static Public Attributes

- static const int32\_t **OK** = 0

## G.135.1 Detailed Description

Type to be returned from **API** (p. [221](#)) methods

## G.135.2 Constructor &amp; Destructor Documentation

### G.135.2.1 Status()

```
BiometricEvaluation::Framework::Status::Status (
 int32_t code = OK,
 const std::string & message = "") [noexcept]
```

**Status** (p. 565) constructor.

Parameters

|                |                                              |
|----------------|----------------------------------------------|
| <i>code</i>    | Return code from a function or method.       |
| <i>message</i> | Message providing insight into code's value. |

## G.135.3 Member Function Documentation

### G.135.3.1 getCode()

```
int32_t BiometricEvaluation::Framework::Status::getCode () const [inline], [noexcept]
```

Obtain the return code from this **Status** (p. 565).

Returns

Return code

### G.135.3.2 getMessage()

```
std::string BiometricEvaluation::Framework::Status::getMessage () const [inline], [noexcept]
```

Obtain the explanatory message from this **Status** (p. 565).

Returns

Explanator message.

Note

May be empty.

## G.135.4 Member Data Documentation

### G.135.4.1 OK

```
const int32_t BiometricEvaluation::Framework::Status::OK = 0 [static]
```

Successful return. Nothing to report.

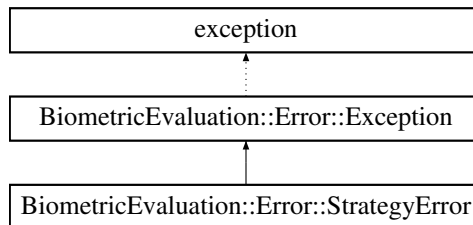


## G.136 BiometricEvaluation::Error::StrategyError Class Reference

A **StrategyError** (p. 567) object is thrown when the underlying implementation of this interface encounters an error.

```
#include <be_error_exception.h>
```

Inheritance diagram for BiometricEvaluation::Error::StrategyError:



### Public Member Functions

- **StrategyError** ()
- **StrategyError** (const std::string &info)

#### G.136.1 Detailed Description

A **StrategyError** (p. 567) object is thrown when the underlying implementation of this interface encounters an error.

#### G.136.2 Constructor & Destructor Documentation

##### G.136.2.1 StrategyError() [1/2]

```
BiometricEvaluation::Error::StrategyError::StrategyError ()
```

Construct a **StrategyError** (p. 567) object with the default information string.

##### G.136.2.2 StrategyError() [2/2]

```
BiometricEvaluation::Error::StrategyError::StrategyError (
 const std::string & info)
```

Construct a **StrategyError** (p. 567) object with an information string appended to the default information string.

## G.137 BiometricEvaluation::Video::Stream Class Reference

### Public Member Functions

- virtual float **getFPS** ()=0  
*Obtain the average frame rate of the video stream.*
- virtual uint64\_t **getFrameCount** ()=0  
*Obtain the number of frames in the video stream.*
- virtual **Video::Frame** **getFrame** (uint32\_t frameNum)=0

- *Obtain a frame from the video stream.*  
virtual std::vector< **Video::Frame** > **getFrameSequence** (int64\_t startTime, int64\_t endTime)=0
- *Obtain a sequence of frames from the video stream.*  
virtual void **setFrameScale** (float xScale, float yScale)=0
- *Set the scaling factors for returned video frames.*  
virtual void **setFramePixelFormat** (const **Image::PixelFormat** pixelFormat)=0
- *Set the pixel format for returned video frames.*

## G.137.1 Member Function Documentation

### G.137.1.1 getFPS()

virtual float BiometricEvaluation::Video::Stream::getFPS ( ) [pure virtual]  
Obtain the average frame rate of the video stream.

#### Returns

The average frame rate. A value of 0 means the frame rate cannot be determined.

### G.137.1.2 getFrame()

virtual **Video::Frame** BiometricEvaluation::Video::Stream::getFrame (   
uint32\_t frameNum ) [pure virtual]  
Obtain a frame from the video stream.

#### Parameters

|                 |                                    |
|-----------------|------------------------------------|
| <i>frameNum</i> | <b>Frame</b> (p. 346) number, >= 1 |
|-----------------|------------------------------------|

#### Exceptions

|                                       |                                                                              |
|---------------------------------------|------------------------------------------------------------------------------|
| <b>Error::ParameterError</b> (p. 474) | frameNum is too large.                                                       |
| <b>Error::StrategyError</b> (p. 567)  | No codec available for the video stream or other failure to read the stream. |

### G.137.1.3 getFrameCount()

virtual uint64\_t BiometricEvaluation::Video::Stream::getFrameCount ( ) [pure virtual]  
Obtain the number of frames in the video stream.

#### Returns

The number of frames in the stream; will be 0 if unknown.

**G.137.1.4 getFrameSequence()**

```
virtual std::vector< Video::Frame> BiometricEvaluation::Video::Stream::getFrameSequence (
 int64_t startTime,
 int64_t endTime) [pure virtual]
```

Obtain a sequence of frames from the video stream.

The end time can be greater than the length of the stream, and is not considered an error. Frames up to and including the last will be returned.

Parameters

|                  |                                                       |
|------------------|-------------------------------------------------------|
| <i>startTime</i> | Approximate time of the starting frame, milliseconds. |
| <i>endTime</i>   | Approximate time of the ending frame, milliseconds    |

Exceptions

|                                                       |                                                                              |
|-------------------------------------------------------|------------------------------------------------------------------------------|
| <b>Error::StrategyError</b> (p. <a href="#">567</a> ) | No codec available for the video stream or other failure to read the stream. |
|-------------------------------------------------------|------------------------------------------------------------------------------|

**G.137.1.5 setFramePixelFormat()**

```
virtual void BiometricEvaluation::Video::Stream::setFramePixelFormat (
 const Image::PixelFormat pixelFormat) [pure virtual]
```

Set the pixel format for returned video frames.

Parameters

|                    |                                          |
|--------------------|------------------------------------------|
| <i>pixelFormat</i> | The pixel format of all returned frames. |
|--------------------|------------------------------------------|

**G.137.1.6 setFrameScale()**

```
virtual void BiometricEvaluation::Video::Stream::setFrameScale (
 float xScale,
 float yScale) [pure virtual]
```

Set the scaling factors for returned video frames.

Parameters

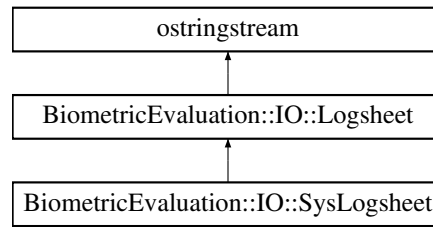
|               |                                      |
|---------------|--------------------------------------|
| <i>xScale</i> | The scaling factor for frame width.  |
| <i>yScale</i> | The scaling factor for frame height. |

**G.138 BiometricEvaluation::IO::SysLogsheet Class Reference**

A class to represent a single logging mechanism to a logging service on the network.

```
#include <be_io_syslogsheet.h>
```

Inheritance diagram for BiometricEvaluation::IO::SysLogsheet:



## Public Member Functions

- **SysLogsheet** (const std::string &url, const std::string &description, const std::string &appname, bool sequenced, bool utc)

*Create a new log sheet.*

- **SysLogsheet** (const std::string &url, const std::string &description, const std::string &appname, const std::string &hostname, bool sequenced, bool utc)

*Create a new log sheet.*

- **~SysLogsheet** ()
- void **write** (const std::string &entry)
- void **writeComment** (const std::string &entry)
- void **writeDebug** (const std::string &entry)
- void **sync** ()

*Write a string as an entry to the backing store.*

*Write a string as a comment to the backing store.*

*Write a string as a debug entry to the backing store.*

*Synchronize any buffered data to the underlying backing store.*

## Protected Member Functions

- **SysLogsheet** (const **SysLogsheet** &)
- **SysLogsheet** & **operator=** (const **SysLogsheet** &)
- void **setup** (const std::string &url, const std::string &description)
- void **writeToLogger** (const std::string &priority, const char delimiter, const std::string &prefix, const std::string &message)

## Protected Attributes

- std::string **\_hostname**
- std::string **\_appname**
- std::string **\_procid**
- int **\_sockFD**
- bool **\_sequenced**
- bool **\_operational**
- bool **\_utc**

Additional Inherited Members

G.138.1 Detailed Description

A class to represent a single logging mechanism to a logging service on the network.

Log entries are sent to the logging server in RFC5424 format with a timestamp of the local system in UTC. Normal and comment entries are sent to the logger with a PRI field indicating the 'local0' facility and a severity of 'Informational'. Debug entries are sent with facility of 'local1' and severity 'Debug'. A basic syslog config file would contain these lines: local0.info /var/log/info.log local1.debug /var/log/debug.log

The hostname is added to each entry but may be overridden by constructing the object with a given host-name, including the RFC5424 NILVALUE character. The PROCID part of each log message will be filled in with the process ID. Multi-line messages are segmented and sent the to logger as separate entries with the same timestamp and sequence number.

G.138.2 Constructor & Destructor Documentation

G.138.2.1 SysLogsheet() [1/3]

```
BiometricEvaluation::IO::SysLogsheet::SysLogsheet (
 const std::string & url,
 const std::string & description,
 const std::string & appname,
 bool sequenced,
 bool utc)
```

Create a new log sheet.

Parameters

|    |                    |                                                                                                         |
|----|--------------------|---------------------------------------------------------------------------------------------------------|
| in | <i>url</i>         | The Uniform Resource Locator describing the logging service. Accepted forms are syslog://hostname:port  |
| in | <i>description</i> | The text used to describe the sheet. This text is written into the log prior to any entries.            |
| in | <i>appname</i>     | The name of the application. This text is written into each log entry.                                  |
| in | <i>sequenced</i>   | True if each entry should include a sequence number, false if not.                                      |
| in | <i>utc</i>         | True if timestamps should be in Coordinated Universal <b>Time</b> (p. 161) (UTC), false for local time. |

Exceptions

|                                      |                                                                               |
|--------------------------------------|-------------------------------------------------------------------------------|
| <b>Error::StrategyError</b> (p. 567) | An error occurred when connecting to the logging system, or URL is malformed. |
|--------------------------------------|-------------------------------------------------------------------------------|

G.138.2.2 SysLogsheet() [2/3]

```
BiometricEvaluation::IO::SysLogsheet::SysLogsheet (
 const std::string & url,
 const std::string & description,
```

```
const std::string & appname,
const std::string & hostname,
bool sequenced,
bool utc)
Create a new log sheet.
```

Parameters

|    |                    |                                                                                                         |
|----|--------------------|---------------------------------------------------------------------------------------------------------|
| in | <i>url</i>         | The Uniform Resource Locator describing the logging service. Accepted forms are syslog://hostname:port  |
| in | <i>description</i> | The text used to describe the sheet. This text is written into the log prior to any entries.            |
| in | <i>appname</i>     | The name of the application. This text is written into each log entry.                                  |
| in | <i>hostname</i>    | The string to use as the hostname for all log entries.                                                  |
| in | <i>sequenced</i>   | True if each entry should include a sequence number, false if not.                                      |
| in | <i>utc</i>         | True if timestamps should be in Coordinated Universal <b>Time</b> (p. 161) (UTC), false for local time. |

Exceptions

|                                      |                                                                               |
|--------------------------------------|-------------------------------------------------------------------------------|
| <b>Error::StrategyError</b> (p. 567) | An error occurred when connecting to the logging system, or URL is malformed. |
|--------------------------------------|-------------------------------------------------------------------------------|

G.138.2.3 ~SysLogsheet()

```
BiometricEvaluation::IO::SysLogsheet::~~SysLogsheet ()
Destructor
```

G.138.2.4 SysLogsheet() [3/3]

```
BiometricEvaluation::IO::SysLogsheet::SysLogsheet (
const SysLogsheet &) [protected]
Prevent copying of SysLogsheet (p. 569) objects
```

G.138.3 Member Function Documentation

G.138.3.1 operator=()

```
SysLogsheet& BiometricEvaluation::IO::SysLogsheet::operator= (
const SysLogsheet &) [protected]
Prevent copying of SysLogsheet (p. 569) objects
```

G.138.3.2 setup()

```
void BiometricEvaluation::IO::SysLogsheet::setup (
const std::string & url,
const std::string & description) [protected]
```

Helper function to build connections

### G.138.3.3 sync()

```
void BiometricEvaluation::IO::SysLogsheet::sync () [virtual]
```

Synchronize any buffered data to the underlying backing store.

This syncing is dependent on the behavior of the underlying storage mechanism.

Exceptions

|                                                       |                                                            |
|-------------------------------------------------------|------------------------------------------------------------|
| <b>Error::StrategyError</b> (p. <a href="#">567</a> ) | An error occurred when using the underlying backing store. |
|-------------------------------------------------------|------------------------------------------------------------|

Reimplemented from **BiometricEvaluation::IO::Logsheet** (p. [427](#)).

### G.138.3.4 write()

```
void BiometricEvaluation::IO::SysLogsheet::write (
 const std::string & entry) [virtual]
```

Write a string as an entry to the backing store.

This does not affect the current log entry buffer, but does increment the entry number.

Parameters

|    |              |                            |
|----|--------------|----------------------------|
| in | <i>entry</i> | The text of the log entry. |
|----|--------------|----------------------------|

Exceptions

|                                                       |                                                            |
|-------------------------------------------------------|------------------------------------------------------------|
| <b>Error::StrategyError</b> (p. <a href="#">567</a> ) | An error occurred when using the underlying backing store. |
|-------------------------------------------------------|------------------------------------------------------------|

Reimplemented from **BiometricEvaluation::IO::Logsheet** (p. [427](#)).

### G.138.3.5 writeComment()

```
void BiometricEvaluation::IO::SysLogsheet::writeComment (
 const std::string & entry) [virtual]
```

Write a string as a comment to the backing store.

This does not affect the current log entry buffer, and does not increment the entry number. A comment line is prefixed with CommentDelimiter followed by a space by this method.

Parameters

|    |              |                          |
|----|--------------|--------------------------|
| in | <i>entry</i> | The text of the comment. |
|----|--------------|--------------------------|

Exceptions

|                                                       |                                                            |
|-------------------------------------------------------|------------------------------------------------------------|
| <b>Error::StrategyError</b> (p. <a href="#">567</a> ) | An error occurred when using the underlying backing store. |
|-------------------------------------------------------|------------------------------------------------------------|

Reimplemented from **BiometricEvaluation::IO::Logsheet** (p. 428).

### G.138.3.6 writeDebug()

```
void BiometricEvaluation::IO::SysLogsheet::writeDebug (
 const std::string & entry) [virtual]
```

Write a string as a debug entry to the backing store.

This does not affect the current log entry buffer, and does not increment the entry number. A debug line is prefixed with DebugDelimiter followed by a space.

Parameters

|    |       |                                |
|----|-------|--------------------------------|
| in | entry | The text of the debug message. |
|----|-------|--------------------------------|

Exceptions

|                                      |                                 |
|--------------------------------------|---------------------------------|
| <b>Error::StrategyError</b> (p. 567) | An error occurred when logging. |
|--------------------------------------|---------------------------------|

Reimplemented from **BiometricEvaluation::IO::Logsheet** (p. 428).

### G.138.3.7 writeToLogger()

```
void BiometricEvaluation::IO::SysLogsheet::writeToLogger (
 const std::string & priority,
 const char delimiter,
 const std::string & prefix,
 const std::string & message) [protected]
```

Helper function to write to the logger

## G.138.4 Member Data Documentation

### G.138.4.1 \_operational

```
bool BiometricEvaluation::IO::SysLogsheet::_operational [protected]
```

Whether the sheet is operational

### G.138.4.2 \_sequenced

```
bool BiometricEvaluation::IO::SysLogsheet::_sequenced [protected]
```

Whether to include entry sequence numbers

### G.138.4.3 \_sockFD

```
int BiometricEvaluation::IO::SysLogsheet::_sockFD [protected]
```

Socket file descriptor for the logging system



G.138.4.4 `_utc`

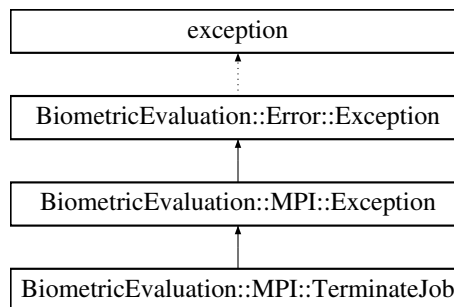
```
bool BiometricEvaluation::IO::SysLogsheet::_utc [protected]
 Whether time stamps are in UTC
```

## G.139 BiometricEvaluation::MPI::TerminateJob Class Reference

An exception that when thrown from a Task should result in the entire job (all tasks) being shut down by the **Distributor** (p. 305).

```
#include <be_mpi_exception.h>
```

Inheritance diagram for BiometricEvaluation::MPI::TerminateJob:



## Public Member Functions

- **TerminateJob** ()
- **TerminateJob** (std::string info)

*Constructor.*

## G.139.1 Detailed Description

An exception that when thrown from a Task should result in the entire job (all tasks) being shut down by the **Distributor** (p. 305).

## G.139.2 Constructor &amp; Destructor Documentation

## G.139.2.1 TerminateJob() [1/2]

```
BiometricEvaluation::MPI::TerminateJob::TerminateJob ()
 Construct with default information string.
```

## G.139.2.2 TerminateJob() [2/2]

```
BiometricEvaluation::MPI::TerminateJob::TerminateJob (
 std::string info)
 Constructor.
```

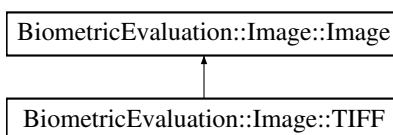
Parameters

|             |                                                                                |
|-------------|--------------------------------------------------------------------------------|
| <i>info</i> | Custom information string. Will be appended to the default information string. |
|-------------|--------------------------------------------------------------------------------|

## G.140 BiometricEvaluation::Image::TIFF Class Reference

```
#include <be_image_tiff.h>
```

Inheritance diagram for BiometricEvaluation::Image::TIFF:



### Public Member Functions

- **TIFF** (const uint8\_t \*data, const uint64\_t size)
- **TIFF** (const **Memory::uint8Array** &data)
- **Memory::uint8Array** **getRawData** () const  
*Accessor for the raw image data. The data returned should not be compressed or encoded.*
- **Memory::uint8Array** **getRawGrayscaleData** (uint8\_t depth) const  
*Accessor for decompressed data in grayscale.*

### Static Public Member Functions

- static bool **isTIFF** (const uint8\_t \*data, const uint64\_t size)  
*Determine if image is encoded as **TIFF** (p. 576).*
- static bool **isTIFF** (const **Memory::uint8Array** &data)  
*Determine if image is encoded as **TIFF** (p. 576).*

### Additional Inherited Members

#### G.140.1 Detailed Description

A TIFF-encoded image.

#### G.140.2 Member Function Documentation

##### G.140.2.1 getRawData()

```
Memory::uint8Array BiometricEvaluation::Image::TIFF::getRawData () const [virtual]
```

Accessor for the raw image data. The data returned should not be compressed or encoded.

Returns

AutoArray holding raw image data.

Exceptions

|                                  |                                                 |
|----------------------------------|-------------------------------------------------|
| <b>Error::DataError</b> (p. 295) | <b>Error</b> (p. 108) decompressing image data. |
|----------------------------------|-------------------------------------------------|

Implements **BiometricEvaluation::Image::Image** (p. 361).

G.140.2.2 **getRawGrayscaleData()**

**Memory::uint8Array** BiometricEvaluation::Image::TIFF::getRawGrayscaleData (   
     uint8\_t depth ) const [virtual]  
     Accessor for decompressed data in grayscale.

Parameters

|       |                                                                                         |
|-------|-----------------------------------------------------------------------------------------|
| depth | The desired bit depth of the resulting raw image. This value may either be 16, 8, or 1. |
|-------|-----------------------------------------------------------------------------------------|

Returns

AutoArray holding raw grayscale image data.

Exceptions

|                                       |                                                     |
|---------------------------------------|-----------------------------------------------------|
| <b>Error::DataError</b> (p. 295)      | <b>Error</b> (p. 108) decompressing image data.     |
| <b>Error::NotImplemented</b> (p. 456) | Unsupported conversion based on source color depth. |
| <b>Error::ParameterError</b> (p. 474) | Invalid value for depth.                            |

Note

This method does not save a cached copy of the decompressed image because the bit depth of the image can be changed between calls.  
When depth is 1, this method returns an image that uses 8 bits to represent a single pixel. The depth parameter is used to adjust the number of gray levels. When depth is 1, there are only 2 gray levels (black and white), despite using 8 bits to represent each pixel.  
Alpha channels are completely ignored when converting to grayscale.

Implements **BiometricEvaluation::Image::Image** (p. 362).

G.140.2.3 **isTIFF()** [1/2]

static bool BiometricEvaluation::Image::TIFF::isTIFF (   
     const uint8\_t \* data,  
     const uint64\_t size ) [static]  
     Determine if image is encoded as **TIFF** (p. 576).

Parameters

|    |      |                               |
|----|------|-------------------------------|
| in | data | <b>Image</b> (p. 355) data.   |
| in | size | <b>Size</b> (p. 549) of data. |

## Returns

true if data appears to be encoded with **TIFF** (p. 576), false otherwise.

**G.140.2.4 isTIFF()** [2/2]

```
static bool BiometricEvaluation::Image::TIFF::isTIFF (
 const Memory::uint8Array & data) [static]
```

Determine if image is encoded as **TIFF** (p. 576).

## Parameters

|    |             |                             |
|----|-------------|-----------------------------|
| in | <i>data</i> | <b>Image</b> (p. 355) data. |
|----|-------------|-----------------------------|

## Returns

true if data appears to be encoded with **TIFF** (p. 576), false otherwise.

**G.141 BiometricEvaluation::Time::Timer Class Reference**

This class can be used by applications to report the amount of time a block of code takes to execute.

```
#include <be-time-timer.h>
```

**Public Types**

- using **BE\_CLOCK\_TYPE** = std::chrono::steady\_clock

**Public Member Functions**

- **Timer** ()
- **Timer** (const std::function< void()> &func)
 

*Construct a timer and time a function immediately.*
- void **start** ()
 

*Start tracking time.*
- void **stop** ()
 

*Stop tracking time.*
- uint64\_t **elapsed** (bool nano=false) const
 

*Get the elapsed time in microseconds or nanoseconds between calls to this object's **start**() (p. 580) and **stop**() (p. 581) methods.*
- std::string **elapsedStr** (bool displayUnits=false, bool nano=false) const
 

*Convenience method for printing elapsed time as a string.*
- **Timer & time** (const std::function< void()> &func)
 

*Record the runtime of a function.*

### G.141.1 Detailed Description

This class can be used by applications to report the amount of time a block of code takes to execute.

Applications wrap the block of code in the **Timer::start()** (p. 580) and **Timer::stop()** (p. 581) calls, then use **Timer::elapsed()** (p. 579) to obtain the calculated time of the operation.

#### Warning

Timers are not threadsafe and should only be used to time operations within the same thread.

### G.141.2 Member Typedef Documentation

#### G.141.2.1 BE\_CLOCK\_TYPE

```
using BiometricEvaluation::Time::Timer::BE_CLOCK_TYPE = std::chrono::steady_clock
```

Clock type to use, aliased for easy replacement.

### G.141.3 Constructor & Destructor Documentation

#### G.141.3.1 Timer() [1/2]

```
BiometricEvaluation::Time::Timer::Timer ()
```

Constructor for the **Timer** (p. 578) object.

#### G.141.3.2 Timer() [2/2]

```
BiometricEvaluation::Time::Timer::Timer (
 const std::function< void()> & func)
```

Construct a timer and time a function immediately.

#### Parameters

|             |                                 |
|-------------|---------------------------------|
| <i>func</i> | A function to time immediately. |
|-------------|---------------------------------|

#### Exceptions

|                                      |                                         |
|--------------------------------------|-----------------------------------------|
| <b>Error::StrategyError</b> (p. 567) | Propagated from <b>time()</b> (p. 581). |
|--------------------------------------|-----------------------------------------|

### G.141.4 Member Function Documentation

#### G.141.4.1 elapsed()

```
uint64_t BiometricEvaluation::Time::Timer::elapsed (
 bool nano = false) const
```

Get the elapsed time in microseconds or nanoseconds between calls to this object's **start()** (p. 580) and **stop()** (p. 581) methods.

Parameters

|             |                                                 |
|-------------|-------------------------------------------------|
| <i>nano</i> | True if to return nanoseconds, false otherwise. |
|-------------|-------------------------------------------------|

Returns

The number of microseconds or nanoseconds.

Exceptions

|                                             |                                                                                                      |
|---------------------------------------------|------------------------------------------------------------------------------------------------------|
| <b><i>Error::StrategyError</i></b> (p. 567) | This object is currently timing an operation or an error occurred when obtaining timing information. |
|---------------------------------------------|------------------------------------------------------------------------------------------------------|

#### G.141.4.2 elapsedStr()

```
std::string BiometricEvaluation::Time::Timer::elapsedStr (
 bool displayUnits = false,
 bool nano = false) const
```

Convenience method for printing elapsed time as a string.

Parameters

|                     |                                                 |
|---------------------|-------------------------------------------------|
| <i>displayUnits</i> | Append the elapsed time units.                  |
| <i>nano</i>         | True if to return nanoseconds, false otherwise. |

Returns

String representing the elapsed time.

Exceptions

|                                             |                                            |
|---------------------------------------------|--------------------------------------------|
| <b><i>Error::StrategyError</i></b> (p. 567) | Propagated from <b>elapsed()</b> (p. 579). |
|---------------------------------------------|--------------------------------------------|

#### G.141.4.3 start()

```
void BiometricEvaluation::Time::Timer::start ()
```

Start tracking time.

Exceptions

|                                             |                                                                                                      |
|---------------------------------------------|------------------------------------------------------------------------------------------------------|
| <b><i>Error::StrategyError</i></b> (p. 567) | This object is currently timing an operation or an error occurred when obtaining timing information. |
|---------------------------------------------|------------------------------------------------------------------------------------------------------|

G.141.4.4 stop()

```
void BiometricEvaluation::Time::Timer::stop ()
 Stop tracking time.
```

Exceptions

|                                      |                                                                                                          |
|--------------------------------------|----------------------------------------------------------------------------------------------------------|
| <i>Error::StrategyError</i> (p. 567) | This object is not currently timing an operation or an error occurred when obtaining timing information. |
|--------------------------------------|----------------------------------------------------------------------------------------------------------|

G.141.4.5 time()

```
Timer& BiometricEvaluation::Time::Timer::time (
 const std::function< void()> & func)
 Record the runtime of a function.
```

Parameters

|             |                   |
|-------------|-------------------|
| <i>func</i> | Function to time. |
|-------------|-------------------|

Returns

Reference to this class.

Exceptions

|                                      |                                                                                            |
|--------------------------------------|--------------------------------------------------------------------------------------------|
| <i>Error::StrategyError</i> (p. 567) | Propagated from <b>start()</b> (p. 580) or <b>stop()</b> (p. 581), and/or func is nullptr. |
|--------------------------------------|--------------------------------------------------------------------------------------------|

G.142 BiometricEvaluation::Device::TLV Class Reference

A class to represent a Tag-Length-Value (TLV (p. 581)) data structure as described in the ISO 7816-4 integrated circuit card standard.

```
#include <be_device_tlv.h>
```

Public Member Functions

- **TLV ()**  
*Construct an empty Tag-Length-Value object that can be filled with setter methods.*
- **TLV (const Memory::uint8Array &buf)**  
*Construct a Tag-Length-Value object from the given buffer.*
- **TLV ( Memory::IndexedBuffer &ibuf)**  
*Construct a single TLV (p. 581) from the indexed buffer.*
- **TLV (const std::string &filename)**

- Construct a Tag-Length-Value object from the given file name.*

  - void **setTag** (const **Memory::uint8Array** &tag)

*Set the encoded tag value.*
- const **Memory::uint8Array** **getTag** () const

*Obtain the encoded tag value.*
- uint32\_t **getTagNum** () const
- uint8\_t **getTagClass** () const
- bool **isPrimitive** () const
- void **setPrimitive** (const **Memory::uint8Array** &value)
- Set the primitive data associated with this **TLV** (p. 581).*

  - **Memory::uint8Array** **getPrimitive** () const

*Obtain the primitive data associated with this **TLV** (p. 581).*
- void **addChild** (const **TLV** &tlv)
- std::vector< **TLV** > **getChildren** () const
- **Memory::uint8Array** **getRawTLV** () const
- Obtain the **TLV** (p. 581) as an array of 8-bit values.*

## Static Public Member Functions

- static std::string **stringFromTLV** (const **TLV** &tlv, const int tabCount)
- Class utility function to print the contents of a **TLV** (p. 581) into a string object, in readable format.*

## G.142.1 Detailed Description

A class to represent a Tag-Length-Value (**TLV** (p. 581)) data structure as described in the ISO 7816-4 integrated circuit card standard.

A **TLV** (p. 581) is composed of tag and length fields, then a value field that may be another **TLV** (p. 581) (a child), or data of another format, represented as the primitive object in this class.

## G.142.2 Constructor & Destructor Documentation

### G.142.2.1 TLV() [1/4]

BiometricEvaluation::Device::TLV::TLV ( )

Construct an empty Tag-Length-Value object that can be filled with setter methods.

Empty **TLV** (p. 581) objects are primitive.

### G.142.2.2 TLV() [2/4]

BiometricEvaluation::Device::TLV::TLV (   
 const **Memory::uint8Array** & buf )

Construct a Tag-Length-Value object from the given buffer.

Exceptions

|                                  |                                           |
|----------------------------------|-------------------------------------------|
| <b>Error::DataError</b> (p. 295) | The data in the buffer is not conforming. |
|----------------------------------|-------------------------------------------|



### G.142.2.3 TLV() [3/4]

```
BiometricEvaluation::Device::TLV::TLV (
 Memory::IndexedBuffer & ibuf)
```

Construct a single **TLV** (p. 581) from the indexed buffer.

Exceptions

|                                  |                                                       |
|----------------------------------|-------------------------------------------------------|
| <b>Error::DataError</b> (p. 295) | <b>Error</b> (p. 108) parsing the data in the buffer. |
|----------------------------------|-------------------------------------------------------|

### G.142.2.4 TLV() [4/4]

```
BiometricEvaluation::Device::TLV::TLV (
 const std::string & filename)
```

Construct a Tag-Length-Value object from the given file name.

Exceptions

|                                  |                                          |
|----------------------------------|------------------------------------------|
| <b>Error::DataError</b> (p. 295) | The data in the file is not conformance. |
|----------------------------------|------------------------------------------|

## G.142.3 Member Function Documentation

### G.142.3.1 addChild()

```
void BiometricEvaluation::Device::TLV::addChild (
 const TLV & tlv)
```

Add a child **TLV** (p. 581).

Parameters

|            |                                                                             |
|------------|-----------------------------------------------------------------------------|
| <i>tlv</i> | The <b>TLV</b> (p. 581) to be added as a child of this <b>TLV</b> (p. 581). |
|------------|-----------------------------------------------------------------------------|

Exceptions

|                                  |                                       |
|----------------------------------|---------------------------------------|
| <b>Error::DataError</b> (p. 295) | The <b>TLV</b> (p. 581) is primitive. |
|----------------------------------|---------------------------------------|

### G.142.3.2 getChildren()

```
std::vector< TLV> BiometricEvaluation::Device::TLV::getChildren () const
```

Get copies of the child **TLVs**.

## Returns

A vector of child TLVs.

## Exceptions

|                                  |                                       |
|----------------------------------|---------------------------------------|
| <b>Error::DataError</b> (p. 295) | The <b>TLV</b> (p. 581) is primitive. |
|----------------------------------|---------------------------------------|

**G.142.3.3 getPrimitive()**

**Memory::uint8Array** BiometricEvaluation::Device::TLV::getPrimitive ( ) const

Obtain the primitive data associated with this **TLV** (p. 581).

## Exceptions

|                                  |                                                     |
|----------------------------------|-----------------------------------------------------|
| <b>Error::DataError</b> (p. 295) | The <b>TLV</b> (p. 581) is of the constructed form. |
|----------------------------------|-----------------------------------------------------|

## See also

**getChildren** (p. 583).

**G.142.3.4 getRawTLV()**

**Memory::uint8Array** BiometricEvaluation::Device::TLV::getRawTLV ( ) const

Obtain the **TLV** (p. 581) as an array of 8-bit values.

The array can be sent to a device that accepts TLV-encoded objects, typically wrapped in device command structures.

## Returns

The **TLV** (p. 581) as an array.

**G.142.3.5 getTagClass()**

uint8\_t BiometricEvaluation::Device::TLV::getTagClass ( ) const

Get the decoded tag class.

## Returns

The tag class.

**G.142.3.6 getTagNum()**

uint32\_t BiometricEvaluation::Device::TLV::getTagNum ( ) const

Get the decoded tag number.

## Returns

The tag number.

G.142.3.7 isPrimitive()

bool BiometricEvaluation::Device::TLV::isPrimitive ( ) const  
Obtain the type of **TLV** (p. 581): primitive/constructed.

Returns

True if is a primitive **TLV** (p. 581), false otherwise.

G.142.3.8 setPrimitive()

void BiometricEvaluation::Device::TLV::setPrimitive (   
const **Memory::uint8Array** & value )  
Set the primitive data associated with this **TLV** (p. 581).  
The primitive data is added as the value data item.

Exceptions

|                                  |                                                                                                                                        |
|----------------------------------|----------------------------------------------------------------------------------------------------------------------------------------|
| <b>Error::DataError</b> (p. 295) | The <b>TLV</b> (p. 581) is already of the constructed form, meaning that there are <b>TLV</b> (p. 581) children set as the value data. |
|----------------------------------|----------------------------------------------------------------------------------------------------------------------------------------|

G.142.3.9 setTag()

void BiometricEvaluation::Device::TLV::setTag (   
const **Memory::uint8Array** & tag )  
Set the encoded tag value.  
This function will cause a recalculation of the decoded tag number, class and primitive indicators.

Exceptions

|                                       |                                                                                                      |
|---------------------------------------|------------------------------------------------------------------------------------------------------|
| <b>Error::DataError</b> (p. 295)      | The primitive indicator conflicts with the presence of children TLVs, or presence of primitive data. |
| <b>Error::ParameterError</b> (p. 474) | The length of the buffer is larger than the maximum tag length.                                      |

G.142.3.10 stringFromTLV()

static std::string BiometricEvaluation::Device::TLV::stringFromTLV (   
const **TLV** & tlv,  
const int tabCount ) [static]  
Class utility function to print the contents of a **TLV** (p. 581) into a string object, in readable format.

Parameters

|          |                                                                        |
|----------|------------------------------------------------------------------------|
| tlv      | The <b>TLV</b> (p. 581) to print.                                      |
| tabCount | The number of tab characters to insert before each line of the output. |

## G.143 BiometricEvaluation::Memory::unique\_if< T > Struct Template Reference

Define a type that is visible when T is not an array.

```
#include <be_memory.h>
```

### Public Types

- using **unique\_single** = std::unique\_ptr< T >

#### G.143.1 Detailed Description

```
template<class T>
```

```
struct BiometricEvaluation::Memory::unique_if< T >
```

Define a type that is visible when T is not an array.

Note

Coming in C++14. This implementation is taken from the LLVM implementation.

#### G.143.2 Member Typedef Documentation

##### G.143.2.1 unique\_single

```
template<class T>
```

```
using BiometricEvaluation::Memory::unique_if< T >:: unique_single = std::unique_ptr<T>
```

Type to use when T is not an array.

## G.144 BiometricEvaluation::Memory::unique\_if< T[] > Struct Template Reference

Define a type that is visible when T is an unknown-bound array.

```
#include <be_memory.h>
```

### Public Types

- using **unique\_array\_unknown\_bound** = std::unique\_ptr< T[] >

#### G.144.1 Detailed Description

```
template<class T>
```

```
struct BiometricEvaluation::Memory::unique_if< T[] >
```

Define a type that is visible when T is an unknown-bound array.

Note

Coming in C++14. This implementation is taken from the LLVM implementation.

## G.144.2 Member Typedef Documentation

### G.144.2.1 `unique_array_unknown_bound`

```
template<class T>
using BiometricEvaluation::Memory::unique_if< T[]>:: unique_array_unknown_bound = std::unique_ptr<T[]>
```

### G.145 BiometricEvaluation::Memory::unique\_if< T[S]> Struct Template Reference

Define a type that is visible when T is an known-bound array.

```
#include <be_memory.h>
```

## Public Types

- using `unique_array_known_bound = void`

### G.145.1 Detailed Description

```
template<class T, size_t S>
struct BiometricEvaluation::Memory::unique_if< T[S]>
```

Define a type that is visible when T is an known-bound array.

### Note

Coming in C++14. This implementation is taken from the LLVM implementation.

## G.145.2 Member Typedef Documentation

### G.145.2.1 `unique_array_known_bound`

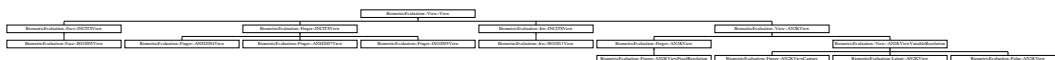
```
template<class T , size_t S>
using BiometricEvaluation::Memory::unique_if< T[S]>:: unique_array_known_bound = void
```

## G.146 BiometricEvaluation::View::View Class Reference

A class to represent single biometric element view.

```
#include <be_view_view.h>
```

Inheritance diagram for BiometricEvaluation::View::View:



## Public Member Functions

- `std::shared_ptr< Image::Image > getImage () const`  
Obtain the image used for the biometric view in the format contained in the record (JPEG, etc.)
- `Image::Size getImageSize () const`  
Obtain the image size.
- `Image::Resolution getImageResolution () const`  
Obtain the image resolution.
- `uint32_t getImageColorDepth () const`  
Obtain the image color depth in bits-per-pixel.
- `Image::CompressionAlgorithm getCompressionAlgorithm () const`  
Obtain the compression algorithm used on the image.
- `Image::Resolution getScanResolution () const`  
Obtain the image scan resolution.

## Protected Member Functions

- `void setImageSize (const BiometricEvaluation::Image::Size &imageSize)`  
Mutator for the image size.
- `void setImageColorDepth (uint32_t imageColorDepth)`  
Mutator for the image color depth.
- `void setImageResolution (const BiometricEvaluation::Image::Resolution &imageResolution)`  
Mutator for the image resolution.
- `void setScanResolution (const BiometricEvaluation::Image::Resolution &scanResolution)`  
Mutator for the image scan resolution.
- `void setImageData (const BiometricEvaluation::Memory::uint8Array &imageData)`  
Mutator for the image data.
- `void setCompressionAlgorithm (const Image::CompressionAlgorithm &ca)`  
Mutator for the compression algorithm.

### G.146.1 Detailed Description

A class to represent single biometric element view.

Included in a view is the biometric image and any derived information, such as minutiae points.

### G.146.2 Member Function Documentation

#### G.146.2.1 getCompressionAlgorithm()

`Image::CompressionAlgorithm BiometricEvaluation::View::View::getCompressionAlgorithm ( ) const`

Obtain the compression algorithm used on the image.

This value is as present in the biometric record, and not obtained from the image data itself.

Returns

The compression algorithm.

### G.146.2.2 getImage()

```
std::shared_ptr< Image::Image> BiometricEvaluation::View::View::getImage () const
```

Obtain the image used for the biometric view in the format contained in the record (JPEG, etc.)

Not all views will have an image, however the derived information, such as minutiae, may be present.

Returns

The image data.

### G.146.2.3 getImageColorDepth()

```
uint32_t BiometricEvaluation::View::View::getImageColorDepth () const
```

Obtain the image color depth in bits-per-pixel.

This value is as present in the biometric record, and not in the image data itself. Normally, this value and the actual image depth must be equal, but applications can check for inconsistencies. In the case of raw images, however, the value obtained with this method must be accepted as correct.

Returns

The image depth.

### G.146.2.4 getImageResolution()

```
Image::Resolution BiometricEvaluation::View::View::getImageResolution () const
```

Obtain the image resolution.

**Image** (p. 120) resolution is taken from the biometric record, and not from the image data.

Returns

The scan resolution.

Note

In some cases, the resolution may be the components of the pixel ratio, and applications must check the **Image::Resolution::Units** (p. 532) field for value NA.

### G.146.2.5 getImageSize()

```
Image::Size BiometricEvaluation::View::View::getImageSize () const
```

Obtain the image size.

This value is as present in the biometric record, and not in the image data itself. Normally, this value and the actual image size must be equal, but applications can check for inconsistencies. In the case of raw images, however, the value obtained with this method must be accepted as correct.

Returns

The image size.

**G.146.2.6 getScanResolution()**

**Image::Resolution** BiometricEvaluation::View::View::getScanResolution ( ) const

Obtain the image scan resolution.

This value is as present in the biometric record, and not in the image data itself. Normally, this value and the actual image resolution must be equal, but applications can check for inconsistencies.

Returns

The scan resolution.

Note

In some cases, the resolution may be the components of the pixel ratio, and applications must check the **Image::Resolution::Units** (p. 532) field for value NA.

**G.146.2.7 setImageColorDepth()**

void BiometricEvaluation::View::View::setImageColorDepth (   
 uint32\_t *imageColorDepth* ) [protected]

Mutator for the image color depth.

Parameters

|    |                        |                        |
|----|------------------------|------------------------|
| in | <i>imageColorDepth</i> | The image color depth. |
|----|------------------------|------------------------|

**G.146.2.8 setImageData()**

void BiometricEvaluation::View::View::setImageData (   
 const **BiometricEvaluation::Memory::uint8Array** & *imageData* ) [protected]

Mutator for the image data.

Parameters

|    |                  |                        |
|----|------------------|------------------------|
| in | <i>imageData</i> | The image data object. |
|----|------------------|------------------------|

**G.146.2.9 setImageResolution()**

void BiometricEvaluation::View::View::setImageResolution (   
 const **BiometricEvaluation::Image::Resolution** & *imageResolution* ) [protected]

Mutator for the image resolution.

Parameters

|    |                        |                              |
|----|------------------------|------------------------------|
| in | <i>imageResolution</i> | The image resolution object. |
|----|------------------------|------------------------------|



**G.146.2.10 setImageSize()**

```
void BiometricEvaluation::View::View::setImageSize (
 const BiometricEvaluation::Image::Size & imageSize) [protected]
```

Mutator for the image size.

Parameters

|    |                  |                        |
|----|------------------|------------------------|
| in | <i>imageSize</i> | The image size object. |
|----|------------------|------------------------|

**G.146.2.11 setScanResolution()**

```
void BiometricEvaluation::View::View::setScanResolution (
 const BiometricEvaluation::Image::Resolution & scanResolution) [protected]
```

Mutator for the image scan resolution.

Parameters

|    |                       |                                   |
|----|-----------------------|-----------------------------------|
| in | <i>scanResolution</i> | The image scan resolution object. |
|----|-----------------------|-----------------------------------|

**G.147 BiometricEvaluation::Time::Watchdog Class Reference**

A **Watchdog** (p. 591) object can be used by applications to limit the amount of processing time taken by a block of code.

```
#include <be_time_watchdog.h>
```

**Public Member Functions**

- **Watchdog** (const uint8\_t type)
- void **setInterval** (uint64\_t interval)
- void **start** ()
- void **stop** ()
- bool **expired** ()
- void **setCanSigJump** ()
- void **clearCanSigJump** ()
- void **setExpired** ()
- void **clearExpired** ()

**Static Public Attributes**

- static const uint8\_t **PROCESSTIME** = 0
- static const uint8\_t **REALTIME** = 1
- static bool **\_canSigJump**
- static sigjmp\_buf **\_sigJumpBuf**

### G.147.1 Detailed Description

A **Watchdog** (p. 591) object can be used by applications to limit the amount of processing time taken by a block of code.

A **Watchdog** (p. 591) object is used to set a timer that, upon expiration, will force a jump to a location within the process. An application can detect whether the timer expired at that point in the code. **Watchdog** (p. 591) builds on the POSIX `setitimer(2)` call. **Timer** (p. 578) intervals are in terms of process virtual time or real time, based on how the object is constructed.

Most applications will not directly invoke the methods of the `WatchDog` class, instead using the `BEGIN_WATCHDOG_BLOCK()` and `END_WATCHDOG_BLOCK()` macros. Applications should not install their own signal handlers, but use the `SignalManager` class instead.

The `BEGIN_WATCHDOG_BLOCK()` macro sets up the jump block and tells the **Watchdog** (p. 591) object to start handling the alarm signal. Applications must call `setInterval()` (p. 593) before invoking the `BEGIN_WATCHDOG_BLOCK()` macro.

The `END_WATCHDOG_BLOCK()` macro disables the watchdog timer, but doesn't affect the assigned interval value. Applications can set the interval once and use the block macros repeatedly. Failure to call `setInterval()` (p. 593) results in an effectively disabled timer, as does setting the interval to 0.

The `ABORT_WATCHDOG()` macro also disables the watchdog timer but does not create the code point destination for the jump point. This macro should be used to disable a **Watchdog** (p. 591) object when the application is no longer interested in the timeout condition.

#### Attention

The `BEGIN_WATCHDOG_BLOCK()` macro must be paired with either the `END_WATCHDOG_BLOCK()` macro or `ABORT_WATCHDOG_BLOCK()` macro. Failure to do so may result in undefined behavior as a running **Watchdog** (p. 591) timer may expire, forcing a jump into an incompletely initialized function.

#### Note

**Process** (p. 150) virtual timing may not be available on all systems. In those cases, an application compilation error will occur because `PROCESSTIME` will not be defined.

#### Attention

On many systems, the `sleep(3)` call is implemented using alarm signals, the same technique used by the **Watchdog** (p. 591) class. Therefore, applications should not call `sleep(3)` inside the **Watchdog** (p. 591) block; behavior is undefined in that case, but usually results in cancellation of the **Watchdog** (p. 591) timer.

The `setCanSigJump()` (p. 593), `clearCanSigJump()` (p. 593), `setExpired()` (p. 593) and `clearExpired()` (p. 593) methods are not meant to be used directly by applications, which should use the `BEGIN_WATCHDOG_BLOCK()/END_WATCHDOG_BLOCK()` macro pair.

See also

**Error::SignalManager** (p. 545)

### G.147.2 Constructor & Destructor Documentation

#### G.147.2.1 Watchdog()

```
BiometricEvaluation::Time::Watchdog::Watchdog (
 const uint8_t type)
```

Construct a new **Watchdog** (p. 591) object.

## Parameters

| in | type |                                             |
|----|------|---------------------------------------------|
|    |      | The type of timer, ProcessTime or RealTime. |

## Exceptions

|                                                        |                                                    |
|--------------------------------------------------------|----------------------------------------------------|
| <b>Error::NotImplemented</b> (p. <a href="#">456</a> ) | The type of watchdog requested is not implemented. |
| <b>Error::ParameterError</b> (p. <a href="#">474</a> ) | The type is invalid.                               |

## Warning

**Watchdog::PROCESSTIME** (p. [594](#)) is not supported under Cygwin.

## G.147.3 Member Function Documentation

### G.147.3.1 clearCanSigJump()

```
void BiometricEvaluation::Time::Watchdog::clearCanSigJump ()
```

Clears the flag for the **Watchdog** (p. [591](#)) object to indicate that the signal jump block is no longer valid.

### G.147.3.2 clearExpired()

```
void BiometricEvaluation::Time::Watchdog::clearExpired ()
```

Clear the flag indicating the timer expired.

### G.147.3.3 expired()

```
bool BiometricEvaluation::Time::Watchdog::expired ()
```

Indicate whether the watchdog timer expired.

## Returns

true if the timer expired, false otherwise.

### G.147.3.4 setCanSigJump()

```
void BiometricEvaluation::Time::Watchdog::setCanSigJump ()
```

Indicate that the signal handler can jump into the application code after handling the signal.

### G.147.3.5 setExpired()

```
void BiometricEvaluation::Time::Watchdog::setExpired ()
```

Set a flag to indicate the timer expired.

**G.147.3.6 setInterval()**

```
void BiometricEvaluation::Time::Watchdog::setInterval (
 uint64_t interval)
```

Set the interval for the timer, but don't start the timer. Setting a value of 0 will essentially disable the timer. **Timer** (p. 578) intervals are in microseconds, however actual intervals are dependent on the resolution of the system clock, and may not be at microsecond resolution.

Parameters

|    |                 |                                      |
|----|-----------------|--------------------------------------|
| in | <i>interval</i> | The timer interval, in microseconds. |
|----|-----------------|--------------------------------------|

**G.147.3.7 start()**

```
void BiometricEvaluation::Time::Watchdog::start ()
```

Start a watchdog timer.

Exceptions

|                                      |                                                                       |
|--------------------------------------|-----------------------------------------------------------------------|
| <b>Error::StrategyError</b> (p. 567) | Could not register the signal handler, or could not create the timer. |
|--------------------------------------|-----------------------------------------------------------------------|

**G.147.3.8 stop()**

```
void BiometricEvaluation::Time::Watchdog::stop ()
```

Stop a watchdog timer.

Exceptions

|                                      |                            |
|--------------------------------------|----------------------------|
| <b>Error::StrategyError</b> (p. 567) | Could not clear the timer. |
|--------------------------------------|----------------------------|

**G.147.4 Member Data Documentation****G.147.4.1 PROCESSTIME**

```
const uint8_t BiometricEvaluation::Time::Watchdog::PROCESSTIME = 0 [static]
```

A **Watchdog** (p. 591) based on process time.

**G.147.4.2 REALTIME**

```
const uint8_t BiometricEvaluation::Time::Watchdog::REALTIME = 1 [static]
```

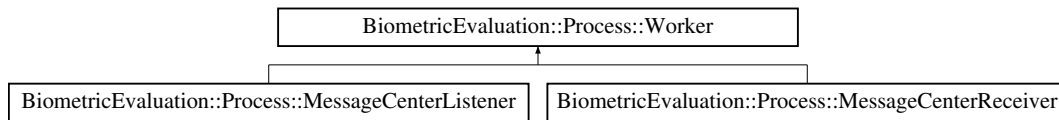
A **Watchdog** (p. 591) based on real (wall clock) time.

## G.148 BiometricEvaluation::Process::Worker Class Reference

An abstraction of an instance that performs work on given data.

```
#include <be_process_worker.h>
```

Inheritance diagram for BiometricEvaluation::Process::Worker:



### Public Member Functions

- virtual int32\_t **workerMain** ()=0  
*The method that will get called to start execution by a ProcessManager.*
- std::shared\_ptr< void > **getParameter** (const std::string &name)  
*Obtain a parameter passed to this **Worker** (p. 595).*
- double **getParameterAsDouble** (const std::string &name)  
*Obtain a parameter passed to this **Worker** (p. 595) as a double.*
- int64\_t **getParameterAsInteger** (const std::string &name)  
*Obtain a parameter passed to this **Worker** (p. 595) as an integer.*
- std::string **getParameterAsString** (const std::string &name)  
*Obtain a parameter passed to this **Worker** (p. 595) as a string.*
- void **setParameter** (const std::string &name, std::shared\_ptr< void > argument)  
*Pass a parameter to this **Worker** (p. 595).*
- virtual void **stop** () final  
*Tell this **Worker** (p. 595) to return ASAP.*
- void **closeWorkerPipeEnds** ()  
*Perform initialization for communication from **Worker** (p. 595) to **Manager** (p. 430).*
- void **closeManagerPipeEnds** ()  
*Perform initialization for communication from **Manager** (p. 430) to **Worker** (p. 595).*
- int **getSendingPipe** () const  
*Obtain the pipe used to send messages to this **Worker** (p. 595).*
- int **getReceivingPipe** () const  
*Obtain the pipe used to receive messages to this **Worker** (p. 595).*
- void **sendMessageToManager** (const Memory::uint8Array &message)  
*Send a message to the **Manager** (p. 430).*
- void **receiveMessageFromManager** (Memory::uint8Array &message)  
*Receive a message from the **Manager** (p. 430).*
- void **\_initCommunication** ()  
*Perform general communication initialization from Constructor.*
- virtual ~**Worker** ()  
***Worker** (p. 595) destructor.*

## Protected Member Functions

- **Worker** ()  
*Worker* (p. 595) constructor.
- virtual bool **stopRequested** () const final  
*Determine if the parent has requested this child to exit.*
- bool **waitForMessage** (int numSeconds=-1) const  
*Block while waiting for a message from the Manager* (p. 430).

### G.148.1 Detailed Description

An abstraction of an instance that performs work on given data.

### G.148.2 Member Function Documentation

#### G.148.2.1 `_initCommunication()`

```
void BiometricEvaluation::Process::Worker::_initCommunication ()
```

Perform general communication initialization from Constructor.

Exceptions

|                                      |                                          |
|--------------------------------------|------------------------------------------|
| <i>Error::StrategyError</i> (p. 567) | <b>Error</b> (p. 108) in initialization. |
|--------------------------------------|------------------------------------------|

#### G.148.2.2 `closeManagerPipeEnds()`

```
void BiometricEvaluation::Process::Worker::closeManagerPipeEnds ()
```

Perform initialization for communication from **Manager** (p. 430) to **Worker** (p. 595).

Note

Behavior is undefined if called by a non-Worker.

Exceptions

|                                      |                             |
|--------------------------------------|-----------------------------|
| <i>Error::StrategyError</i> (p. 567) | Communications not enabled. |
|--------------------------------------|-----------------------------|

#### G.148.2.3 `closeWorkerPipeEnds()`

```
void BiometricEvaluation::Process::Worker::closeWorkerPipeEnds ()
```

Perform initialization for communication from **Worker** (p. 595) to **Manager** (p. 430).

Note

Behavior is undefined if called by a non-Manager.

Exceptions

|                                      |                             |
|--------------------------------------|-----------------------------|
| <i>Error::StrategyError</i> (p. 567) | Communications not enabled. |
|--------------------------------------|-----------------------------|

G.148.2.4 `getParameter()`

```
std::shared_ptr<void> BiometricEvaluation::Process::Worker::getParameter (
 const std::string & name)
```

Obtain a parameter passed to this **Worker** (p. 595).

Parameters

|             |                                 |
|-------------|---------------------------------|
| <i>name</i> | The parameter name to retrieve. |
|-------------|---------------------------------|

Returns

shared\_ptr to the parameter argument.

Exceptions

|                          |                   |
|--------------------------|-------------------|
| <i>std::out_of_range</i> | name was not set. |
|--------------------------|-------------------|

G.148.2.5 `getParameterAsDouble()`

```
double BiometricEvaluation::Process::Worker::getParameterAsDouble (
 const std::string & name)
```

Obtain a parameter passed to this **Worker** (p. 595) as a double.

Parameters

|             |                                 |
|-------------|---------------------------------|
| <i>name</i> | The parameter name to retrieve. |
|-------------|---------------------------------|

Returns

Parameter as a double.

Exceptions

|                          |                   |
|--------------------------|-------------------|
| <i>std::out_of_range</i> | name was not set. |
|--------------------------|-------------------|

G.148.2.6 `getParameterAsInteger()`

```
int64_t BiometricEvaluation::Process::Worker::getParameterAsInteger (
```

```
const std::string & name)
```

Obtain a parameter passed to this **Worker** (p. 595) as an integer.

Parameters

|             |                                 |
|-------------|---------------------------------|
| <i>name</i> | The parameter name to retrieve. |
|-------------|---------------------------------|

Returns

Parameter as an integer.

Exceptions

|                          |                   |
|--------------------------|-------------------|
| <i>std::out_of_range</i> | name was not set. |
|--------------------------|-------------------|

G.148.2.7 **getParameterAsString()**

```
std::string BiometricEvaluation::Process::Worker::getParameterAsString (
 const std::string & name)
```

Obtain a parameter passed to this **Worker** (p. 595) as a string.

Parameters

|             |                                 |
|-------------|---------------------------------|
| <i>name</i> | The parameter name to retrieve. |
|-------------|---------------------------------|

Returns

Parameter as a string.

Exceptions

|                          |                   |
|--------------------------|-------------------|
| <i>std::out_of_range</i> | name was not set. |
|--------------------------|-------------------|

G.148.2.8 **getReceivingPipe()**

```
int BiometricEvaluation::Process::Worker::getReceivingPipe () const
```

Obtain the pipe used to receive messages to this **Worker** (p. 595).

Returns

Receiving pipe.

Exceptions

|                                           |                                                              |
|-------------------------------------------|--------------------------------------------------------------|
| <i>Error::ObjectDoesNotExist</i> (p. 457) | <b>Worker</b> (p. 595) exiting soon, communication disabled. |
| <i>Error::StrategyError</i> (p. 567)      | Communications not enabled.                                  |



### G.148.2.9 getSendingPipe()

```
int BiometricEvaluation::Process::Worker::getSendingPipe () const
```

Obtain the pipe used to send messages to this **Worker** (p. 595).

Returns

Sending pipe.

Exceptions

|                                           |                                                              |
|-------------------------------------------|--------------------------------------------------------------|
| <i>Error::ObjectDoesNotExist</i> (p. 457) | <b>Worker</b> (p. 595) exiting soon, communication disabled. |
| <i>Error::StrategyError</i> (p. 567)      | Communications not enabled.                                  |

### G.148.2.10 receiveMessageFromManager()

```
void BiometricEvaluation::Process::Worker::receiveMessageFromManager (
```

```
 Memory::uint8Array & message)
```

Receive a message from the **Manager** (p. 430).

Parameters

|     |                |                                       |
|-----|----------------|---------------------------------------|
| out | <i>message</i> | Buffer to store the received message. |
|-----|----------------|---------------------------------------|

Exceptions

|                                           |                             |
|-------------------------------------------|-----------------------------|
| <i>Error::ObjectDoesNotExist</i> (p. 457) | Widowed pipe.               |
| <i>Error::StrategyError</i> (p. 567)      | Communications not enabled. |

See also

**waitForMessage** (p. 600)

### G.148.2.11 sendMessageToManager()

```
void BiometricEvaluation::Process::Worker::sendMessageToManager (
```

```
 const Memory::uint8Array & message)
```

Send a message to the **Manager** (p. 430).

Parameters

|    |                |                  |
|----|----------------|------------------|
| in | <i>message</i> | Message to send. |
|----|----------------|------------------|

## Exceptions

|                                                            |                             |
|------------------------------------------------------------|-----------------------------|
| <b>Error::ObjectDoesNotExist</b> (p. <a href="#">457</a> ) | Widowed pipe.               |
| <b>Error::StrategyError</b> (p. <a href="#">567</a> )      | Communications not enabled. |

**G.148.2.12 setParameter()**

```
void BiometricEvaluation::Process::Worker::setParameter (
 const std::string & name,
 std::shared_ptr< void > argument)
```

Pass a parameter to this **Worker** (p. [595](#)).

## Parameters

|                 |                                        |
|-----------------|----------------------------------------|
| <i>name</i>     | A unique identifier for this parameter |
| <i>argument</i> | A shared_ptr to the object to store.   |

**G.148.2.13 stopRequested()**

```
virtual bool BiometricEvaluation::Process::Worker::stopRequested () const [final], [protected],
[virtual]
```

Determine if the parent has requested this child to exit.

## Returns

Whether or not this child should exit.

**G.148.2.14 waitForMessage()**

```
bool BiometricEvaluation::Process::Worker::waitForMessage (
 int numSeconds = -1) const [protected]
```

Block while waiting for a message from the **Manager** (p. [430](#)).

## Parameters

|                   |                                                                            |
|-------------------|----------------------------------------------------------------------------|
| <i>numSeconds</i> | Number of seconds to wait for a message, or any value < 0 to wait forever. |
|-------------------|----------------------------------------------------------------------------|

## Returns

true once a message is ready to be read or false if an error occurred.

**G.148.2.15 workerMain()**

```
virtual int32_t BiometricEvaluation::Process::Worker::workerMain () [pure virtual]
```

The method that will get called to start execution by a ProcessManager.

## Returns

Status code.

## Note

If an object of this class is added to a **Process::ForkManager** (p. 335) object, the implementation of **Process::Worker::workerMain()** (p. 600) should release all resources prior to returning. Any exceptions thrown by this method will cause the worker to exit with a return status of EXIT\_FAILURE. The type and contents of the exception is not maintained.

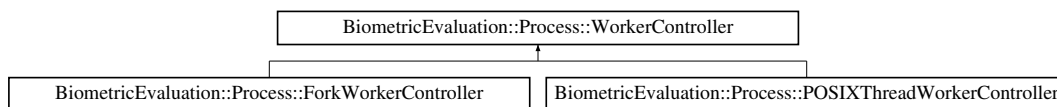
Implemented in **BiometricEvaluation::Process::MessageCenterReceiver** (p. 441), and **BiometricEvaluation::Process::MessageCenterListener** (p. 439).

## G.149 BiometricEvaluation::Process::WorkerController Class Reference

Wrapper of a **Worker** (p. 595) returned from a **Process::Manager** (p. 430).

```
#include <be_process_workercontroller.h>
```

Inheritance diagram for BiometricEvaluation::Process::WorkerController:



### Public Member Functions

- **WorkerController** (std::shared\_ptr< **Worker** > worker)
- virtual void **sendMessageToWorker** (const **Memory::uint8Array** &message)
 

*Send a message to the **Worker** (p. 595) contained within this **WorkerController** (p. 601).*
- virtual void **setParameter** (const std::string &name, std::shared\_ptr< void > argument)
 

*Set the parameter to be passed to the **Worker** (p. 595).*
- virtual void **setParameterFromDouble** (const std::string &name, double argument)
 

*Set a double parameter to be passed to the **Worker** (p. 595).*
- virtual void **setParameterFromInteger** (const std::string &name, int64\_t argument)
 

*Set an integer parameter to be passed to the **Worker** (p. 595).*
- virtual void **setParameterFromString** (const std::string &name, const std::string &argument)
 

*Set a string parameter to be passed to the **Worker** (p. 595).*
- virtual void **reset** ()
 

*Reuse the **Worker** (p. 595).*
- virtual bool **isWorking** () const =0
 

*Obtain whether or not **Worker** (p. 595) is working.*
- virtual bool **everWorked** () const =0
 

*Obtain whether or not this **Worker** (p. 595) has ever worked.*
- bool **finishedWorking** () const
 

*Obtain whether or not this **Worker** (p. 595) has both started and finished its task.*
- std::shared\_ptr< **Worker** > **getWorker** () const
 

*Obtain the **Worker** (p. 595) instance being wrapped.*

- virtual int32\_t **getExitStatus** () const final  
*Obtain the exit status of the wrapped **Worker** (p. 595).*
- virtual ~**WorkerController** ()  
***WorkerController** (p. 601) destructor.*

## Protected Attributes

- std::shared\_ptr< **Worker** > **\_worker**
- bool **\_rvSet**
- int32\_t **\_rv**

## G.149.1 Detailed Description

Wrapper of a **Worker** (p. 595) returned from a **Process::Manager** (p. 430).

## G.149.2 Constructor & Destructor Documentation

### G.149.2.1 WorkerController()

```
BiometricEvaluation::Process::WorkerController::WorkerController (
 std::shared_ptr< Worker > worker)
WorkerController (p. 601) constructor.
```

Parameters

|               |                                              |
|---------------|----------------------------------------------|
| <i>worker</i> | The <b>Worker</b> (p. 595) instance to wrap. |
|---------------|----------------------------------------------|

## G.149.3 Member Function Documentation

### G.149.3.1 everWorked()

```
virtual bool BiometricEvaluation::Process::WorkerController::everWorked () const [pure virtual]
 Obtain whether or not this Worker (p. 595) has ever worked.
```

Returns

true the **Worker** (p. 595) has ever or is currently working, false otherwise.

Note

**reset**() (p. 603) will change the result of this method.

Implemented in **BiometricEvaluation::Process::ForkWorkerController** (p. 342), and **BiometricEvaluation::Process::POSIXThreadWorkerController** (p. 484).

**G.149.3.2 finishedWorking()**

```
bool BiometricEvaluation::Process::WorkerController::finishedWorking () const [inline]
```

Obtain whether or not this **Worker** (p. 595) has both started and finished its task.

Returns

true if the **Worker** (p. 595) has both started and finished performing its task, false otherwise.

Note

**reset()** (p. 603) will change the result of this method.

**G.149.3.3 getExitStatus()**

```
virtual int32_t BiometricEvaluation::Process::WorkerController::getExitStatus () const [final],
[virtual]
```

Obtain the exit status of the wrapped **Worker** (p. 595).

Returns

Exit status of the wrapped **Worker** (p. 595).

Exceptions

|                                                  |                                                                                                                     |
|--------------------------------------------------|---------------------------------------------------------------------------------------------------------------------|
| <i><b>Error::ObjectDoesNotExist</b></i> (p. 457) | Exit status not set.                                                                                                |
| <i><b>Error::StrategyError</b></i> (p. 567)      | Exit status not set (e.g., <b>Worker</b> (p. 595) has not been started or <b>Worker</b> (p. 595) has not finished). |

**G.149.3.4 getWorker()**

```
std::shared_ptr< Worker> BiometricEvaluation::Process::WorkerController::getWorker () const
```

Obtain the **Worker** (p. 595) instance being wrapped.

Returns

**Worker** (p. 595) instance.

**G.149.3.5 isWorking()**

```
virtual bool BiometricEvaluation::Process::WorkerController::isWorking () const [pure virtual]
```

Obtain whether or not **Worker** (p. 595) is working.

Returns

Whether or not the **Worker** (p. 595) is working.

Implemented in **BiometricEvaluation::Process::ForkWorkerController** (p. 343), and **BiometricEvaluation::Process::POSIXThreadWorkerController** (p. 484).

**G.149.3.6 reset()**

```
virtual void BiometricEvaluation::Process::WorkerController::reset () [virtual]
```

Reuse the **Worker** (p. 595).

Exceptions

|                                            |                                                                 |
|--------------------------------------------|-----------------------------------------------------------------|
| <i><b>Error::ObjectExists</b></i> (p. 458) | The previously started <b>Worker</b> (p. 595) is still running. |
|--------------------------------------------|-----------------------------------------------------------------|

Reimplemented in **BiometricEvaluation::Process::ForkWorkerController** (p. 343), and **BiometricEvaluation::Process::POSIXThreadWorkerController** (p. 484).

**G.149.3.7 sendMessageToWorker()**

```
virtual void BiometricEvaluation::Process::WorkerController::sendMessageToWorker (
 const Memory::uint8Array & message) [virtual]
```

Send a message to the **Worker** (p. 595) contained within this **WorkerController** (p. 601).

Parameters

|                |                                                |
|----------------|------------------------------------------------|
| <i>message</i> | Message to send to the <b>Worker</b> (p. 595). |
|----------------|------------------------------------------------|

Exceptions

|                                                  |                                                                                                  |
|--------------------------------------------------|--------------------------------------------------------------------------------------------------|
| <i><b>Error::ObjectDoesNotExist</b></i> (p. 457) | <b>Worker</b> (p. 595) receive pipe is closed ( <b>Worker</b> (p. 595) object likely destroyed). |
| <i><b>Error::StrategyError</b></i> (p. 567)      | Message sending failed.                                                                          |

**G.149.3.8 setParameter()**

```
virtual void BiometricEvaluation::Process::WorkerController::setParameter (
 const std::string & name,
 std::shared_ptr< void > argument) [virtual]
```

Set the parameter to be passed to the **Worker** (p. 595).

Parameters

|    |                 |                                                                   |
|----|-----------------|-------------------------------------------------------------------|
| in | <i>name</i>     | The name representing the argument in the <b>Worker</b> (p. 595). |
| in | <i>argument</i> | The argument to be passed to the <b>Worker</b> (p. 595).          |

Note

Subsequent calls to **setParameter()** (p. 604) with the same name will overwrite any exiting argument.

### G.149.3.9 setParameterFromDouble()

```
virtual void BiometricEvaluation::Process::WorkerController::setParameterFromDouble (
 const std::string & name,
 double argument) [virtual]
```

Set a double parameter to be passed to the **Worker** (p. 595).

Parameters

|    |                 |                                                                   |
|----|-----------------|-------------------------------------------------------------------|
| in | <i>name</i>     | The name representing the argument in the <b>Worker</b> (p. 595). |
| in | <i>argument</i> | The double to be passed to the <b>Worker</b> (p. 595).            |

Note

Subsequent calls to setParameter\*() with the same name will overwrite any exiting argument.

### G.149.3.10 setParameterFromInteger()

```
virtual void BiometricEvaluation::Process::WorkerController::setParameterFromInteger (
 const std::string & name,
 int64_t argument) [virtual]
```

Set an integer parameter to be passed to the **Worker** (p. 595).

Parameters

|    |                 |                                                                   |
|----|-----------------|-------------------------------------------------------------------|
| in | <i>name</i>     | The name representing the argument in the <b>Worker</b> (p. 595). |
| in | <i>argument</i> | The integer to be passed to the <b>Worker</b> (p. 595).           |

Note

Subsequent calls to setParameter\*() with the same name will overwrite any exiting argument.

### G.149.3.11 setParameterFromString()

```
virtual void BiometricEvaluation::Process::WorkerController::setParameterFromString (
 const std::string & name,
 const std::string & argument) [virtual]
```

Set a string parameter to be passed to the **Worker** (p. 595).

Parameters

|    |                 |                                                                   |
|----|-----------------|-------------------------------------------------------------------|
| in | <i>name</i>     | The name representing the argument in the <b>Worker</b> (p. 595). |
| in | <i>argument</i> | The string to be passed to the <b>Worker</b> (p. 595).            |

## Note

Subsequent calls to `setParameter*()` with the same name will overwrite any exiting argument.

## G.149.4 Member Data Documentation

### G.149.4.1 `_rv`

```
int32_t BiometricEvaluation::Process::WorkerController::_rv [protected]
 Exit status from _worker.workerMain()
```

### G.149.4.2 `_rvSet`

```
bool BiometricEvaluation::Process::WorkerController::_rvSet [protected]
 Whether or not _rv contains a true value.
```

### G.149.4.3 `_worker`

```
std::shared_ptr< Worker> BiometricEvaluation::Process::WorkerController::_worker [protected]
 The Worker (p. 595) instance that is running in this child
```

## G.150 BiometricEvaluation::MPI::WorkPackage Class Reference

A class to represent a piece of work to be acted upon by a processor.

```
#include <be_mpi_workpackage.h>
```

### Public Member Functions

- **WorkPackage** ()  
*Construct an empty work package.*
- **WorkPackage** (const **Memory::uint8Array** &data)  
*Construct a work package with some data.*
- void **getData** ( **Memory::uint8Array** &data) const  
*Obtain the package data in raw form.*
- void **setData** (const **Memory::uint8Array** &data)  
*Set the package data from raw data.*
- uint64\_t **getSize** () const  
*Obtain the size of the package data.*
- uint64\_t **getNumElements** () const  
*Obtain the number of elements in the package.*
- void **setNumElements** (const uint64\_t numElements)  
*Set the number of elements in the package.*

### G.150.1 Detailed Description

A class to represent a piece of work to be acted upon by a processor.

The work package is an wrapper around the data to be processed, along with some ancillary information.



## G.150.2 Constructor & Destructor Documentation

### G.150.2.1 WorkPackage()

```
BiometricEvaluation::MPI::WorkPackage::WorkPackage (
 const Memory::uint8Array & data)
```

Construct a work package with some data.

Parameters

|    |             |                                                     |
|----|-------------|-----------------------------------------------------|
| in | <i>data</i> | The data that will be managed by this work package. |
|----|-------------|-----------------------------------------------------|

## G.150.3 Member Function Documentation

### G.150.3.1 getNumElements()

```
uint64_t BiometricEvaluation::MPI::WorkPackage::getNumElements () const
```

Obtain the number of elements in the package.

This value is determined by the application and must be set therein, otherwise 0 is returned.

Returns

The number of application defined elements in the work package.

### G.150.3.2 getSize()

```
uint64_t BiometricEvaluation::MPI::WorkPackage::getSize () const
```

Obtain the size of the package data.

Returns

The size (in octets) of the raw data item.

### G.150.3.3 setData()

```
void BiometricEvaluation::MPI::WorkPackage::setData (
 const Memory::uint8Array & data)
```

Set the package data from raw data.

Parameters

|    |             |                                        |
|----|-------------|----------------------------------------|
| in | <i>data</i> | The data copied into the work package. |
|----|-------------|----------------------------------------|

### G.150.3.4 setNumElements()

```
void BiometricEvaluation::MPI::WorkPackage::setNumElements (
 const uint64_t numElements)
```

Set the number of elements in the package.

Parameters

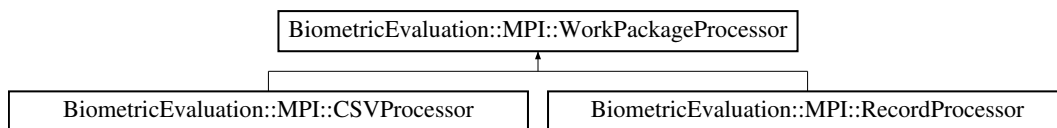
|    |                    |                                                                 |
|----|--------------------|-----------------------------------------------------------------|
| in | <i>numElements</i> | The number of application-defined elements in the work package. |
|----|--------------------|-----------------------------------------------------------------|

## G.151 BiometricEvaluation::MPI::WorkPackageProcessor Class Reference

Represents an object that processes the contents of a work package.

```
#include <be_mpi_workpackageprocessor.h>
```

Inheritance diagram for BiometricEvaluation::MPI::WorkPackageProcessor:



### Public Member Functions

- virtual std::shared\_ptr< **WorkPackageProcessor** > **newProcessor** (std::shared\_ptr< **IO::Logsheet** > &logsheet)=0  
*Obtain an object that will process work packages. This method is part of the factory personality.*
- virtual void **performInitialization** (std::shared\_ptr< **IO::Logsheet** > &logsheet)=0  
*Initialization function to be called before work is distributed to the work package processor.*
- virtual void **processWorkPackage** ( **MPI::WorkPackage** &workPackage)=0  
*Process (p. 150) the data contents of the work package. This method is part of the worker personality.*
- virtual void **performShutdown** ()  
*Termination function to be called during shut down after all work package processing is done.*
- void **setLogsheet** (std::shared\_ptr< **IO::Logsheet** > &logsheet)  
*Set the **IO::Logsheet** (p. 420) object that can be used to save message for objects of this class.*
- std::shared\_ptr< **IO::Logsheet** > **getLogsheet** ()  
*Obtain the **IO::Logsheet** (p. 420) object that can be used to save message for objects of this class.*

### G.151.1 Detailed Description

Represents an object that processes the contents of a work package.

A **WorkPackageProcessor** (p. 608) presents two personalities: One that of a worker to process work packages, and one that is a factory to return worker objects of the implementation class.

Subclasses of this class implement the functionality needed to perform an action on the work package data. The processing done by the implementation is application and data type specific.

Ultimately, the final implementation of the **WorkPackageProcessor** (p. 608) class is done in the application. Access to the Logsheet object maintained by the framework is provided by this class.

## G.151.2 Member Function Documentation

### G.151.2.1 getLogsheet()

```
std::shared_ptr< IO::Logsheet> BiometricEvaluation::MPI::WorkPackageProcessor::getLogsheet (
)
```

Obtain the **IO::Logsheet** (p. 420) object that can be used to save message for objects of this class.

Returns

logsheet A shared pointer to the Logsheet object.

### G.151.2.2 newProcessor()

```
virtual std::shared_ptr< WorkPackageProcessor> BiometricEvaluation::MPI::WorkPackageProcessor←
::newProcessor (
 std::shared_ptr< IO::Logsheet > & logsheet) [pure virtual]
```

Obtain an object that will process work packages. This method is part of the factory personality.

Parameters

|                 |                                                                                                                 |
|-----------------|-----------------------------------------------------------------------------------------------------------------|
| <i>logsheet</i> | A shared pointer to the <b>IO::Logsheet</b> (p. 420) that may be used to save messages generated by the object. |
|-----------------|-----------------------------------------------------------------------------------------------------------------|

Returns

A shared pointer to the work package processor.

Note

This method should always create a non-null **WorkPackageProcessor** (p. 608). If an error occurs during construction, throw a **Error::Exception** (p. 310) with a message to be caught and logged.

Implemented in **BiometricEvaluation::MPI::CSVProcessor** (p. 289), and **BiometricEvaluation::MPI::RecordProcessor** (p. 502).

### G.151.2.3 performInitialization()

```
virtual void BiometricEvaluation::MPI::WorkPackageProcessor::performInitialization (
 std::shared_ptr< IO::Logsheet > & logsheet) [pure virtual]
```

Initialization function to be called before work is distributed to the work package processor.

Implementations of this class can use this function to do any processing necessary before work is given to the processor, pre-forking.

This method is part of the factory personality. All state that is to be common across all package processor objects can be initialized in this method.

## Parameters

|                 |                                                                                                                 |
|-----------------|-----------------------------------------------------------------------------------------------------------------|
| <i>logsheet</i> | A shared pointer to the <b>IO::Logsheet</b> (p. 420) that may be used to save messages generated by the object. |
|-----------------|-----------------------------------------------------------------------------------------------------------------|

## Exceptions

|                                  |                                                                                                                  |
|----------------------------------|------------------------------------------------------------------------------------------------------------------|
| <b>Error::Exception</b> (p. 310) | An implementation specific error occurred. The exception string will be logged by the <b>Framework</b> (p. 117). |
|----------------------------------|------------------------------------------------------------------------------------------------------------------|

Implemented in **BiometricEvaluation::MPI::CSVProcessor** (p. 290), and **BiometricEvaluation::MPI::RecordProcessor** (p. 502).

**G.151.2.4 performShutdown()**

```
virtual void BiometricEvaluation::MPI::WorkPackageProcessor::performShutdown () [virtual]
```

Termination function to be called during shut down after all work package processing is done.

Implementations of this class can use this function to do any processing necessary after all work is given to the processors. The default implementation does nothing.

This method is part of the factory personality. All state that is created in **performInitialization()** (p. 609) processor objects can be accessed in this method.

## Exceptions

|                                  |                                                                                                                  |
|----------------------------------|------------------------------------------------------------------------------------------------------------------|
| <b>Error::Exception</b> (p. 310) | An implementation specific error occurred. The exception string will be logged by the <b>Framework</b> (p. 117). |
|----------------------------------|------------------------------------------------------------------------------------------------------------------|

**G.151.2.5 processWorkPackage()**

```
virtual void BiometricEvaluation::MPI::WorkPackageProcessor::processWorkPackage (
 MPI::WorkPackage & workPackage) [pure virtual]
```

**Process** (p. 150) the data contents of the work package. This method is part of the worker personality.

## Parameters

|    |                    |                   |
|----|--------------------|-------------------|
| in | <i>workPackage</i> | The work package. |
|----|--------------------|-------------------|

## Exceptions

|                                  |                                                                                                                        |
|----------------------------------|------------------------------------------------------------------------------------------------------------------------|
| <b>Error::Exception</b> (p. 310) | An fatal error occurred when processing the work package; the processing responsible for this object should shut down. |
|----------------------------------|------------------------------------------------------------------------------------------------------------------------|

Implemented in **BiometricEvaluation::MPI::CSVProcessor** (p. 291), and **BiometricEvaluation::MPI::RecordProcessor** (p. 503).

### G.151.2.6 setLogsheet()

```
void BiometricEvaluation::MPI::WorkPackageProcessor::setLogsheet (
 std::shared_ptr< IO::Logsheet > & logsheet)
```

Set the **IO::Logsheet** (p. 420) object that can be used to save message for objects of this class.

Parameters

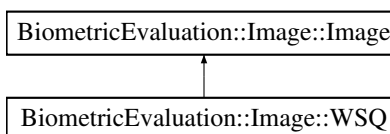
|    |                 |                                          |
|----|-----------------|------------------------------------------|
| in | <i>logsheet</i> | A shared pointer to the Logsheet object. |
|----|-----------------|------------------------------------------|

## G.152 BiometricEvaluation::Image::WSQ Class Reference

A WSQ-encoded image.

```
#include <be_image_wsq.h>
```

Inheritance diagram for BiometricEvaluation::Image::WSQ:



### Public Member Functions

- **WSQ** (const uint8\_t \*data, const uint64\_t size)
- **WSQ** (const **Memory::uint8Array** &data)
- **Memory::uint8Array** **getRawData** () const  
*Accessor for the raw image data. The data returned should not be compressed or encoded.*
- **Memory::uint8Array** **getRawGrayscaleData** (uint8\_t depth) const  
*Accessor for decompressed data in grayscale.*

### Static Public Member Functions

- static bool **isWSQ** (const uint8\_t \*data, uint64\_t size)

### Additional Inherited Members

#### G.152.1 Detailed Description

A WSQ-encoded image.

#### G.152.2 Member Function Documentation

**G.152.2.1 getRawData()**

**Memory::uint8Array** BiometricEvaluation::Image::WSQ::getRawData ( ) const [virtual]

Accessor for the raw image data. The data returned should not be compressed or encoded.

Returns

AutoArray holding raw image data.

Exceptions

|                                  |                                                 |
|----------------------------------|-------------------------------------------------|
| <b>Error::DataError</b> (p. 295) | <b>Error</b> (p. 108) decompressing image data. |
|----------------------------------|-------------------------------------------------|

Implements **BiometricEvaluation::Image::Image** (p. 361).

**G.152.2.2 getRawGrayscaleData()**

**Memory::uint8Array** BiometricEvaluation::Image::WSQ::getRawGrayscaleData (   
 uint8\_t depth ) const [virtual]

Accessor for decompressed data in grayscale.

Parameters

|              |                                                                                         |
|--------------|-----------------------------------------------------------------------------------------|
| <i>depth</i> | The desired bit depth of the resulting raw image. This value may either be 16, 8, or 1. |
|--------------|-----------------------------------------------------------------------------------------|

Returns

AutoArray holding raw grayscale image data.

Exceptions

|                                       |                                                     |
|---------------------------------------|-----------------------------------------------------|
| <b>Error::DataError</b> (p. 295)      | <b>Error</b> (p. 108) decompressing image data.     |
| <b>Error::NotImplemented</b> (p. 456) | Unsupported conversion based on source color depth. |
| <b>Error::ParameterError</b> (p. 474) | Invalid value for depth.                            |

Note

This method does not save a cached copy of the decompressed image because the bit depth of the image can be changed between calls.

When depth is 1, this method returns an image that uses 8 bits to represent a single pixel. The depth parameter is used to adjust the number of gray levels. When depth is 1, there are only 2 gray levels (black and white), despite using 8 bits to represent each pixel.

Alpha channels are completely ignored when converting to grayscale.

Implements **BiometricEvaluation::Image::Image** (p. 362).

**G.152.2.3 isWSQ()**

static bool BiometricEvaluation::Image::WSQ::isWSQ (

```
const uint8_t * data,
uint64_t size) [static]
```

Whether or not data is a **WSQ** (p. 611) image.

Parameters

|    |             |                      |
|----|-------------|----------------------|
| in | <i>data</i> | The buffer to check. |
| in | <i>size</i> | The size of data.    |

Returns

true if data appears to be a **WSQ** (p. 611) image, false otherwise

## G.153 BiometricEvaluation::Feature::Sort::XY Class Reference

```
#include <be_feature_sort.h>
```

### Public Member Functions

- **bool operator()** (const **BiometricEvaluation::Feature::MinutiaPoint** &lhs, const **BiometricEvaluation::Feature::MinutiaPoint** &rhs) const  
*MinutiaPoint* (p. 444) Cartesian X-Y ascending comparator.

#### G.153.1 Detailed Description

**Sort** (p. 113) by increasing Cartesian X-Y coordinate

## G.154 BiometricEvaluation::Feature::Sort::YX Class Reference

```
#include <be_feature_sort.h>
```

### Public Member Functions

- **bool operator()** (const **BiometricEvaluation::Feature::MinutiaPoint** &lhs, const **BiometricEvaluation::Feature::MinutiaPoint** &rhs) const  
*MinutiaPoint* (p. 444) Cartesian Y-X ascending comparator.

#### G.154.1 Detailed Description

**Sort** (p. 113) by increasing Cartesian Y-X coordinate





# Index

- `_canSigJump`
  - BiometricEvaluation::Error::SignalManager, 548
- `_cursor`
  - BiometricEvaluation::IO::FileLogsheet, 325
- `_initCommunication`
  - BiometricEvaluation::Process::Worker, 596
- `_operational`
  - BiometricEvaluation::IO::SysLogsheet, 574
- `_pendingExit`
  - BiometricEvaluation::Process::Manager, 435
- `_rv`
  - BiometricEvaluation::Process::WorkerController, 606
- `_rvSet`
  - BiometricEvaluation::Process::WorkerController, 606
- `_sequenceFile`
  - BiometricEvaluation::IO::FileLogsheet, 325
- `_sequenced`
  - BiometricEvaluation::IO::SysLogsheet, 574
- `_sigJumpBuf`
  - BiometricEvaluation::Error::SignalManager, 548
- `_sockFD`
  - BiometricEvaluation::IO::SysLogsheet, 574
- `_stop`
  - BiometricEvaluation::Process::ForkWorkerController, 342
- `_theLogFile`
  - BiometricEvaluation::IO::FileLogsheet, 325
- `_utc`
  - BiometricEvaluation::IO::SysLogsheet, 574
- `_worker`
  - BiometricEvaluation::Process::WorkerController, 606
- `_workers`
  - BiometricEvaluation::Process::Manager, 435
- `~ArchiveRecordStore`
  - BiometricEvaluation::IO::ArchiveRecordStore, 226
- `~AutoArray`
  - BiometricEvaluation::Memory::AutoArray, 237
- `~AutoArrayIterator`
  - BiometricEvaluation::Memory::AutoArrayIterator, 246
- `~CommandCenter`
  - BiometricEvaluation::Process::CommandCenter, 257
- `~CommandParser`
  - BiometricEvaluation::Process::CommandParser, 260
- `~Compressor`
  - BiometricEvaluation::IO::Compressor, 272
- `~Exception`
  - BiometricEvaluation::MPI::Exception, 313
- `~FileLogsheet`
  - BiometricEvaluation::IO::FileLogsheet, 321
- `~IndexedBuffer`
  - BiometricEvaluation::Memory::IndexedBuffer, 397
- `~ListRecordStore`
  - BiometricEvaluation::IO::ListRecordStore, 414
- `~Logsheet`
  - BiometricEvaluation::IO::Logsheet, 423
- `~MessageCenterReceiver`
  - BiometricEvaluation::Process::MessageCenterReceiver, 441
- `~MutableIndexedBuffer`
  - BiometricEvaluation::Memory::MutableIndexedBuffer, 447
- `~OrderedMap`
  - BiometricEvaluation::Memory::OrderedMap, 462
- `~OrderedMapConstIterator`
  - BiometricEvaluation::Memory::OrderedMapConstIterator, 467
- `~OrderedMapIterator`
  - BiometricEvaluation::Memory::OrderedMapIterator, 471
- `~PersistentRecordStoreUnion`
  - BiometricEvaluation::IO::PersistentRecordStoreUnion, 476
- `~Properties`
  - BiometricEvaluation::IO::Properties, 488
- `~PropertiesFile`
  - BiometricEvaluation::IO::PropertiesFile, 495
- `~RecordStoreIterator`
  - BiometricEvaluation::IO::RecordStoreIterator, 522
- `~RecordStoreUnion`
  - BiometricEvaluation::IO::RecordStoreUnion, 529
- `~Smartcard`

- BiometricEvaluation::Device::Smartcard, 551
- ~SysLogsheet
  - BiometricEvaluation::IO::SysLogsheet, 572
- AN2K7Minutiae
  - BiometricEvaluation::Feature::AN2K7Minutiae, 169
- AN2KMinutiaeDataRecord
  - BiometricEvaluation::Finger::AN2KMinutiaeData↔
    - Record, 172
- AN2KRecord
  - BiometricEvaluation::DataInterchange::AN2K↔
    - Record, 176, 177
- AN2KView
  - BiometricEvaluation::Finger::AN2KView, 190
  - BiometricEvaluation::Latent::AN2KView, 187
  - BiometricEvaluation::Palm::AN2KView, 182
  - BiometricEvaluation::View::AN2KView, 185
- AN2KViewCapture
  - BiometricEvaluation::Finger::AN2KViewCapture, 195
- AN2KViewFixedResolution
  - BiometricEvaluation::Finger::AN2KViewFixed↔
    - Resolution, 198, 199
- AN2KViewVariableResolution
  - BiometricEvaluation::View::AN2KViewVariable↔
    - Resolution, 201
- ANSI2004Record
  - BiometricEvaluation::DataInterchange::ANSI2004↔
    - Record, 206, 207
- ANSI2004View
  - BiometricEvaluation::Finger::ANSI2004View, 213
- ANSI2007View
  - BiometricEvaluation::Finger::ANSI2007View, 215
- APDUException
  - BiometricEvaluation::Device::Smartcard::APD↔
    - UException, 219
- APDUResponse
  - BiometricEvaluation::Device::Smartcard::APD↔
    - UResponse, 220
- APICurrentState
  - BiometricEvaluation::Framework, 118
- API
  - BiometricEvaluation::Framework::API, 222
- ARCHIVE\_FILE\_NAME
  - BiometricEvaluation::IO::ArchiveRecordStore, 233
- ASCIIBitmapTo8Bit
  - BiometricEvaluation::Image::NetPBM, 452
- ASCIIPixmapToBinaryPixmap
  - BiometricEvaluation::Image::NetPBM, 452
- abort
  - BiometricEvaluation::MPI::Runtime, 541
- addChild
  - BiometricEvaluation::Device::TLV, 583
- addMinutiaeDataRecord
  - BiometricEvaluation::Finger::AN2KView, 191
- addWorker
  - BiometricEvaluation::Process::ForkManager, 336
  - BiometricEvaluation::Process::Manager, 431
  - BiometricEvaluation::Process::POSIXThreadManager, 481
- AmputatedBandaged
  - BiometricEvaluation::Finger::AN2KViewCapture, 195
- apdu
  - BiometricEvaluation::Device::Smartcard::APD↔
    - UException, 219
- ArchiveRecordStore
  - BiometricEvaluation::IO::ArchiveRecordStore, 225
- arguments
  - BiometricEvaluation::Process::CommandCenter↔
    - ::Command, 256
- at
  - BiometricEvaluation::Memory::AutoArray, 237, 238
- AutoArray
  - BiometricEvaluation::Memory::AutoArray, 236, 237
- AutoArrayIterator
  - BiometricEvaluation::Memory::AutoArrayIterator, 245, 246
- BE\_CLOCK\_TYPE
  - BiometricEvaluation::Time::Timer, 579
- BE\_FILELOGSHEET\_SEQ\_NEXT
  - BiometricEvaluation::IO::FileLogsheet, 325
- BE\_FILELOGSHEET\_SEQ\_START
  - BiometricEvaluation::IO::FileLogsheet, 325
- BE\_RECSTORE\_SEQ\_NEXT
  - BiometricEvaluation::IO::RecordStore, 517
- BE\_RECSTORE\_SEQ\_START
  - BiometricEvaluation::IO::RecordStore, 517
- basename
  - BiometricEvaluation::Text, 155
- begin
  - BiometricEvaluation::IO::RecordStore, 506
  - BiometricEvaluation::Memory::AutoArray, 238
  - BiometricEvaluation::Memory::OrderedMap, 462
- BinaryBitmapTo8Bit
  - BiometricEvaluation::Image::NetPBM, 453
- BiometricEvaluation, 107
- BiometricEvaluation::DataInterchange::AN2KRecord, 175
- AN2KRecord, 176, 177
- CharacterSet, 176

- DomainName, [176](#)
- getDate, [177](#)
- getDestinationAgency, [177](#)
- getDirectoryOfCharacterSets, [177](#)
- getDomainName, [178](#)
- getFingerCaptureCount, [178](#)
- getFingerCaptures, [178](#)
- getFingerLatentCount, [178](#)
- getFingerLatents, [178](#)
- getGreenwichMeanTime, [179](#)
- getMinutiaeDataRecordSet, [179](#)
- getNativeScanningResolution, [179](#)
- getNominalTransmittingResolution, [179](#)
- getOriginatingAgency, [179](#)
- getPriority, [180](#)
- getTransactionControlNumber, [180](#)
- getVersionNumber, [180](#)
- recordLocations, [180](#), [181](#)
- BiometricEvaluation::DataInterchange::AN2KRecord↔
  - ::CharacterSet, [254](#)
  - CharacterSet, [254](#)
  - commonName, [254](#)
  - identifier, [255](#)
  - version, [255](#)
- BiometricEvaluation::DataInterchange::AN2KRecord↔ BiometricEvaluation::Device::TLV, [581](#)
  - ::DomainName, [308](#)
  - DomainName, [308](#)
  - identifier, [309](#)
  - version, [309](#)
- BiometricEvaluation::DataInterchange::ANSI2004Record, [205](#)
  - ANSI2004Record, [206](#), [207](#)
  - getEDBLength, [207](#)
  - getFMRLength, [208](#)
  - getFMR, [207](#)
  - getMinutia, [208](#)
  - getNumFingerViews, [208](#)
  - getView, [208](#)
  - insertView, [209](#)
  - isolateView, [210](#)
  - removeView, [210](#)
  - setMinutia, [210](#), [211](#)
  - updateView, [211](#)
- BiometricEvaluation::Device::Smartcard, [550](#)
  - ~Smartcard, [551](#)
  - getDedicatedFileObject, [552](#)
  - getLastAPDU, [552](#)
  - getLastResponseData, [552](#)
  - getReaderID, [553](#)
  - operator=, [553](#)
  - sendAPDU, [553](#)
  - setDryrun, [553](#)
  - Smartcard, [550](#), [551](#)
- BiometricEvaluation::Device::Smartcard::APDUException, [219](#)
  - APDUException, [219](#)
  - apdu, [219](#)
  - response, [220](#)
- BiometricEvaluation::Device::Smartcard::APDUResponse, [220](#)
  - APDUResponse, [220](#)
  - data, [221](#)
  - sw1, [221](#)
  - sw2, [221](#)
- BiometricEvaluation::Device::Smartcard::APDU, [216](#)
  - cla, [218](#)
  - FIELD\_LC, [218](#)
  - FIELD\_LE, [218](#)
  - field\_mask, [218](#)
  - ins, [218](#)
  - lc, [218](#)
  - le, [218](#)
  - nc, [218](#)
  - p1, [218](#)
  - p2, [218](#)
- addChild, [583](#)
- getChildren, [583](#)
- getPrimitive, [584](#)
- getRawTLV, [584](#)
- getTagClass, [584](#)
- getTagNum, [584](#)
- isPrimitive, [584](#)
- setPrimitive, [585](#)
- setTag, [585](#)
- stringFromTLV, [585](#)
- TLV, [582](#), [583](#)
- BiometricEvaluation::Error, [108](#)
  - errorStr, [109](#)
- BiometricEvaluation::Error::ConversionError, [283](#)
  - ConversionError, [284](#)
- BiometricEvaluation::Error::DataError, [295](#)
  - DataError, [295](#)
- BiometricEvaluation::Error::Exception, [310](#)
  - Exception, [311](#)
  - what, [312](#)
  - whatString, [312](#)
- BiometricEvaluation::Error::FileError, [316](#)
  - FileError, [317](#)
- BiometricEvaluation::Error::MemoryError, [436](#)
  - MemoryError, [436](#)
- BiometricEvaluation::Error::NotImplemented, [456](#)

- NotImplemented, [457](#)
- BiometricEvaluation::Error::ObjectDoesNotExist, [457](#)
  - ObjectDoesNotExist, [458](#)
- BiometricEvaluation::Error::ObjectExists, [458](#)
  - ObjectExists, [458](#), [459](#)
- BiometricEvaluation::Error::ObjectIsClosed, [459](#)
  - ObjectIsClosed, [459](#)
- BiometricEvaluation::Error::ObjectIsOpen, [460](#)
  - ObjectIsOpen, [460](#)
- BiometricEvaluation::Error::ParameterError, [474](#)
  - ParameterError, [474](#)
- BiometricEvaluation::Error::SignalManager, [545](#)
  - \_canSigJump, [548](#)
  - \_sigJumpBuf, [548](#)
  - clearSigHandled, [547](#)
  - clearSignalSet, [547](#)
  - setDefaultSignalSet, [547](#)
  - setSigHandled, [547](#)
  - setSignalSet, [547](#)
  - sigHandled, [547](#)
  - SignalManager, [546](#)
  - start, [548](#)
  - stop, [548](#)
- BiometricEvaluation::Error::StrategyError, [567](#)
  - StrategyError, [567](#)
- BiometricEvaluation::Face, [109](#)
  - PropertySet, [111](#)
- BiometricEvaluation::Face::INCITSView, [371](#)
  - getColorSpace, [373](#)
  - getDeviceType, [374](#)
  - getEyeColor, [374](#)
  - getFIDData, [374](#)
  - getFeaturePointSet, [374](#)
  - getGender, [374](#)
  - getHairColor, [375](#)
  - getImageDataType, [375](#)
  - getImageType, [375](#)
  - getPoseAngle, [375](#)
  - getPropertySet, [375](#)
  - getSourceType, [375](#)
  - INCITSView, [372](#), [373](#)
  - propertiesConsidered, [376](#)
  - readFaceView, [376](#)
  - readHeader, [376](#)
- BiometricEvaluation::Face::ISO2005View, [400](#)
  - ISO2005View, [401](#)
  - readISOHeader, [402](#)
- BiometricEvaluation::Face::PoseAngle, [480](#)
- BiometricEvaluation::Feature, [111](#)
  - operator<<, [113](#)
- BiometricEvaluation::Feature::AN2K11EFS::CorePoint, [285](#)
- BiometricEvaluation::Feature::AN2K11EFS::DeltaPoint, [304](#)
- BiometricEvaluation::Feature::AN2K11EFS::Extended↔  
FeatureSet, [313](#)
  - ExtendedFeatureSet, [314](#)
  - getCPS, [315](#)
  - getDPS, [315](#)
  - getImageInfo, [315](#)
  - getMPS, [315](#)
  - getMRCI, [316](#)
  - getNFP, [316](#)
- BiometricEvaluation::Feature::AN2K11EFS::FPPPosition, [345](#)
  - fgp, [346](#)
  - fsm, [346](#)
  - ocf, [346](#)
  - sgp, [346](#)
- BiometricEvaluation::Feature::AN2K11EFS::Image↔  
Info, [367](#)
  - fpp, [367](#)
  - ort, [367](#)
  - plr, [367](#)
  - roi, [367](#)
  - trv, [368](#)
- BiometricEvaluation::Feature::AN2K11EFS::Minutiae↔  
Point, [445](#)
  - mdu, [445](#)
  - mru, [445](#)
- BiometricEvaluation::Feature::AN2K11EFS::Minutiae↔  
RidgeCount, [442](#)
  - mia, [443](#)
  - mib, [443](#)
  - mir, [443](#)
  - mrn, [443](#)
  - mrs, [443](#)
- BiometricEvaluation::Feature::AN2K11EFS::Minutiae↔  
RidgeCountConfidence, [443](#)
- BiometricEvaluation::Feature::AN2K11EFS::Minutiae↔  
RidgeCountInfo, [444](#)
- BiometricEvaluation::Feature::AN2K11EFS::NoFeatures↔  
Present, [456](#)
- BiometricEvaluation::Feature::AN2K11EFS::Orientation, [473](#)
  - EOODDefault, [473](#)
  - EUCDefault, [473](#)
  - eod, [473](#)
  - euc, [473](#)
  - is\_default, [473](#)
- BiometricEvaluation::Feature::AN2K7Minutiae, [167](#)
- BiometricEvaluation::Feature::AN2K7Minutiae, [169](#)

- convertCoordinate, 169
- convertEncodingMethod, 170
- convertPatternClassification, 170, 171
- EncodingMethod, 168
- getOriginatingFingerprintReadingSystem, 171
- getPatternClassificationSet, 171
- BiometricEvaluation::Feature::AN2K7Minutiae::Fingerprint←
  - ReadingSystem, 333
- equipment, 333
- method, 334
- name, 334
- BiometricEvaluation::Feature::AN2K7Minutiae::Pattern↔
  - Classification, 474
- BiometricEvaluation::Feature::AN2K7Minutiae::Pattern↔
  - Classification::Entry, 309
- code, 310
- Entry, 309
- standard, 310
- BiometricEvaluation::Feature::CorePoint, 285
- BiometricEvaluation::Feature::DeltaPoint, 304
- BiometricEvaluation::Feature::FrictionRidgeGeneralized↔
  - Position, 346
- BiometricEvaluation::Feature::INCITSMinutiae, 368
  - INCITSMinutiae, 370
  - setCorePointSet, 370
  - setDeltaPointSet, 370
  - setMinutiaPoints, 370
  - setRidgeCountItems, 371
- BiometricEvaluation::Feature::MPEGFacePoint, 445
- BiometricEvaluation::Feature::MinutiaPoint, 444
- BiometricEvaluation::Feature::Minutiae, 442
- BiometricEvaluation::Feature::RidgeCountItem, 538
- BiometricEvaluation::Feature::Sort, 113
  - Kind, 114
  - sort, 114
  - stableSort, 115
- BiometricEvaluation::Feature::Sort::Angle, 205
  - operator(), 205
- BiometricEvaluation::Feature::Sort::Polar, 478
  - centerOfImage, 479
  - centerOfMinutiaeMass, 479
  - operator(), 480
  - Polar, 479
- BiometricEvaluation::Feature::Sort::Quality, 496
- BiometricEvaluation::Feature::Sort::XY, 613
- BiometricEvaluation::Feature::Sort::YX, 613
- BiometricEvaluation::Finger, 115
  - FingerImageCode, 117
  - Impression, 117
  - PatternClassification, 117
  - Position, 117
- BiometricEvaluation::Finger::AN2KMinutiaeDataRecord, 171
  - AN2KMinutiaeDataRecord, 172
  - getAN2K11EFS, 173
  - getAN2K7Minutiae, 173
  - getImpressionType, 173
  - getRegisteredVendorBlock, 174
- BiometricEvaluation::Finger::AN2KView, 188
  - AN2KView, 190
  - addMinutiaeDataRecord, 191
  - convertFingerImageCode, 191
  - convertPosition, 192
  - getImpressionType, 192
  - getMinutiaeDataRecordSet, 192
  - getPositions, 192
  - populateFGP, 192
  - setImpressionType, 193
  - setPositions, 193
- BiometricEvaluation::Finger::AN2KViewCapture, 193
  - AN2KViewCapture, 195
  - AmputatedBandaged, 195
  - extractNISTQuality, 196
  - getAlternateFingerSegmentPositionSet, 196
  - getAmputatedBandaged, 196
  - getFingerSegmentPositionSet, 197
  - getFingerprintQualityMetric, 196
  - getNISTQualityMetric, 197
  - getPosition, 197
  - getPrintPositionCoordinates, 197
  - getSegmentationQualityMetric, 197
- BiometricEvaluation::Finger::AN2KViewCapture::Finger↔
  - SegmentPosition, 334
- coordinates, 335
- fingerPosition, 335
- FingerSegmentPosition, 334
- BiometricEvaluation::Finger::AN2KViewFixedResolution, 198
  - AN2KViewFixedResolution, 198, 199
- BiometricEvaluation::Finger::ANSI2004View, 212
  - ANSI2004View, 213
  - readCoreDeltaData, 214
- BiometricEvaluation::Finger::ANSI2007View, 214
  - ANSI2007View, 215
  - readCoreDeltaData, 216
- BiometricEvaluation::Finger::INCITSView, 383
  - convertImpression, 387
  - convertPosition, 387
  - getCaptureEquipmentID, 387
  - getEDBLength, 388
  - getFIRData, 388
  - getFMRData, 388

- getFMRReservedByte, 388
- getImpressionType, 388
- getMinutiaeReservedData, 388
- getNumFingerViews, 389
- getPosition, 389
- getProductIDOwner, 389
- getProductIDType, 389
- getQuality, 389
- getRecordLength, 390
- getViewNumber, 390
- INCITSView, 386
- isAppendixFCompliant, 390
- readCoreDeltaData, 390
- readExtendedDataBlock, 391
- readFMRHeader, 391
- readFVMR, 391
- readMinutiaeDataPoints, 392
- readRidgeCountData, 392
- setAppendixFCompliance, 393
- setCBEFFProductIDs, 393
- setCaptureEquipmentID, 393
- setImpressionType, 393
- setMinutiaeData, 394
- setMinutiaeReservedData, 394
- setPosition, 394
- setQuality, 394
- setViewNumber, 395
- BiometricEvaluation::Finger::ISO2005View, 402
  - ISO2005View, 403
  - readCoreDeltaData, 404
- BiometricEvaluation::Framework, 117
  - APICurrentState, 118
  - getCompileDate, 118
  - getCompileTime, 119
  - getCompiler, 119
  - getCompilerVersion, 119
  - getMajorVersion, 119
  - getMinorVersion, 119
  - operator<<, 119
  - to\_string, 120
- BiometricEvaluation::Framework::API< T >, 221
- BiometricEvaluation::Framework::API< T >::Result, 537
- BiometricEvaluation::Framework::API::Result
  - elapsed, 538
  - operator bool, 537
  - operator!, 538
  - Result, 537
  - status, 538
- BiometricEvaluation::Framework::API
  - API, 222
  - call, 222
  - getSignalManager, 223
  - getTimer, 223
  - getWatchdog, 223
- BiometricEvaluation::Framework::Status, 565
  - getCode, 566
  - getMessage, 566
  - OK, 566
  - Status, 565
- BiometricEvaluation::IO::ArchiveRecordStore, 223
  - ~ArchiveRecordStore, 226
  - ARCHIVE\_FILE\_NAME, 233
  - ArchiveRecordStore, 225
  - changeDescription, 226
  - flush, 226
  - getArchiveName, 227
  - getCount, 227
  - getDescription, 227
  - getManifestName, 227
  - getPathname, 227
  - getSpaceUsed, 227
  - insert, 228
  - length, 228
  - MANIFEST\_FILE\_NAME, 233
  - move, 229
  - needsVacuum, 229
  - OFFSET\_RECORD\_REMOVED, 233
  - read, 230
  - remove, 230
  - sequence, 231
  - sequenceKey, 231
  - setCursorAtKey, 232
  - sync, 232
  - vacuum, 232
- BiometricEvaluation::IO::CompressedRecordStore, 261
  - changeDescription, 264
  - CompressedRecordStore, 262–264
  - flush, 264
  - getCount, 265
  - getDescription, 265
  - getPathname, 265
  - getSpaceUsed, 265
  - insert, 266
  - length, 266
  - move, 267
  - operator=, 267
  - read, 267
  - remove, 268
  - sequence, 268
  - sequenceKey, 269
  - setCursorAtKey, 269

- sync, 270
- BiometricEvaluation::IO::Compressor, 270
  - ~Compressor, 272
  - compress, 272–274
  - Compressor, 272
  - createCompressor, 275
  - decompress, 275–278
  - getOption, 278
  - getOptionAsInteger, 278
  - Kind, 271
  - operator=, 280
  - removeOption, 280
  - setOption, 280, 281
- BiometricEvaluation::IO::DBRecordStore, 295
  - changeDescription, 297
  - DBRecordStore, 296, 297
  - flush, 298
  - getCount, 298
  - getDescription, 298
  - getPathname, 298
  - getSpaceUsed, 298
  - insert, 299
  - length, 299
  - move, 300
  - read, 300
  - remove, 301
  - sequence, 301
  - sequenceKey, 301
  - setCursorAtKey, 302
  - sync, 302
- BiometricEvaluation::IO::FileLogCabinet, 317
  - FileLogCabinet, 317, 318
  - getCount, 318
  - getDescription, 318
  - getPathname, 318
  - newLogsheet, 318
- BiometricEvaluation::IO::FileLogsheet, 319
  - \_cursor, 325
  - \_sequenceFile, 325
  - \_theLogFile, 325
  - ~FileLogsheet, 321
  - BE\_FILELOGSHEET\_SEQ\_NEXT, 325
  - BE\_FILELOGSHEET\_SEQ\_START, 325
  - FileLogsheet, 320, 321
  - mergeLogsheets, 322
  - operator=, 322
  - sequence, 322
  - sync, 323
  - trim, 323
  - updateCursor, 323
  - write, 323
  - writeComment, 324
  - writeDebug, 324
- BiometricEvaluation::IO::FileRecordStore, 325
  - changeDescription, 327
  - FileRecordStore, 326, 327
  - flush, 328
  - getCount, 328
  - getDescription, 328
  - getPathname, 328
  - getSpaceUsed, 328
  - insert, 329
  - length, 329
  - move, 330
  - read, 330
  - remove, 331
  - replace, 331
  - sequence, 331
  - sequenceKey, 332
  - setCursorAtKey, 332
  - sync, 333
- BiometricEvaluation::IO::GZip, 347
  - CHUNK\_SIZE, 354
  - COMPRESSION\_LEVEL, 354
  - COMPRESSION\_METHOD, 354
  - COMPRESSION\_STRATEGY, 355
  - compress, 349–351
  - decompress, 351–354
  - GZip, 348
  - INPUT\_DATA\_TYPE, 355
  - MEMORY\_LEVEL, 355
  - operator=, 354
  - WINDOW\_BITS, 355
- BiometricEvaluation::IO::ListRecordStore, 413
  - ~ListRecordStore, 414
  - changeDescription, 414
  - flush, 415
  - getCount, 415
  - getDescription, 415
  - getPathname, 415
  - getSpaceUsed, 415
  - insert, 416
  - length, 416
  - ListRecordStore, 414
  - move, 417
  - read, 417
  - remove, 418
  - replace, 418
  - sequence, 418
  - sequenceKey, 419
  - setCursorAtKey, 419
  - sync, 420



- BiometricEvaluation::IO::Logsheet, 420
  - ~Logsheet, 423
  - CommentDelimiter, 429
  - DebugDelimiter, 429
  - DescriptionTag, 429
  - EntryDelimiter, 429
  - FILEURLSCHEME, 429
  - getAutoSync, 423
  - getCommentCommit, 423
  - getCommit, 423
  - getCurrentEntry, 423
  - getCurrentEntryNumber, 423
  - getCurrentEntryNumberAsString, 424
  - getDebugCommit, 424
  - getTypeFromURL, 424
  - Kind, 422
  - lineIsComment, 424
  - lineIsDebug, 425
  - lineIsEntry, 425
  - newEntry, 425
  - resetCurrentEntry, 426
  - SYSLOGURLSCHEME, 429
  - setAutoSync, 426
  - setCommentCommit, 426
  - setCommit, 426
  - setDebugCommit, 427
  - sync, 427
  - trim, 427
  - write, 427
  - writeComment, 428
  - writeDebug, 428
- BiometricEvaluation::IO::PersistentRecordStoreUnion, 475
  - ~PersistentRecordStoreUnion, 476
  - PersistentRecordStoreUnion, 475, 476
- BiometricEvaluation::IO::Properties, 486
  - ~Properties, 488
  - getMode, 489
  - getProperty, 489
  - getPropertyAsDouble, 489
  - getPropertyAsInteger, 489
  - getPropertyKeys, 490
  - initWithBuffer, 490
  - Properties, 488
  - removeProperty, 491
  - setProperty, 491
  - setPropertyFromBoolean, 492
  - setPropertyFromDouble, 492
  - setPropertyFromInteger, 493
- BiometricEvaluation::IO::PropertiesFile, 493
  - ~PropertiesFile, 495
  - changeName, 495
  - operator=, 495
  - PropertiesFile, 494, 495
  - sync, 496
- BiometricEvaluation::IO::RecordStore, 504
  - BE\_RECSTORE\_SEQ\_NEXT, 517
  - BE\_RECSTORE\_SEQ\_START, 517
  - begin, 506
  - changeDescription, 506
  - containsKey, 507
  - createRecordStore, 507
  - end, 507
  - flush, 508
  - getCount, 508
  - getDescription, 508
  - getPathname, 509
  - getSpaceUsed, 509
  - INVALIDKEYCHARS, 517
  - insert, 509, 510
  - Kind, 506
  - length, 510
  - mergeRecordStores, 511
  - move, 511
  - openRecordStore, 512
  - read, 512
  - remove, 513
  - removeRecordStore, 513
  - replace, 514
  - sequence, 515
  - sequenceKey, 515
  - setCursorAtKey, 516
  - sync, 516
- BiometricEvaluation::IO::RecordStore::Record, 500
  - Record, 500
- BiometricEvaluation::IO::RecordStoreIterator, 520
  - ~RecordStoreIterator, 522
  - difference\_type, 521
  - iterator\_category, 521
  - operator!=, 522
  - operator\*, 522
  - operator+, 522
  - operator++, 523
  - operator+=, 523
  - operator-=, 523
  - operator=>, 523
  - operator=, 524
  - operator==, 524
  - pointer, 521
  - RecordStoreIterator, 521, 522
  - reference, 521
  - value\_type, 521
- BiometricEvaluation::IO::RecordStoreUnion, 526



- ~RecordStoreUnion, 529
  - getNames, 529
  - getRecordStore, 530
  - length, 530
  - read, 530
  - RecordStoreUnion, 527–529
  - setImpl, 531
- BiometricEvaluation::IO::SQLiteRecordStore, 554
  - changeDescription, 555
  - flush, 555
  - getCount, 555
  - getDescription, 556
  - getPathname, 556
  - getSpaceUsed, 556
  - insert, 556
  - length, 557
  - move, 557
  - read, 558
  - remove, 558
  - sequence, 559
  - sequenceKey, 559
  - setCursorAtKey, 560
  - sync, 560
- BiometricEvaluation::IO::SysLogsheet, 569
  - .\_operational, 574
  - .\_sequenced, 574
  - .\_sockFD, 574
  - .\_utc, 574
  - ~SysLogsheet, 572
  - operator=, 572
  - setup, 572
  - sync, 573
  - SysLogsheet, 571, 572
  - write, 573
  - writeComment, 573
  - writeDebug, 574
  - writeToLogger, 574
- BiometricEvaluation::IO::Utility, 127
  - copyDirectoryContents, 129
  - countLines, 129
  - createTemporaryFile, 130
  - fileExists, 131
  - getFileSize, 131
  - isReadable, 132
  - isWritable, 132
  - makePath, 133
  - readFile, 133
  - readPipe, 134
  - removeDirectory, 135
  - setAsideName, 135
  - sumDirectoryUsage, 136
  - writeFile, 136, 137
  - writePipe, 137, 138
- BiometricEvaluation::Image, 120
  - CentimetersPerInch, 126
  - CompressionAlgorithm, 122
  - distance, 123
  - MillimetersPerInch, 126
  - operator<<, 123
  - PixelFormat, 122
  - removeComponents, 123
  - to\_string, 124, 125
- BiometricEvaluation::Image::BMP::ColorTableEntry, 255
  - blue, 255
  - green, 255
  - red, 255
  - reserved, 255
- BiometricEvaluation::Image::BMP, 252
  - getRawData, 252
  - getRawGrayscaleData, 253
  - isBMP, 253
- BiometricEvaluation::Image::Coordinate, 284
  - Coordinate, 284
  - x, 285
  - xDistance, 285
  - y, 285
  - yDistance, 285
- BiometricEvaluation::Image::Image, 355
  - getBitDepth, 359
  - getColorDepth, 359
  - getCompressionAlgorithm, 359, 360
  - getData, 361
  - getDataPointer, 361
  - getDataSize, 361
  - getDimensions, 361
  - getRawData, 361, 362
  - getRawGrayscaleData, 362
  - getRawImage, 363
  - getResolution, 363
  - hasAlphaChannel, 363
  - Image, 358
  - openImage, 364, 365
  - setBitDepth, 365
  - setColorDepth, 365
  - setDimensions, 366
  - setHasAlphaChannel, 366
  - setResolution, 366
  - valueInColorspace, 366
- BiometricEvaluation::Image::JPEG2000, 408
  - getRawData, 409
  - getRawGrayscaleData, 410
  - isJPEG2000, 410

- JPEG2000, 409
- BiometricEvaluation::Image::JPEG, 411
  - getRawData, 411
  - getRawGrayscaleData, 412
  - isJPEG, 412
- BiometricEvaluation::Image::JPEG, 406
  - getRawData, 407
  - getRawGrayscaleData, 407
  - isJPEG, 408
- BiometricEvaluation::Image::NetPBM, 450
  - ASCIIBitmapTo8Bit, 452
  - ASCIIPixmapToBinaryPixmap, 452
  - BinaryBitmapTo8Bit, 453
  - getNextValue, 453
  - getRawData, 454
  - getRawGrayscaleData, 454
  - isNetPBM, 455
  - skipComment, 455
  - skipLine, 456
- BiometricEvaluation::Image::PNG, 476
  - getRawData, 477
  - getRawGrayscaleData, 477
  - isPNG, 478
- BiometricEvaluation::Image::ROI, 539
  - ROI, 539
- BiometricEvaluation::Image::Raw, 497
  - getRawData, 497
  - getRawGrayscaleData, 498
- BiometricEvaluation::Image::Resolution, 531
  - Resolution, 532
  - toUnits, 533
  - Units, 532
  - units, 533
  - xRes, 533
  - yRes, 533
- BiometricEvaluation::Image::Size, 549
  - Size, 549
  - xSize, 549
  - ySize, 549
- BiometricEvaluation::Image::TIFF, 576
  - getRawData, 576
  - getRawGrayscaleData, 577
  - isTIFF, 577, 578
- BiometricEvaluation::Image::WSQ, 611
  - getRawData, 611
  - getRawGrayscaleData, 612
  - isWSQ, 612
- BiometricEvaluation::IO, 126
  - Mode, 127
- BiometricEvaluation::Iris, 138
- BiometricEvaluation::Iris::INCITSView, 377
  - getCameraRange, 379
  - getCaptureDateString, 380
  - getCaptureDeviceTechnology, 380
  - getCaptureDeviceType, 380
  - getCaptureDeviceVendor, 380
  - getCertificationFlag, 380
  - getEyeLabel, 381
  - getIIRData, 381
  - getImageProperties, 381
  - getImageType, 381
  - getIrisCenterInfo, 382
  - getQualitySet, 382
  - getRollAngleInfo, 382
  - INCITSView, 379
  - readHeader, 383
  - readIrisView, 383
- BiometricEvaluation::Iris::INCITSView::QualitySub←  
Block, 496
- BiometricEvaluation::Iris::ISO2011View, 404
  - ISO2011View, 405, 406
- BiometricEvaluation::Latent::AN2KView, 187
  - AN2KView, 187
  - getLatentQualityMetric, 188
  - getPositions, 188
  - getPrintPositionCoordinates, 188
- BiometricEvaluation::MPI::CSVDistributor, 286
  - CHECKPOINTLINECOUNT, 288
  - CHECKPOINTRANDOMSEED, 288
  - CSVDistributor, 287
  - checkpointRestore, 287
  - checkpointSave, 287
  - createWorkPackage, 288
- BiometricEvaluation::MPI::CSVProcessor, 288
  - CSVProcessor, 289
  - newProcessor, 289
  - performInitialization, 290
  - processLine, 290
  - processWorkPackage, 291
- BiometricEvaluation::MPI::CSVResources, 291
  - CHUNKSIZEPROPERTY, 294
  - DELIMITERPROPERTY, 294
  - getDelimiter, 292
  - getNumLines, 292
  - getNumRemainingLines, 293
  - getRandomSeed, 293
  - INPUTCSVPROPERTY, 294
  - RANDOMIZEPROPERTY, 294
  - RANDOMSEEDPROPERTY, 294
  - randomizeLines, 293
  - readLine, 293
  - TRIMPROPERTY, 294

- USEBUFFERPROPERTY, 294
- useBuffer, 294
- BiometricEvaluation::MPI::Distributor, 305
  - CHECKPOINTFILENAME, 308
  - CHECKPOINTPID, 308
  - CHECKPOINTREASON, 308
  - checkpointRestore, 306
  - checkpointSave, 307
  - createWorkPackage, 307
  - Distributor, 306
  - getCheckpointData, 307
  - getLogsheet, 307
  - start, 307
- BiometricEvaluation::MPI::Exception, 312
  - ~Exception, 313
  - Exception, 313
- BiometricEvaluation::MPI::Receiver, 498
  - Receiver, 499
  - start, 499
- BiometricEvaluation::MPI::RecordProcessor, 500
  - newProcessor, 502
  - performInitialization, 502
  - processRecord, 503
  - processWorkPackage, 503
  - RecordProcessor, 501
- BiometricEvaluation::MPI::RecordStoreDistributor, 517
  - CHECKPOINTLASTKEY, 519
  - CHECKPOINTNUMKEYS, 519
  - checkpointRestore, 519
  - checkpointSave, 519
  - createWorkPackage, 519
  - RecordStoreDistributor, 518
- BiometricEvaluation::MPI::RecordStoreResources, 524
  - getOptionalProperties, 525
  - getRecordStore, 525
  - getRequiredProperties, 526
  - haveRecordStore, 526
  - RecordStoreResources, 525
- BiometricEvaluation::MPI::Resources, 533
  - getCheckpointPath, 535
  - getLogsheetURL, 535
  - getOptionalProperties, 535
  - getPropertiesFileName, 536
  - getRequiredProperties, 536
  - NUMCORES, 536
  - NUMCPUS, 536
  - NUMSOCKETS, 536
  - Resources, 535
  - WORKERSPERNODEPROPERTY, 536
- BiometricEvaluation::MPI::Runtime, 540
  - abort, 541
  - Runtime, 540
  - shutdown, 541
  - start, 541
- BiometricEvaluation::MPI::TerminateJob, 575
  - TerminateJob, 575
- BiometricEvaluation::MPI::WorkPackage, 606
  - getNumElements, 607
  - getSize, 607
  - setData, 607
  - setNumElements, 607
  - WorkPackage, 607
- BiometricEvaluation::MPI::WorkPackageProcessor, 608
  - getLogsheet, 609
  - newProcessor, 609
  - performInitialization, 609
  - performShutdown, 610
  - processWorkPackage, 610
  - setLogsheet, 610
- BiometricEvaluation::MPI, 145
  - generateUniqueID, 148
  - logEntry, 148
  - logMessage, 148
  - MessageTag, 147
  - msgtag\_t, 146
  - openLogsheet, 148
  - printStatus, 149
  - TaskCommand, 147
  - TaskStatus, 147
  - taskcmd\_t, 146
  - taskstat\_t, 146
- BiometricEvaluation::Memory, 139
  - isLittleEndian, 140
  - make\_unique, 140, 141
  - operator!=, 141
  - operator<, 141
  - operator<=, 142
  - operator>, 142
  - operator>=, 142
  - operator==, 142
- BiometricEvaluation::Memory::AutoArray
  - ~AutoArray, 237
  - at, 237, 238
  - AutoArray, 236, 237
  - begin, 238
  - cbegin, 238
  - cend, 239
  - const\_iterator, 235
  - const\_reference, 235
  - copy, 239
  - end, 240
  - iterator, 235

- operator const T \*, 240
- operator T\*, 240
- operator=, 240, 241
- operator[], 241, 242
- reference, 235
- resize, 242
- size, 242
- size\_type, 235
- to\_vector, 243
- value\_type, 235
- BiometricEvaluation::Memory::AutoArray< T >, 233
- BiometricEvaluation::Memory::AutoArrayIterator
  - ~AutoArrayIterator, 246
  - AutoArrayIterator, 245, 246
  - difference\_type, 244
  - iterator\_category, 245
  - operator!=, 246
  - operator<, 248
  - operator<=, 249
  - operator>, 249
  - operator>=, 250
  - operator\*, 246
  - operator+, 246, 250
  - operator++, 247
  - operator+=, 247
  - operator-, 247, 250
  - operator->, 248
  - operator--, 248
  - operator-=, 248
  - operator=, 249
  - operator==, 249
  - operator[], 250
  - pointer, 245
  - reference, 245
  - value\_type, 245
- BiometricEvaluation::Memory::AutoArrayIterator< C↵BiometricEvaluation::Memory::OrderedMapConstIterator
  - ONST, T >, 243
- BiometricEvaluation::Memory::AutoArrayUtility, 143
  - cstr, 143
  - getString, 143
  - setString, 144
- BiometricEvaluation::Memory::AutoBuffer
  - value\_type, 251
- BiometricEvaluation::Memory::AutoBuffer< T >, 251
- BiometricEvaluation::Memory::IndexedBuffer, 395
  - ~IndexedBuffer, 397
  - get, 397
  - getIndex, 397
  - getSize, 397
  - IndexedBuffer, 396, 397
  - scan, 397
  - scanBeU16Val, 398
  - scanBeU32Val, 398
  - scanU16Val, 399
  - scanU32Val, 399
  - scanU64Val, 399
  - scanU8Val, 399
  - setIndex, 400
- BiometricEvaluation::Memory::MutableIndexedBuffer, 446
  - ~MutableIndexedBuffer, 447
  - get, 447
  - MutableIndexedBuffer, 446, 447
  - push, 447
  - pushBeU16Val, 448
  - pushBeU32Val, 448
  - pushU16Val, 449
  - pushU32Val, 449
  - pushU64Val, 450
  - pushU8Val, 450
- BiometricEvaluation::Memory::OrderedMap
  - ~OrderedMap, 462
  - begin, 462
  - cbegin, 462
  - cend, 462
  - end, 463
  - erase, 463
  - find, 464
  - key\_eq, 464
  - keyExists, 464
  - operator[], 464
  - OrderedMap, 461
  - push\_back, 465
  - size, 465
- BiometricEvaluation::Memory::OrderedMap< Key, T >, 460
  - ~OrderedMapConstIterator, 467
  - difference\_type, 466
  - iterator\_category, 466
  - operator!=, 467
  - operator\*, 468
  - operator++, 468
  - operator->, 469
  - operator--, 468
  - operator==, 469
  - OrderedMapConstIterator, 467
  - pointer, 466
  - reference, 467
  - value\_type, 467
- BiometricEvaluation::Memory::OrderedMapConstIterator< Key, T >, 465

- BiometricEvaluation::Memory::OrderedMapIterator
  - ~OrderedMapIterator, 471
  - difference\_type, 470
  - iterator\_category, 470
  - operator!=, 471
  - operator\*, 471
  - operator++, 471, 472
  - operator->, 472
  - operator--, 472
  - operator==, 472
  - OrderedMapIterator, 471
  - pointer, 470
  - reference, 470
  - value\_type, 470
- BiometricEvaluation::Memory::OrderedMapIterator< Key, T >, 469
- BiometricEvaluation::Memory::unique\_if
  - unique\_single, 586
- BiometricEvaluation::Memory::unique\_if< T >, 586
- BiometricEvaluation::Memory::unique\_if< T[] >, 586
  - unique\_array\_unknown\_bound, 587
- BiometricEvaluation::Memory::unique\_if< T[S] >, 587
  - unique\_array\_known\_bound, 587
- BiometricEvaluation::Palm, 149
  - Position, 150
- BiometricEvaluation::Palm::AN2KView, 181
  - AN2KView, 182
  - getPalmQualityMetric, 182
  - getPosition, 182
- BiometricEvaluation::Plantar, 150
  - Position, 150
- BiometricEvaluation::Process, 150
  - ParameterList, 151
- BiometricEvaluation::Process::CommandCenter
  - ~CommandCenter, 257
  - CommandCenter, 257
  - disconnectClient, 258
  - getNextCommand, 258
  - hasPendingCommands, 258
  - sendResponse, 259
- BiometricEvaluation::Process::CommandCenter< T, type-name >, 256
- BiometricEvaluation::Process::CommandCenter< T, type-name >::Command, 256
- BiometricEvaluation::Process::CommandCenter::Command
  - arguments, 256
  - clientId, 256
  - command, 256
- BiometricEvaluation::Process::CommandParser
  - ~CommandParser, 260
  - CommandParser, 260
  - getNextCommand, 260
  - getUsage, 260
  - parse, 261
  - setUsage, 261
- BiometricEvaluation::Process::CommandParser< T >, 259
- BiometricEvaluation::Process::ForkManager, 335
  - addWorker, 336
  - broadcastSignal, 337
  - defaultExitCallback, 337
  - FORKMANAGERS, 341
  - ForkManager, 336
  - getIsWorkingStatus, 337
  - responsibleFor, 337
  - setExitCallback, 338
  - setExitStatus, 338
  - setNotWorking, 339
  - startWorker, 339
  - startWorkers, 339
  - stopWorker, 340
  - waitForWorkerExit, 340
- BiometricEvaluation::Process::ForkWorkerController, 341
  - \_stop, 342
  - everWorked, 342
  - ForkManager::addWorker, 343
  - ForkManager::setExitStatus, 344
  - ForkManager::startWorker, 344
  - ForkManager::startWorkers, 345
  - ForkManager::stopWorker, 345
  - getPID, 342
  - isWorking, 343
  - reset, 343
- BiometricEvaluation::Process::Manager, 430
  - \_pendingExit, 435
  - \_workers, 435
  - addWorker, 431
  - broadcastMessage, 431
  - getNextMessage, 432
  - getNumActiveWorkers, 432
  - getNumCompletedWorkers, 432
  - getTotalWorkers, 433
  - reset, 433
  - startWorker, 433
  - startWorkers, 434
  - stopWorker, 434
  - waitForMessage, 435
  - waitForWorkerExit, 435
- BiometricEvaluation::Process::MessageCenter, 436
  - CONNECTION\_BACKLOG, 438
  - DEFAULT\_PORT, 439

- DEFAULT\_TIMEOUT, 439
- disconnectClient, 437
- getNextMessage, 437
- hasUnseenMessages, 438
- MAX\_MESSAGE\_LENGTH, 439
- MessageCenter, 437
- sendResponse, 438
- BiometricEvaluation::Process::MessageCenterListener, 439
- PARAM\_PORT, 440
- workerMain, 439
- BiometricEvaluation::Process::MessageCenterReceiver, 440
  - ~MessageCenterReceiver, 441
  - MSG\_DISCONNECT, 441
  - MessageCenterReceiver, 441
  - PARAM\_CLIENT\_ID, 441
  - PARAM\_CLIENT\_SOCKET, 441
  - workerMain, 441
- BiometricEvaluation::Process::POSIXThreadManager, 480
  - addWorker, 481
  - POSIXThreadManager, 481
  - startWorker, 482
  - startWorkers, 482
  - stopWorker, 483
  - waitForWorkerExit, 483
- BiometricEvaluation::Process::POSIXThreadWorker←Controller, 483
  - everWorked, 484
  - isWorking, 484
  - reset, 484
- BiometricEvaluation::Process::Semaphore, 541
  - getName, 543
  - post, 543
  - Semaphore, 542, 543
  - timedwait, 543
  - trywait, 544
  - wait, 544
- BiometricEvaluation::Process::Statistics, 560
  - callStatistics\_logStats, 562
  - getCPUTimes, 562
  - getMemorySizes, 563
  - getNumThreads, 563
  - logStats, 564
  - startAutoLogging, 564
  - Statistics, 561, 562
  - stopAutoLogging, 565
- BiometricEvaluation::Process::Worker, 595
  - \_initCommunication, 596
  - closeManagerPipeEnds, 596
  - closeWorkerPipeEnds, 596
  - getParameter, 597
  - getParameterAsDouble, 597
  - getParameterAsInteger, 597
  - getParameterAsString, 598
  - getReceivingPipe, 598
  - getSendingPipe, 599
  - receiveMessageFromManager, 599
  - sendMessageToManager, 599
  - setParameter, 600
  - stopRequested, 600
  - waitForMessage, 600
  - workerMain, 601
- BiometricEvaluation::Process::WorkerController, 601
  - \_rv, 606
  - \_rvSet, 606
  - \_worker, 606
  - everWorked, 602
  - finishedWorking, 603
  - getExitStatus, 603
  - getWorker, 603
  - isWorking, 603
  - reset, 604
  - sendMessageToWorker, 604
  - setParameter, 604
  - setParameterFromDouble, 605
  - setParameterFromInteger, 605
  - setParameterFromString, 605
  - WorkerController, 602
- BiometricEvaluation::System, 152
  - getCPUCoreCount, 152
  - getCPUCount, 152
  - getCPUSocketCount, 153
  - getLoadAverage, 153
  - getRealMemorySize, 153
- BiometricEvaluation::Text, 154
  - basename, 155
  - caseInsensitiveCompare, 155
  - decodeBase64, 155
  - digest, 156
  - dirname, 157
  - encodeBase64, 157
  - ltrim, 157
  - ltrimWhitespace, 158
  - rtrim, 158
  - rtrimWhitespace, 158
  - split, 159
  - toLowercase, 159
  - toUppercase, 160
  - trim, 160
  - trimWhitespace, 160

- BiometricEvaluation::Time, 161
  - getCurrentCalendarInformation, 161
  - getCurrentDate, 162
  - getCurrentDateAndTime, 162
  - getCurrentTime, 162
  - operator<<, 162
  - put\_time, 163
- BiometricEvaluation::Time::Timer, 578
  - BE\_CLOCK\_TYPE, 579
  - elapsed, 579
  - elapsedStr, 580
  - start, 580
  - stop, 581
  - time, 581
  - Timer, 579
- BiometricEvaluation::Time::Watchdog, 591
  - clearCanSigJump, 593
  - clearExpired, 593
  - expired, 593
  - PROCESSTIME, 594
  - REALTIME, 594
  - setCanSigJump, 593
  - setExpired, 593
  - setInterval, 593
  - start, 594
  - stop, 594
  - Watchdog, 592
- BiometricEvaluation::Video, 163
  - CodingFormat, 164
  - ContainerFormat, 164
- BiometricEvaluation::Video::Container, 281
  - Container, 282
  - getVideoStream, 283
- BiometricEvaluation::Video::Frame, 346
- BiometricEvaluation::Video::Stream, 567
  - getFPS, 568
  - getFrame, 568
  - getFrameCount, 568
  - getFrameSequence, 568
  - setFramePixelFormat, 569
  - setFrameScale, 569
- BiometricEvaluation::View, 164
  - operator<<, 165
- BiometricEvaluation::View::AN2KView, 183
  - AN2KView, 185
  - convertCompressionAlgorithm, 185
  - convertDeviceMonitoringMode, 186
  - DeviceMonitoringMode, 184
  - getAN2KRecord, 186
  - getMinutiaeDataRecordSet, 186
  - getRecordType, 186
  - RecordType, 184
- BiometricEvaluation::View::AN2KViewVariableResolution, 200
  - AN2KViewVariableResolution, 201
  - extractQuality, 201
  - getCaptureDate, 202
  - getComment, 202
  - getImpressionType, 202
  - getPositionDescriptors, 202
  - getPositions, 203
  - getPrintPositionCoordinates, 203
  - getQualityMetric, 203
  - getSourceAgency, 203
  - getUserDefinedField, 203
  - parseUserDefinedField, 204
- BiometricEvaluation::View::AN2KViewVariableResolution↔
  - ::AN2KQualityMetric, 174
- BiometricEvaluation::View::AN2KViewVariableResolution↔
  - ::PrintPositionCoordinate, 486
  - coordinates, 486
  - fingerView, 486
  - segment, 486
- BiometricEvaluation::View::View, 587
  - getCompressionAlgorithm, 588
  - getImage, 588
  - getImageColorDepth, 589
  - getImageResolution, 589
  - getImageSize, 589
  - getScanResolution, 589
  - setImageColorDepth, 590
  - setImageData, 590
  - setImageResolution, 590
  - setImageSize, 590
  - setScanResolution, 591
- blue
  - BiometricEvaluation::Image::BMP::ColorTable↔
    - Entry, 255
- broadcastMessage
  - BiometricEvaluation::Process::Manager, 431
- broadcastSignal
  - BiometricEvaluation::Process::ForkManager, 337
- CHECKPOINTFILENAME
  - BiometricEvaluation::MPI::Distributor, 308
- CHECKPOINTLASTKEY
  - BiometricEvaluation::MPI::RecordStoreDistributor, 519
- CHECKPOINTLINECOUNT
  - BiometricEvaluation::MPI::CSVDistributor, 288
- CHECKPOINTNUMKEYS
  - BiometricEvaluation::MPI::RecordStoreDistributor, 519



- CHECKPOINTPID
  - BiometricEvaluation::MPI::Distributor, 308
- CHECKPOINTRANDOMSEED
  - BiometricEvaluation::MPI::CSVDistributor, 288
- CHECKPOINTREASON
  - BiometricEvaluation::MPI::Distributor, 308
- CHUNK\_SIZE
  - BiometricEvaluation::IO::GZip, 354
- CHUNKSIZEPROPERTY
  - BiometricEvaluation::MPI::CSVResources, 294
- COMPRESSION\_LEVEL
  - BiometricEvaluation::IO::GZip, 354
- COMPRESSION\_METHOD
  - BiometricEvaluation::IO::GZip, 354
- COMPRESSION\_STRATEGY
  - BiometricEvaluation::IO::GZip, 355
- CONNECTION\_BACKLOG
  - BiometricEvaluation::Process::MessageCenter, 438
- CSVDistributor
  - BiometricEvaluation::MPI::CSVDistributor, 287
- CSVProcessor
  - BiometricEvaluation::MPI::CSVProcessor, 289
- call
  - BiometricEvaluation::Framework::API, 222
- callStatistics.logStats
  - BiometricEvaluation::Process::Statistics, 562
- caseInsensitiveCompare
  - BiometricEvaluation::Text, 155
- cbegin
  - BiometricEvaluation::Memory::AutoArray, 238
  - BiometricEvaluation::Memory::OrderedMap, 462
- cend
  - BiometricEvaluation::Memory::AutoArray, 239
  - BiometricEvaluation::Memory::OrderedMap, 462
- centerOfImage
  - BiometricEvaluation::Feature::Sort::Polar, 479
- centerOfMinutiaeMass
  - BiometricEvaluation::Feature::Sort::Polar, 479
- CentimetersPerInch
  - BiometricEvaluation::Image, 126
- changeDescription
  - BiometricEvaluation::IO::ArchiveRecordStore, 226
  - BiometricEvaluation::IO::CompressedRecordStore, 264
  - BiometricEvaluation::IO::DBRecordStore, 297
  - BiometricEvaluation::IO::FileRecordStore, 327
  - BiometricEvaluation::IO::ListRecordStore, 414
  - BiometricEvaluation::IO::RecordStore, 506
  - BiometricEvaluation::IO::SQLiteRecordStore, 555
- changeName
  - BiometricEvaluation::IO::PropertiesFile, 495
- CharacterSet
  - BiometricEvaluation::DataInterchange::AN2K↔Record, 176
  - BiometricEvaluation::DataInterchange::AN2K↔Record::CharacterSet, 254
- checkpointRestore
  - BiometricEvaluation::MPI::CSVDistributor, 287
  - BiometricEvaluation::MPI::Distributor, 306
  - BiometricEvaluation::MPI::RecordStoreDistributor, 519
- checkpointSave
  - BiometricEvaluation::MPI::CSVDistributor, 287
  - BiometricEvaluation::MPI::Distributor, 307
  - BiometricEvaluation::MPI::RecordStoreDistributor, 519
- cla
  - BiometricEvaluation::Device::Smartcard::APDU, 218
- clearCanSigJump
  - BiometricEvaluation::Time::Watchdog, 593
- clearExpired
  - BiometricEvaluation::Time::Watchdog, 593
- clearSigHandled
  - BiometricEvaluation::Error::SignalManager, 547
- clearSignalSet
  - BiometricEvaluation::Error::SignalManager, 547
- clientID
  - BiometricEvaluation::Process::CommandCenter↔::Command, 256
- closeManagerPipeEnds
  - BiometricEvaluation::Process::Worker, 596
- closeWorkerPipeEnds
  - BiometricEvaluation::Process::Worker, 596
- code
  - BiometricEvaluation::Feature::AN2K7Minutiae↔::PatternClassification::Entry, 310
- CodingFormat
  - BiometricEvaluation::Video, 164
- command
  - BiometricEvaluation::Process::CommandCenter↔::Command, 256
- CommandCenter
  - BiometricEvaluation::Process::CommandCenter, 257
- CommandParser
  - BiometricEvaluation::Process::CommandParser, 260
- CommentDelimiter
  - BiometricEvaluation::IO::Logsheet, 429
- commonName
  - BiometricEvaluation::DataInterchange::AN2K↔Record::CharacterSet, 254



- compress
  - BiometricEvaluation::IO::Compressor, 272–274
  - BiometricEvaluation::IO::GZip, 349–351
- CompressedRecordStore
  - BiometricEvaluation::IO::CompressedRecordStore, createRecordStore 262–264
- CompressionAlgorithm
  - BiometricEvaluation::Image, 122
- Compressor
  - BiometricEvaluation::IO::Compressor, 272
- const\_iterator
  - BiometricEvaluation::Memory::AutoArray, 235
- const\_reference
  - BiometricEvaluation::Memory::AutoArray, 235
- Container
  - BiometricEvaluation::Video::Container, 282
- ContainerFormat
  - BiometricEvaluation::Video, 164
- containsKey
  - BiometricEvaluation::IO::RecordStore, 507
- ConversionError
  - BiometricEvaluation::Error::ConversionError, 284
- convertCompressionAlgorithm
  - BiometricEvaluation::View::AN2KView, 185
- convertCoordinate
  - BiometricEvaluation::Feature::AN2K7Minutiae, 169
- convertDeviceMonitoringMode
  - BiometricEvaluation::View::AN2KView, 186
- convertEncodingMethod
  - BiometricEvaluation::Feature::AN2K7Minutiae, 170
- convertFingerImageCode
  - BiometricEvaluation::Finger::AN2KView, 191
- convertImpression
  - BiometricEvaluation::Finger::INCITSView, 387
- convertPatternClassification
  - BiometricEvaluation::Feature::AN2K7Minutiae, 170, 171
- convertPosition
  - BiometricEvaluation::Finger::AN2KView, 192
  - BiometricEvaluation::Finger::INCITSView, 387
- Coordinate
  - BiometricEvaluation::Image::Coordinate, 284
- coordinates
  - BiometricEvaluation::Finger::AN2KViewCapture←  
::FingerSegmentPosition, 335
  - BiometricEvaluation::View::AN2KViewVariable←  
Resolution::PrintPositionCoordinate, 486
- copy
  - BiometricEvaluation::Memory::AutoArray, 239
- copyDirectoryContents
  - BiometricEvaluation::IO::Utility, 129
- countLines
  - BiometricEvaluation::IO::Utility, 129
- createCompressor
  - BiometricEvaluation::IO::Compressor, 275
- createRecordStore
  - BiometricEvaluation::IO::RecordStore, 507
- createTemporaryFile
  - BiometricEvaluation::IO::Utility, 130
- createWorkPackage
  - BiometricEvaluation::MPI::CSVDistributor, 288
  - BiometricEvaluation::MPI::Distributor, 307
  - BiometricEvaluation::MPI::RecordStoreDistributor, 519
- cstr
  - BiometricEvaluation::Memory::AutoArrayUtility, 143
- DBRecordStore
  - BiometricEvaluation::IO::DBRecordStore, 296, 297
- DEFAULT\_PORT
  - BiometricEvaluation::Process::MessageCenter, 439
- DEFAULT\_TIMEOUT
  - BiometricEvaluation::Process::MessageCenter, 439
- DELIMITERPROPERTY
  - BiometricEvaluation::MPI::CSVResources, 294
- data
  - BiometricEvaluation::Device::Smartcard::APD←  
UResponse, 221
- DataError
  - BiometricEvaluation::Error::DataError, 295
- DebugDelimiter
  - BiometricEvaluation::IO::Logsheet, 429
- decodeBase64
  - BiometricEvaluation::Text, 155
- decompress
  - BiometricEvaluation::IO::Compressor, 275–278
  - BiometricEvaluation::IO::GZip, 351–354
- defaultExitCallback
  - BiometricEvaluation::Process::ForkManager, 337
- DescriptionTag
  - BiometricEvaluation::IO::Logsheet, 429
- DeviceMonitoringMode
  - BiometricEvaluation::View::AN2KView, 184
- difference\_type
  - BiometricEvaluation::IO::RecordStoreIterator, 521
  - BiometricEvaluation::Memory::AutoArrayIterator, 244
  - BiometricEvaluation::Memory::OrderedMapConst←  
Iterator, 466
  - BiometricEvaluation::Memory::OrderedMapIterator, 470
- digest

- BiometricEvaluation::Text, [156](#)
- dirname
  - BiometricEvaluation::Text, [157](#)
- disconnectClient
  - BiometricEvaluation::Process::CommandCenter, [258](#)
  - BiometricEvaluation::Process::MessageCenter, [437](#)
- distance
  - BiometricEvaluation::Image, [123](#)
- Distributor
  - BiometricEvaluation::MPI::Distributor, [306](#)
- DomainName
  - BiometricEvaluation::DataInterchange::AN2K←Record, [176](#)
  - BiometricEvaluation::DataInterchange::AN2K←Record::DomainName, [308](#)
- EODDefault
  - BiometricEvaluation::Feature::AN2K11EFS::Orientation, [473](#)
- EUCDefault
  - BiometricEvaluation::Feature::AN2K11EFS::Orientation, [473](#)
- elapsed
  - BiometricEvaluation::Framework::API::Result, [538](#)
  - BiometricEvaluation::Time::Timer, [579](#)
- elapsedStr
  - BiometricEvaluation::Time::Timer, [580](#)
- encodeBase64
  - BiometricEvaluation::Text, [157](#)
- EncodingMethod
  - BiometricEvaluation::Feature::AN2K7Minutiae, [168](#)
- end
  - BiometricEvaluation::IO::RecordStore, [507](#)
  - BiometricEvaluation::Memory::AutoArray, [240](#)
  - BiometricEvaluation::Memory::OrderedMap, [463](#)
- Entry
  - BiometricEvaluation::Feature::AN2K7Minutiae←::PatternClassification::Entry, [309](#)
- EntryDelimiter
  - BiometricEvaluation::IO::Logsheet, [429](#)
- eod
  - BiometricEvaluation::Feature::AN2K11EFS::Orientation, [473](#)
- equipment
  - BiometricEvaluation::Feature::AN2K7Minutiae←::FingerprintReadingSystem, [333](#)
- erase
  - BiometricEvaluation::Memory::OrderedMap, [463](#)
- errorStr
  - BiometricEvaluation::Error, [109](#)
- euc
  - BiometricEvaluation::Feature::AN2K11EFS::Orientation, [473](#)
  - everWorked
    - BiometricEvaluation::Process::ForkWorkerController, [342](#)
    - BiometricEvaluation::Process::POSIXThreadWorker←Controller, [484](#)
    - BiometricEvaluation::Process::WorkerController, [602](#)
  - Exception
    - BiometricEvaluation::Error::Exception, [311](#)
    - BiometricEvaluation::MPI::Exception, [313](#)
  - expired
    - BiometricEvaluation::Time::Watchdog, [593](#)
  - ExtendedFeatureSet
    - BiometricEvaluation::Feature::AN2K11EFS::Extended←FeatureSet, [314](#)
  - extractNISTQuality
    - BiometricEvaluation::Finger::AN2KViewCapture, [196](#)
  - extractQuality
    - BiometricEvaluation::View::AN2KViewVariable←Resolution, [201](#)
  - FIELD\_LC
    - BiometricEvaluation::Device::Smartcard::APDU, [218](#)
  - FIELD\_LE
    - BiometricEvaluation::Device::Smartcard::APDU, [218](#)
  - FILEURLSCHEME
    - BiometricEvaluation::IO::Logsheet, [429](#)
  - FORKMANAGERS
    - BiometricEvaluation::Process::ForkManager, [341](#)
  - fgp
    - BiometricEvaluation::Feature::AN2K11EFS::F←PPPosition, [346](#)
  - field\_mask
    - BiometricEvaluation::Device::Smartcard::APDU, [218](#)
  - FileError
    - BiometricEvaluation::Error::FileError, [317](#)
  - fileExists
    - BiometricEvaluation::IO::Utility, [131](#)
  - FileLogCabinet
    - BiometricEvaluation::IO::FileLogCabinet, [317](#), [318](#)
  - FileLogsheet
    - BiometricEvaluation::IO::FileLogsheet, [320](#), [321](#)
  - FileRecordStore
    - BiometricEvaluation::IO::FileRecordStore, [326](#), [327](#)
  - find
    - BiometricEvaluation::Memory::OrderedMap, [464](#)

- FingerImageCode
  - BiometricEvaluation::Finger, [117](#)
- fingerPosition
  - BiometricEvaluation::Finger::AN2KViewCapture←getAN2K11EFS  
::FingerSegmentPosition, [335](#)
- FingerSegmentPosition
  - BiometricEvaluation::Finger::AN2KViewCapture←getAN2K7Minutiae  
::FingerSegmentPosition, [334](#)
- fingerView
  - BiometricEvaluation::View::AN2KViewVariable←getAN2KRecord  
Resolution::PrintPositionCoordinate, [486](#)
- finishedWorking
  - BiometricEvaluation::Process::WorkerController,  
[603](#)
- flush
  - BiometricEvaluation::IO::ArchiveRecordStore, [226](#)
  - BiometricEvaluation::IO::CompressedRecordStore,  
[264](#)
  - BiometricEvaluation::IO::DBRecordStore, [298](#)
  - BiometricEvaluation::IO::FileRecordStore, [328](#)
  - BiometricEvaluation::IO::ListRecordStore, [415](#)
  - BiometricEvaluation::IO::RecordStore, [508](#)
  - BiometricEvaluation::IO::SQLiteRecordStore, [555](#)
- ForkManager
  - BiometricEvaluation::Process::ForkManager, [336](#)
- ForkManager::addWorker
  - BiometricEvaluation::Process::ForkWorkerController,  
[343](#)
- ForkManager::setExitStatus
  - BiometricEvaluation::Process::ForkWorkerController,  
[344](#)
- ForkManager::startWorker
  - BiometricEvaluation::Process::ForkWorkerController,  
[344](#)
- ForkManager::startWorkers
  - BiometricEvaluation::Process::ForkWorkerController,  
[345](#)
- ForkManager::stopWorker
  - BiometricEvaluation::Process::ForkWorkerController,  
[345](#)
- fpp
  - BiometricEvaluation::Feature::AN2K11EFS::Image←  
Info, [367](#)
- fsm
  - BiometricEvaluation::Feature::AN2K11EFS::F←  
PPPPosition, [346](#)
- GZip
  - BiometricEvaluation::IO::GZip, [348](#)
- generateUniqueID
  - BiometricEvaluation::MPI, [148](#)
- get
  - BiometricEvaluation::Memory::IndexedBuffer, [397](#)
  - BiometricEvaluation::Memory::MutableIndexed←  
Buffer, [447](#)
  - BiometricEvaluation::Finger::AN2KMinutiaeData←  
Record, [173](#)
  - BiometricEvaluation::Finger::AN2KMinutiaeData←  
Record, [173](#)
  - BiometricEvaluation::View::AN2KView, [186](#)
  - getAlternateFingerSegmentPositionSet  
BiometricEvaluation::Finger::AN2KViewCapture,  
[196](#)
  - getAmputatedBandaged  
BiometricEvaluation::Finger::AN2KViewCapture,  
[196](#)
  - getArchiveName  
BiometricEvaluation::IO::ArchiveRecordStore, [227](#)
  - getAutoSync  
BiometricEvaluation::IO::Logsheet, [423](#)
  - getBitDepth  
BiometricEvaluation::Image::Image, [359](#)
  - getCPUCoreCount  
BiometricEvaluation::System, [152](#)
  - getCPUCount  
BiometricEvaluation::System, [152](#)
  - getCPUSocketCount  
BiometricEvaluation::System, [153](#)
  - getCPUTimes  
BiometricEvaluation::Process::Statistics, [562](#)
  - getCPS  
BiometricEvaluation::Feature::AN2K11EFS::Extended←  
FeatureSet, [315](#)
  - getCameraRange  
BiometricEvaluation::Iris::INCITSView, [379](#)
  - getCaptureDate  
BiometricEvaluation::View::AN2KViewVariable←  
Resolution, [202](#)
  - getCaptureDateString  
BiometricEvaluation::Iris::INCITSView, [380](#)
  - getCaptureDeviceTechnology  
BiometricEvaluation::Iris::INCITSView, [380](#)
  - getCaptureDeviceType  
BiometricEvaluation::Iris::INCITSView, [380](#)
  - getCaptureDeviceVendor  
BiometricEvaluation::Iris::INCITSView, [380](#)
  - getCaptureEquipmentID  
BiometricEvaluation::Finger::INCITSView, [387](#)
  - getCertificationFlag  
BiometricEvaluation::Iris::INCITSView, [380](#)

- getCheckpointData
  - BiometricEvaluation::MPI::Distributor, 307
- getCheckpointPath
  - BiometricEvaluation::MPI::Resources, 535
- getChildren
  - BiometricEvaluation::Device::TLV, 583
- getCode
  - BiometricEvaluation::Framework::Status, 566
- getColorDepth
  - BiometricEvaluation::Image::Image, 359
- getColorSpace
  - BiometricEvaluation::Face::INCITSView, 373
- getComment
  - BiometricEvaluation::View::AN2KViewVariable↔ Resolution, 202
- getCommentCommit
  - BiometricEvaluation::IO::Logsheet, 423
- getCommit
  - BiometricEvaluation::IO::Logsheet, 423
- getCompileDate
  - BiometricEvaluation::Framework, 118
- getCompileTime
  - BiometricEvaluation::Framework, 119
- getCompiler
  - BiometricEvaluation::Framework, 119
- getCompilerVersion
  - BiometricEvaluation::Framework, 119
- getCompressionAlgorithm
  - BiometricEvaluation::Image::Image, 359, 360
  - BiometricEvaluation::View::View, 588
- getCount
  - BiometricEvaluation::IO::ArchiveRecordStore, 227
  - BiometricEvaluation::IO::CompressedRecordStore, 265
  - BiometricEvaluation::IO::DBRecordStore, 298
  - BiometricEvaluation::IO::FileLogCabinet, 318
  - BiometricEvaluation::IO::FileRecordStore, 328
  - BiometricEvaluation::IO::ListRecordStore, 415
  - BiometricEvaluation::IO::RecordStore, 508
  - BiometricEvaluation::IO::SQLiteRecordStore, 555
- getCurrentCalendarInformation
  - BiometricEvaluation::Time, 161
- getCurrentDate
  - BiometricEvaluation::Time, 162
- getCurrentDateAndTime
  - BiometricEvaluation::Time, 162
- getCurrentEntry
  - BiometricEvaluation::IO::Logsheet, 423
- getCurrentEntryNumber
  - BiometricEvaluation::IO::Logsheet, 423
- getCurrentEntryNumberAsString
  - BiometricEvaluation::IO::Logsheet, 424
- getCurrentTime
  - BiometricEvaluation::Time, 162
- getDPS
  - BiometricEvaluation::Feature::AN2K11EFS::Extended↔ FeatureSet, 315
- getData
  - BiometricEvaluation::Image::Image, 361
- getDataPointer
  - BiometricEvaluation::Image::Image, 361
- getDataSize
  - BiometricEvaluation::Image::Image, 361
- getDate
  - BiometricEvaluation::DataInterchange::AN2K↔ Record, 177
- getDebugCommit
  - BiometricEvaluation::IO::Logsheet, 424
- getDedicatedFileObject
  - BiometricEvaluation::Device::Smartcard, 552
- getDelimiter
  - BiometricEvaluation::MPI::CSVResources, 292
- getDescription
  - BiometricEvaluation::IO::ArchiveRecordStore, 227
  - BiometricEvaluation::IO::CompressedRecordStore, 265
  - BiometricEvaluation::IO::DBRecordStore, 298
  - BiometricEvaluation::IO::FileLogCabinet, 318
  - BiometricEvaluation::IO::FileRecordStore, 328
  - BiometricEvaluation::IO::ListRecordStore, 415
  - BiometricEvaluation::IO::RecordStore, 508
  - BiometricEvaluation::IO::SQLiteRecordStore, 556
- getDestinationAgency
  - BiometricEvaluation::DataInterchange::AN2K↔ Record, 177
- getDeviceType
  - BiometricEvaluation::Face::INCITSView, 374
- getDimensions
  - BiometricEvaluation::Image::Image, 361
- getDirectoryOfCharacterSets
  - BiometricEvaluation::DataInterchange::AN2K↔ Record, 177
- getDomainName
  - BiometricEvaluation::DataInterchange::AN2K↔ Record, 178
- getEDBLength
  - BiometricEvaluation::DataInterchange::ANSI2004↔ Record, 207
  - BiometricEvaluation::Finger::INCITSView, 388
- getExitStatus
  - BiometricEvaluation::Process::WorkerController, 603

- getEyeColor
  - BiometricEvaluation::Face::INCITSView, [374](#)
- getEyeLabel
  - BiometricEvaluation::Iris::INCITSView, [381](#)
- getFIDData
  - BiometricEvaluation::Face::INCITSView, [374](#)
- getFIRData
  - BiometricEvaluation::Finger::INCITSView, [388](#)
- getFMRData
  - BiometricEvaluation::Finger::INCITSView, [388](#)
- getFMRLength
  - BiometricEvaluation::DataInterchange::ANSI2004↔
    - Record, [208](#)
- getFMRReservedByte
  - BiometricEvaluation::Finger::INCITSView, [388](#)
- getFMR
  - BiometricEvaluation::DataInterchange::ANSI2004↔
    - Record, [207](#)
- getFPS
  - BiometricEvaluation::Video::Stream, [568](#)
- getFeaturePointSet
  - BiometricEvaluation::Face::INCITSView, [374](#)
- getFileSize
  - BiometricEvaluation::IO::Utility, [131](#)
- getFingerCaptureCount
  - BiometricEvaluation::DataInterchange::AN2K↔
    - Record, [178](#)
- getFingerCaptures
  - BiometricEvaluation::DataInterchange::AN2K↔
    - Record, [178](#)
- getFingerLatentCount
  - BiometricEvaluation::DataInterchange::AN2K↔
    - Record, [178](#)
- getFingerLatents
  - BiometricEvaluation::DataInterchange::AN2K↔
    - Record, [178](#)
- getFingerSegmentPositionSet
  - BiometricEvaluation::Finger::AN2KViewCapture, [197](#)
- getFingerprintQualityMetric
  - BiometricEvaluation::Finger::AN2KViewCapture, [196](#)
- getFrame
  - BiometricEvaluation::Video::Stream, [568](#)
- getFrameCount
  - BiometricEvaluation::Video::Stream, [568](#)
- getFrameSequence
  - BiometricEvaluation::Video::Stream, [568](#)
- getGender
  - BiometricEvaluation::Face::INCITSView, [374](#)
- getGreenwichMeanTime
  - BiometricEvaluation::DataInterchange::AN2K↔
    - Record, [179](#)
- getHairColor
  - BiometricEvaluation::Face::INCITSView, [375](#)
- getIIRData
  - BiometricEvaluation::Iris::INCITSView, [381](#)
- getImage
  - BiometricEvaluation::View::View, [588](#)
- getImageColorDepth
  - BiometricEvaluation::View::View, [589](#)
- getImageDataType
  - BiometricEvaluation::Face::INCITSView, [375](#)
- getImageInfo
  - BiometricEvaluation::Feature::AN2K11EFS::Extended↔
    - FeatureSet, [315](#)
- getImageProperties
  - BiometricEvaluation::Iris::INCITSView, [381](#)
- getImageResolution
  - BiometricEvaluation::View::View, [589](#)
- getImageSize
  - BiometricEvaluation::View::View, [589](#)
- getImageType
  - BiometricEvaluation::Face::INCITSView, [375](#)
  - BiometricEvaluation::Iris::INCITSView, [381](#)
- getImpressionType
  - BiometricEvaluation::Finger::AN2KMinutiaeData↔
    - Record, [173](#)
  - BiometricEvaluation::Finger::AN2KView, [192](#)
  - BiometricEvaluation::Finger::INCITSView, [388](#)
  - BiometricEvaluation::View::AN2KViewVariable↔
    - Resolution, [202](#)
- getIndex
  - BiometricEvaluation::Memory::IndexedBuffer, [397](#)
- getIrisCenterInfo
  - BiometricEvaluation::Iris::INCITSView, [382](#)
- getIsWorkingStatus
  - BiometricEvaluation::Process::ForkManager, [337](#)
- getLastAPDU
  - BiometricEvaluation::Device::Smartcard, [552](#)
- getLastResponseData
  - BiometricEvaluation::Device::Smartcard, [552](#)
- getLatentQualityMetric
  - BiometricEvaluation::Latent::AN2KView, [188](#)
- getLoadAverage
  - BiometricEvaluation::System, [153](#)
- getLogsheets
  - BiometricEvaluation::MPI::Distributor, [307](#)
  - BiometricEvaluation::MPI::WorkPackageProcessor, [609](#)
- getLogsheetsURL
  - BiometricEvaluation::MPI::Resources, [535](#)

- getMPS
  - BiometricEvaluation::Feature::AN2K11EFS::Extended↔  
FeatureSet, 315
- getMRCI
  - BiometricEvaluation::Feature::AN2K11EFS::Extended↔  
FeatureSet, 316
- getMajorVersion
  - BiometricEvaluation::Framework, 119
- getManifestName
  - BiometricEvaluation::IO::ArchiveRecordStore, 227
- getMemorySizes
  - BiometricEvaluation::Process::Statistics, 563
- getMessage
  - BiometricEvaluation::Framework::Status, 566
- getMinorVersion
  - BiometricEvaluation::Framework, 119
- getMinutia
  - BiometricEvaluation::DataInterchange::ANSI2004↔  
Record, 208
- getMinutiaeDataRecordSet
  - BiometricEvaluation::DataInterchange::AN2K↔  
Record, 179
  - BiometricEvaluation::Finger::AN2KView, 192
  - BiometricEvaluation::View::AN2KView, 186
- getMinutiaeReservedData
  - BiometricEvaluation::Finger::INCITSView, 388
- getMode
  - BiometricEvaluation::IO::Properties, 489
- getNFP
  - BiometricEvaluation::Feature::AN2K11EFS::Extended↔  
FeatureSet, 316
- getNISTQualityMetric
  - BiometricEvaluation::Finger::AN2KViewCapture,  
197
- getName
  - BiometricEvaluation::Process::Semaphore, 543
- getNames
  - BiometricEvaluation::IO::RecordStoreUnion, 529
- getNativeScanningResolution
  - BiometricEvaluation::DataInterchange::AN2K↔  
Record, 179
- getNextCommand
  - BiometricEvaluation::Process::CommandCenter,  
258
  - BiometricEvaluation::Process::CommandParser, 266
- getNextMessage
  - BiometricEvaluation::Process::Manager, 432
  - BiometricEvaluation::Process::MessageCenter, 437
- getNextValue
  - BiometricEvaluation::Image::NetPBM, 453
- getNominalTransmittingResolution
  - BiometricEvaluation::DataInterchange::AN2K↔  
Record, 179
- getNumActiveWorkers
  - BiometricEvaluation::Process::Manager, 432
- getNumCompletedWorkers
  - BiometricEvaluation::Process::Manager, 432
- getNumElements
  - BiometricEvaluation::MPI::WorkPackage, 607
- getNumFingerViews
  - BiometricEvaluation::DataInterchange::ANSI2004↔  
Record, 208
  - BiometricEvaluation::Finger::INCITSView, 389
- getNumLines
  - BiometricEvaluation::MPI::CSVResources, 292
- getNumRemainingLines
  - BiometricEvaluation::MPI::CSVResources, 293
- getNumThreads
  - BiometricEvaluation::Process::Statistics, 563
- getOption
  - BiometricEvaluation::IO::Compressor, 278
- getOptionAsInteger
  - BiometricEvaluation::IO::Compressor, 278
- getOptionalProperties
  - BiometricEvaluation::MPI::RecordStoreResources,  
525
  - BiometricEvaluation::MPI::Resources, 535
- getOriginatingAgency
  - BiometricEvaluation::DataInterchange::AN2K↔  
Record, 179
- getOriginatingFingerprintReadingSystem
  - BiometricEvaluation::Feature::AN2K7Minutiae, 171
- getPID
  - BiometricEvaluation::Process::ForkWorkerController,  
342
- getPalmQualityMetric
  - BiometricEvaluation::Palm::AN2KView, 182
- getParameter
  - BiometricEvaluation::Process::Worker, 597
- getParameterAsDouble
  - BiometricEvaluation::Process::Worker, 597
- getParameterAsInteger
  - BiometricEvaluation::Process::Worker, 597
- getParameterAsString
  - BiometricEvaluation::Process::Worker, 598
- getPathname
  - BiometricEvaluation::IO::ArchiveRecordStore, 227
  - BiometricEvaluation::IO::CompressedRecordStore,  
265
  - BiometricEvaluation::IO::DBRecordStore, 298
  - BiometricEvaluation::IO::FileLogCabinet, 318
  - BiometricEvaluation::IO::FileRecordStore, 328



- BiometricEvaluation::IO::ListRecordStore, 415
- BiometricEvaluation::IO::RecordStore, 509
- BiometricEvaluation::IO::SQLiteRecordStore, 556
- getPatternClassificationSet
  - BiometricEvaluation::Feature::AN2K7Minutiae, 171
- getPoseAngle
  - BiometricEvaluation::Face::INCITSView, 375
- getPosition
  - BiometricEvaluation::Finger::AN2KViewCapture, 197
  - BiometricEvaluation::Finger::INCITSView, 389
  - BiometricEvaluation::Palm::AN2KView, 182
- getPositionDescriptors
  - BiometricEvaluation::View::AN2KViewVariable↔ Resolution, 202
- getPositions
  - BiometricEvaluation::Finger::AN2KView, 192
  - BiometricEvaluation::Latent::AN2KView, 188
  - BiometricEvaluation::View::AN2KViewVariable↔ Resolution, 203
- getPrimitive
  - BiometricEvaluation::Device::TLV, 584
- getPrintPositionCoordinates
  - BiometricEvaluation::Finger::AN2KViewCapture, 197
  - BiometricEvaluation::Latent::AN2KView, 188
  - BiometricEvaluation::View::AN2KViewVariable↔ Resolution, 203
- getPriority
  - BiometricEvaluation::DataInterchange::AN2K↔ Record, 180
- getProductIDOwner
  - BiometricEvaluation::Finger::INCITSView, 389
- getProductIDType
  - BiometricEvaluation::Finger::INCITSView, 389
- getPropertiesFileName
  - BiometricEvaluation::MPI::Resources, 536
- getProperty
  - BiometricEvaluation::IO::Properties, 489
- getPropertyAsDouble
  - BiometricEvaluation::IO::Properties, 489
- getPropertyAsInteger
  - BiometricEvaluation::IO::Properties, 489
- getPropertyKeys
  - BiometricEvaluation::IO::Properties, 490
- getPropertySet
  - BiometricEvaluation::Face::INCITSView, 375
- getQuality
  - BiometricEvaluation::Finger::INCITSView, 389
- getQualityMetric
  - BiometricEvaluation::View::AN2KViewVariable↔ Resolution, 203
- getQualitySet
  - BiometricEvaluation::Iris::INCITSView, 382
- getRandomSeed
  - BiometricEvaluation::MPI::CSVResources, 293
- getRawData
  - BiometricEvaluation::Image::BMP, 252
  - BiometricEvaluation::Image::Image, 361, 362
  - BiometricEvaluation::Image::JPEG2000, 409
  - BiometricEvaluation::Image::JPEG, 411
  - BiometricEvaluation::Image::JPEG, 407
  - BiometricEvaluation::Image::NetPBM, 454
  - BiometricEvaluation::Image::PNG, 477
  - BiometricEvaluation::Image::Raw, 497
  - BiometricEvaluation::Image::TIFF, 576
  - BiometricEvaluation::Image::WSQ, 611
- getRawGrayscaleData
  - BiometricEvaluation::Image::BMP, 253
  - BiometricEvaluation::Image::Image, 362
  - BiometricEvaluation::Image::JPEG2000, 410
  - BiometricEvaluation::Image::JPEG, 412
  - BiometricEvaluation::Image::JPEG, 407
  - BiometricEvaluation::Image::NetPBM, 454
  - BiometricEvaluation::Image::PNG, 477
  - BiometricEvaluation::Image::Raw, 498
  - BiometricEvaluation::Image::TIFF, 577
  - BiometricEvaluation::Image::WSQ, 612
- getRawImage
  - BiometricEvaluation::Image::Image, 363
- getRawTLV
  - BiometricEvaluation::Device::TLV, 584
- getReaderID
  - BiometricEvaluation::Device::Smartcard, 553
- getRealMemorySize
  - BiometricEvaluation::System, 153
- getReceivingPipe
  - BiometricEvaluation::Process::Worker, 598
- getRecordLength
  - BiometricEvaluation::Finger::INCITSView, 390
- getRecordStore
  - BiometricEvaluation::IO::RecordStoreUnion, 530
  - BiometricEvaluation::MPI::RecordStoreResources, 525
- getRecordType
  - BiometricEvaluation::View::AN2KView, 186
- getRegisteredVendorBlock
  - BiometricEvaluation::Finger::AN2KMinutiaeData↔ Record, 174
- getRequiredProperties
  - BiometricEvaluation::MPI::RecordStoreResources, 526

- BiometricEvaluation::MPI::Resources, 536
- getResolution
  - BiometricEvaluation::Image::Image, 363
- getRollAngleInfo
  - BiometricEvaluation::Iris::INCITSView, 382
- getScanResolution
  - BiometricEvaluation::View::View, 589
- getSegmentationQualityMetric
  - BiometricEvaluation::Finger::AN2KViewCapture, 197
- getSendingPipe
  - BiometricEvaluation::Process::Worker, 599
- getSignalManager
  - BiometricEvaluation::Framework::API, 223
- getSize
  - BiometricEvaluation::MPI::WorkPackage, 607
  - BiometricEvaluation::Memory::IndexedBuffer, 397
- getSourceAgency
  - BiometricEvaluation::View::AN2KViewVariable↔ Resolution, 203
- getSourceType
  - BiometricEvaluation::Face::INCITSView, 375
- getSpaceUsed
  - BiometricEvaluation::IO::ArchiveRecordStore, 227
  - BiometricEvaluation::IO::CompressedRecordStore, 265
  - BiometricEvaluation::IO::DBRecordStore, 298
  - BiometricEvaluation::IO::FileRecordStore, 328
  - BiometricEvaluation::IO::ListRecordStore, 415
  - BiometricEvaluation::IO::RecordStore, 509
  - BiometricEvaluation::IO::SQLiteRecordStore, 556
- getString
  - BiometricEvaluation::Memory::AutoArrayUtility, 143
- getTagClass
  - BiometricEvaluation::Device::TLV, 584
- getTagNum
  - BiometricEvaluation::Device::TLV, 584
- getTimer
  - BiometricEvaluation::Framework::API, 223
- getTotalWorkers
  - BiometricEvaluation::Process::Manager, 433
- getTransactionControlNumber
  - BiometricEvaluation::DataInterchange::AN2K↔ Record, 180
- getTypeFromURL
  - BiometricEvaluation::IO::Logsheet, 424
- getUsage
  - BiometricEvaluation::Process::CommandParser, 260
- getUserDefinedField
  - BiometricEvaluation::View::AN2KViewVariable↔ Resolution, 203
- getVersionNumber
  - BiometricEvaluation::DataInterchange::AN2K↔ Record, 180
- getVideoStream
  - BiometricEvaluation::Video::Container, 283
- getView
  - BiometricEvaluation::DataInterchange::ANSI2004↔ Record, 208
- getViewNumber
  - BiometricEvaluation::Finger::INCITSView, 390
- getWatchdog
  - BiometricEvaluation::Framework::API, 223
- getWorker
  - BiometricEvaluation::Process::WorkerController, 603
- green
  - BiometricEvaluation::Image::BMP::ColorTable↔ Entry, 255
- hasAlphaChannel
  - BiometricEvaluation::Image::Image, 363
- hasPendingCommands
  - BiometricEvaluation::Process::CommandCenter, 258
- hasUnseenMessages
  - BiometricEvaluation::Process::MessageCenter, 438
- haveRecordStore
  - BiometricEvaluation::MPI::RecordStoreResources, 526
- INCITSMinutiae
  - BiometricEvaluation::Feature::INCITSMinutiae, 370
- INCITSView
  - BiometricEvaluation::Face::INCITSView, 372, 373
  - BiometricEvaluation::Finger::INCITSView, 386
  - BiometricEvaluation::Iris::INCITSView, 379
- INPUT\_DATA\_TYPE
  - BiometricEvaluation::IO::GZip, 355
- INPUTCSVPROPERTY
  - BiometricEvaluation::MPI::CSVResources, 294
- INVALIDKEYCHARS
  - BiometricEvaluation::IO::RecordStore, 517
- ISO2005View
  - BiometricEvaluation::Face::ISO2005View, 401
  - BiometricEvaluation::Finger::ISO2005View, 403
- ISO2011View
  - BiometricEvaluation::Iris::ISO2011View, 405, 406
- Identifier
  - BiometricEvaluation::DataInterchange::AN2K↔ Record::CharacterSet, 255



- BiometricEvaluation::DataInterchange::AN2K↔ isWSQ
  - Record::DomainName, 309
- Image
  - BiometricEvaluation::Image::Image, 358
- Impression
  - BiometricEvaluation::Finger, 117
- IndexedBuffer
  - BiometricEvaluation::Memory::IndexedBuffer, 396, 397
- initWithBuffer
  - BiometricEvaluation::IO::Properties, 490
- ins
  - BiometricEvaluation::Device::Smartcard::APDU, 218
- insert
  - BiometricEvaluation::IO::ArchiveRecordStore, 228
  - BiometricEvaluation::IO::CompressedRecordStore, 266
  - BiometricEvaluation::IO::DBRecordStore, 299
  - BiometricEvaluation::IO::FileRecordStore, 329
  - BiometricEvaluation::IO::ListRecordStore, 416
  - BiometricEvaluation::IO::RecordStore, 509, 510
  - BiometricEvaluation::IO::SQLiteRecordStore, 556
- insertView
  - BiometricEvaluation::DataInterchange::ANSI2004↔ Record, 209
- is\_default
  - BiometricEvaluation::Feature::AN2K11EFS::Orientation, 473
- isAppendixFCompliant
  - BiometricEvaluation::Finger::INCITSView, 390
- isBMP
  - BiometricEvaluation::Image::BMP, 253
- isJPEG2000
  - BiometricEvaluation::Image::JPEG2000, 410
- isJPEGL
  - BiometricEvaluation::Image::JPEGL, 412
- isJPEG
  - BiometricEvaluation::Image::JPEG, 408
- isLittleEndian
  - BiometricEvaluation::Memory, 140
- isNetPBM
  - BiometricEvaluation::Image::NetPBM, 455
- isPNG
  - BiometricEvaluation::Image::PNG, 478
- isPrimitive
  - BiometricEvaluation::Device::TLV, 584
- isReadable
  - BiometricEvaluation::IO::Utility, 132
- isTIFF
  - BiometricEvaluation::Image::TIFF, 577, 578
- BiometricEvaluation::Image::WSQ, 612
- isWorking
  - BiometricEvaluation::Process::ForkWorkerController, 343
  - BiometricEvaluation::Process::POSIXThreadWorker↔ Controller, 484
  - BiometricEvaluation::Process::WorkerController, 603
- isWritable
  - BiometricEvaluation::IO::Utility, 132
- isolateView
  - BiometricEvaluation::DataInterchange::ANSI2004↔ Record, 210
- iterator
  - BiometricEvaluation::Memory::AutoArray, 235
- iterator\_category
  - BiometricEvaluation::IO::RecordStoreIterator, 521
  - BiometricEvaluation::Memory::AutoArrayIterator, 245
  - BiometricEvaluation::Memory::OrderedMapConst↔ Iterator, 466
  - BiometricEvaluation::Memory::OrderedMapIterator, 470
- JPEG2000
  - BiometricEvaluation::Image::JPEG2000, 409
- key\_eq
  - BiometricEvaluation::Memory::OrderedMap, 464
- keyExists
  - BiometricEvaluation::Memory::OrderedMap, 464
- Kind
  - BiometricEvaluation::Feature::Sort, 114
  - BiometricEvaluation::IO::Compressor, 271
  - BiometricEvaluation::IO::Logsheet, 422
  - BiometricEvaluation::IO::RecordStore, 506
- lc
  - BiometricEvaluation::Device::Smartcard::APDU, 218
- le
  - BiometricEvaluation::Device::Smartcard::APDU, 218
- length
  - BiometricEvaluation::IO::ArchiveRecordStore, 228
  - BiometricEvaluation::IO::CompressedRecordStore, 266
  - BiometricEvaluation::IO::DBRecordStore, 299
  - BiometricEvaluation::IO::FileRecordStore, 329
  - BiometricEvaluation::IO::ListRecordStore, 416
  - BiometricEvaluation::IO::RecordStore, 510

- BiometricEvaluation::IO::RecordStoreUnion, 530
- BiometricEvaluation::IO::SQLiteRecordStore, 557
- lineIsComment
  - BiometricEvaluation::IO::Logsheet, 424
- lineIsDebug
  - BiometricEvaluation::IO::Logsheet, 425
- lineIsEntry
  - BiometricEvaluation::IO::Logsheet, 425
- ListRecordStore
  - BiometricEvaluation::IO::ListRecordStore, 414
- logEntry
  - BiometricEvaluation::MPI, 148
- logMessage
  - BiometricEvaluation::MPI, 148
- logStats
  - BiometricEvaluation::Process::Statistics, 564
- ltrim
  - BiometricEvaluation::Text, 157
- ltrimWhitespace
  - BiometricEvaluation::Text, 158
- MANIFEST\_FILE\_NAME
  - BiometricEvaluation::IO::ArchiveRecordStore, 233
- MAX\_MESSAGE\_LENGTH
  - BiometricEvaluation::Process::MessageCenter, 439
- MEMORY\_LEVEL
  - BiometricEvaluation::IO::GZip, 355
- MSG\_DISCONNECT
  - BiometricEvaluation::Process::MessageCenterReceiver, 441
- make\_unique
  - BiometricEvaluation::Memory, 140, 141
- makePath
  - BiometricEvaluation::IO::Utility, 133
- mdu
  - BiometricEvaluation::Feature::AN2K11EFS::Minutiae↔Point, 445
- MemoryError
  - BiometricEvaluation::Error::MemoryError, 436
- mergeLogsheets
  - BiometricEvaluation::IO::FileLogsheet, 322
- mergeRecordStores
  - BiometricEvaluation::IO::RecordStore, 511
- MessageCenter
  - BiometricEvaluation::Process::MessageCenter, 437
- MessageCenterReceiver
  - BiometricEvaluation::Process::MessageCenterReceiver, 441
- MessageTag
  - BiometricEvaluation::MPI, 147
- method
  - BiometricEvaluation::Feature::AN2K7Minutiae↔::FingerprintReadingSystem, 334
  - BiometricEvaluation::Feature::AN2K11EFS::Minutiae↔RidgeCount, 443
  - BiometricEvaluation::Feature::AN2K11EFS::Minutiae↔RidgeCount, 443
  - BiometricEvaluation::Image, 126
  - BiometricEvaluation::Feature::AN2K11EFS::Minutiae↔RidgeCount, 443
  - BiometricEvaluation::IO, 127
  - BiometricEvaluation::IO::ArchiveRecordStore, 229
  - BiometricEvaluation::IO::CompressedRecordStore, 267
  - BiometricEvaluation::IO::DBRecordStore, 300
  - BiometricEvaluation::IO::FileRecordStore, 330
  - BiometricEvaluation::IO::ListRecordStore, 417
  - BiometricEvaluation::IO::RecordStore, 511
  - BiometricEvaluation::IO::SQLiteRecordStore, 557
  - BiometricEvaluation::Feature::AN2K11EFS::Minutiae↔RidgeCount, 443
  - BiometricEvaluation::Feature::AN2K11EFS::Minutiae↔RidgeCount, 443
  - BiometricEvaluation::Feature::AN2K11EFS::Minutiae↔Point, 445
  - BiometricEvaluation::MPI, 146
  - BiometricEvaluation::Memory::MutableIndexed↔Buffer, 446, 447
  - BiometricEvaluation::Device::Smartcard::APDU, 218
  - needsVacuum
- mia
- mib
- MillimetersPerInch
- mir
- Mode
- move
- mrn
- mrs
- mrp
- msgtag\_t
- MutableIndexedBuffer
- name
- nc

- BiometricEvaluation::IO::ArchiveRecordStore, 229
- newEntry
  - BiometricEvaluation::IO::Logsheet, 425
- newLogsheet
  - BiometricEvaluation::IO::FileLogCabinet, 318
- newProcessor
  - BiometricEvaluation::MPI::CSVProcessor, 289
  - BiometricEvaluation::MPI::RecordProcessor, 502
  - BiometricEvaluation::MPI::WorkPackageProcessor, 609
- NotImplemented
  - BiometricEvaluation::Error::NotImplemented, 457
- OFFSET\_RECORD\_REMOVED
  - BiometricEvaluation::IO::ArchiveRecordStore, 233
- ObjectDoesNotExist
  - BiometricEvaluation::Error::ObjectDoesNotExist, 458
- ObjectExists
  - BiometricEvaluation::Error::ObjectExists, 458, 459
- ObjectIsClosed
  - BiometricEvaluation::Error::ObjectIsClosed, 459
- ObjectIsOpen
  - BiometricEvaluation::Error::ObjectIsOpen, 460
- ocf
  - BiometricEvaluation::Feature::AN2K11EFS::F←PPPosition, 346
- OK
  - BiometricEvaluation::Framework::Status, 566
- openImage
  - BiometricEvaluation::Image::Image, 364, 365
- openLogsheet
  - BiometricEvaluation::MPI, 148
- openRecordStore
  - BiometricEvaluation::IO::RecordStore, 512
- operator bool
  - BiometricEvaluation::Framework::API::Result, 537
- operator const T\*
  - BiometricEvaluation::Memory::AutoArray, 240
- operator T\*
  - BiometricEvaluation::Memory::AutoArray, 240
- operator!
  - BiometricEvaluation::Framework::API::Result, 538
- operator!=
  - BiometricEvaluation::IO::RecordStoreIterator, 522
  - BiometricEvaluation::Memory, 141
  - BiometricEvaluation::Memory::AutoArrayIterator, 246
  - BiometricEvaluation::Memory::OrderedMapConst←Iterator, 467
  - BiometricEvaluation::Memory::OrderedMapIterator, 471
- operator<
  - BiometricEvaluation::Memory, 141
  - BiometricEvaluation::Memory::AutoArrayIterator, 248
- operator<<
  - BiometricEvaluation::Feature, 113
  - BiometricEvaluation::Framework, 119
  - BiometricEvaluation::Image, 123
  - BiometricEvaluation::Time, 162
  - BiometricEvaluation::View, 165
- operator<=
  - BiometricEvaluation::Memory, 142
  - BiometricEvaluation::Memory::AutoArrayIterator, 249
- operator>
  - BiometricEvaluation::Memory, 142
  - BiometricEvaluation::Memory::AutoArrayIterator, 249
- operator>=
  - BiometricEvaluation::Memory, 142
  - BiometricEvaluation::Memory::AutoArrayIterator, 250
- operator\*
  - BiometricEvaluation::IO::RecordStoreIterator, 522
  - BiometricEvaluation::Memory::AutoArrayIterator, 246
  - BiometricEvaluation::Memory::OrderedMapConst←Iterator, 468
  - BiometricEvaluation::Memory::OrderedMapIterator, 471
- operator()
  - BiometricEvaluation::Feature::Sort::Angle, 205
  - BiometricEvaluation::Feature::Sort::Polar, 480
- operator+
  - BiometricEvaluation::IO::RecordStoreIterator, 522
  - BiometricEvaluation::Memory::AutoArrayIterator, 246, 250
- operator++
  - BiometricEvaluation::IO::RecordStoreIterator, 523
  - BiometricEvaluation::Memory::AutoArrayIterator, 247
  - BiometricEvaluation::Memory::OrderedMapConst←Iterator, 468
  - BiometricEvaluation::Memory::OrderedMapIterator, 471, 472
- operator+=
  - BiometricEvaluation::IO::RecordStoreIterator, 523
  - BiometricEvaluation::Memory::AutoArrayIterator, 247
- operator-
  - BiometricEvaluation::Memory::AutoArrayIterator,

- 247, 250
- operator->  
 BiometricEvaluation::IO::RecordStoreIterator, 523  
 BiometricEvaluation::Memory::AutoArrayIterator, 248  
 BiometricEvaluation::Memory::OrderedMapConstIterator, 469  
 BiometricEvaluation::Memory::OrderedMapIterator, 472
- operator--  
 BiometricEvaluation::Memory::AutoArrayIterator, 248  
 BiometricEvaluation::Memory::OrderedMapConstIterator, 468  
 BiometricEvaluation::Memory::OrderedMapIterator, 472
- operator-=  
 BiometricEvaluation::Memory::AutoArrayIterator, 248
- operator=  
 BiometricEvaluation::Device::Smartcard, 553  
 BiometricEvaluation::IO::CompressedRecordStore, 267  
 BiometricEvaluation::IO::Compressor, 280  
 BiometricEvaluation::IO::FileLogsheet, 322  
 BiometricEvaluation::IO::GZip, 354  
 BiometricEvaluation::IO::PropertiesFile, 495  
 BiometricEvaluation::IO::RecordStoreIterator, 524  
 BiometricEvaluation::IO::SysLogsheet, 572  
 BiometricEvaluation::Memory::AutoArray, 240, 241  
 BiometricEvaluation::Memory::AutoArrayIterator, 249
- operator==  
 BiometricEvaluation::IO::RecordStoreIterator, 524  
 BiometricEvaluation::Memory, 142  
 BiometricEvaluation::Memory::AutoArrayIterator, 249  
 BiometricEvaluation::Memory::OrderedMapConstIterator, 469  
 BiometricEvaluation::Memory::OrderedMapIterator, 472
- operator[]  
 BiometricEvaluation::Memory::AutoArray, 241, 242  
 BiometricEvaluation::Memory::AutoArrayIterator, 250  
 BiometricEvaluation::Memory::OrderedMap, 464
- OrderedMap  
 BiometricEvaluation::Memory::OrderedMap, 461
- OrderedMapConstIterator  
 BiometricEvaluation::Memory::OrderedMapConstIterator, 467  
 BiometricEvaluation::Memory::OrderedMapIterator, 471  
 BiometricEvaluation::Feature::AN2K11EFS::ImageInfo, 367  
 BiometricEvaluation::Device::Smartcard::APDU, 218  
 BiometricEvaluation::Device::Smartcard::APDU, 218  
 PARAM\_CLIENT\_ID  
 BiometricEvaluation::Process::MessageCenterReceiver, 441  
 PARAM\_CLIENT\_SOCKET  
 BiometricEvaluation::Process::MessageCenterReceiver, 441  
 PARAM\_PORT  
 BiometricEvaluation::Process::MessageCenterListener, 440  
 POSIXThreadManager  
 BiometricEvaluation::Process::POSIXThreadManager, 481  
 PROCESSTIME  
 BiometricEvaluation::Time::Watchdog, 594  
 ParameterError  
 BiometricEvaluation::Error::ParameterError, 474  
 ParameterList  
 BiometricEvaluation::Process, 151
- parse  
 BiometricEvaluation::Process::CommandParser, 261
- parseUserDefinedField  
 BiometricEvaluation::View::AN2KViewVariableResolution, 204
- PatternClassification  
 BiometricEvaluation::Finger, 117
- performInitialization  
 BiometricEvaluation::MPI::CSVProcessor, 290  
 BiometricEvaluation::MPI::RecordProcessor, 502  
 BiometricEvaluation::MPI::WorkPackageProcessor, 609
- performShutdown  
 BiometricEvaluation::MPI::WorkPackageProcessor, 610
- PersistentRecordStoreUnion  
 BiometricEvaluation::IO::PersistentRecordStoreUnion, 475, 476
- PixelFormat

- BiometricEvaluation::Image, 122
- plr
  - BiometricEvaluation::Feature::AN2K11EFS::Image↔ Info, 367
- pointer
  - BiometricEvaluation::IO::RecordStoreIterator, 521
  - BiometricEvaluation::Memory::AutoArrayIterator, 245
  - BiometricEvaluation::Memory::OrderedMapConst↔ Iterator, 466
  - BiometricEvaluation::Memory::OrderedMapIterator, 470
- Polar
  - BiometricEvaluation::Feature::Sort::Polar, 479
- populateFGP
  - BiometricEvaluation::Finger::AN2KView, 192
- Position
  - BiometricEvaluation::Finger, 117
  - BiometricEvaluation::Palm, 150
  - BiometricEvaluation::Plantar, 150
- post
  - BiometricEvaluation::Process::Semaphore, 543
- printStatus
  - BiometricEvaluation::MPI, 149
- processLine
  - BiometricEvaluation::MPI::CSVProcessor, 290
- processRecord
  - BiometricEvaluation::MPI::RecordProcessor, 503
- processWorkPackage
  - BiometricEvaluation::MPI::CSVProcessor, 291
  - BiometricEvaluation::MPI::RecordProcessor, 503
  - BiometricEvaluation::MPI::WorkPackageProcessor, 610
- Properties
  - BiometricEvaluation::IO::Properties, 488
- propertiesConsidered
  - BiometricEvaluation::Face::INCITSView, 376
- PropertiesFile
  - BiometricEvaluation::IO::PropertiesFile, 494, 495
- PropertySet
  - BiometricEvaluation::Face, 111
- push
  - BiometricEvaluation::Memory::MutableIndexed↔ Buffer, 447
- push\_back
  - BiometricEvaluation::Memory::OrderedMap, 465
- pushBeU16Val
  - BiometricEvaluation::Memory::MutableIndexed↔ Buffer, 448
- pushBeU32Val
  - BiometricEvaluation::Memory::MutableIndexed↔ Buffer, 448
- pushU16Val
  - BiometricEvaluation::Memory::MutableIndexed↔ Buffer, 449
- pushU32Val
  - BiometricEvaluation::Memory::MutableIndexed↔ Buffer, 449
- pushU64Val
  - BiometricEvaluation::Memory::MutableIndexed↔ Buffer, 450
- pushU8Val
  - BiometricEvaluation::Memory::MutableIndexed↔ Buffer, 450
- put\_time
  - BiometricEvaluation::Time, 163
- RANDOMIZEPROPERTY
  - BiometricEvaluation::MPI::CSVResources, 294
- RANDOMSEEDPROPERTY
  - BiometricEvaluation::MPI::CSVResources, 294
- REALTIME
  - BiometricEvaluation::Time::Watchdog, 594
- ROI
  - BiometricEvaluation::Image::ROI, 539
- randomizeLines
  - BiometricEvaluation::MPI::CSVResources, 293
- read
  - BiometricEvaluation::IO::ArchiveRecordStore, 230
  - BiometricEvaluation::IO::CompressedRecordStore, 267
  - BiometricEvaluation::IO::DBRecordStore, 300
  - BiometricEvaluation::IO::FileRecordStore, 330
  - BiometricEvaluation::IO::ListRecordStore, 417
  - BiometricEvaluation::IO::RecordStore, 512
  - BiometricEvaluation::IO::RecordStoreUnion, 530
  - BiometricEvaluation::IO::SQLiteRecordStore, 558
- readCoreDeltaData
  - BiometricEvaluation::Finger::ANSI2004View, 214
  - BiometricEvaluation::Finger::ANSI2007View, 216
  - BiometricEvaluation::Finger::INCITSView, 390
  - BiometricEvaluation::Finger::ISO2005View, 404
- readExtendedDataBlock
  - BiometricEvaluation::Finger::INCITSView, 391
- readFMRHeader
  - BiometricEvaluation::Finger::INCITSView, 391
- readFVMR
  - BiometricEvaluation::Finger::INCITSView, 391
- readFaceView
  - BiometricEvaluation::Face::INCITSView, 376
- readFile
  - BiometricEvaluation::IO::Utility, 133
- readHeader

- BiometricEvaluation::Face::INCITSView, [376](#)
- BiometricEvaluation::Iris::INCITSView, [383](#)
- readISOHeader
  - BiometricEvaluation::Face::ISO2005View, [402](#)
- readIrisView
  - BiometricEvaluation::Iris::INCITSView, [383](#)
- readLine
  - BiometricEvaluation::MPI::CSVResources, [293](#)
- readMinutiaeDataPoints
  - BiometricEvaluation::Finger::INCITSView, [392](#)
- readPipe
  - BiometricEvaluation::IO::Utility, [134](#)
- readRidgeCountData
  - BiometricEvaluation::Finger::INCITSView, [392](#)
- receiveMessageFromManager
  - BiometricEvaluation::Process::Worker, [599](#)
- Receiver
  - BiometricEvaluation::MPI::Receiver, [499](#)
- Record
  - BiometricEvaluation::IO::RecordStore::Record, [500](#)
- recordLocations
  - BiometricEvaluation::DataInterchange::AN2K↔
    - Record, [180](#), [181](#)
- RecordProcessor
  - BiometricEvaluation::MPI::RecordProcessor, [501](#)
- RecordStoreDistributor
  - BiometricEvaluation::MPI::RecordStoreDistributor↔
    - [518](#)
- RecordStoreIterator
  - BiometricEvaluation::IO::RecordStoreIterator, [521](#), [522](#)
- RecordStoreResources
  - BiometricEvaluation::MPI::RecordStoreResources, [525](#)
- RecordStoreUnion
  - BiometricEvaluation::IO::RecordStoreUnion, [527](#)–[529](#)
- RecordType
  - BiometricEvaluation::View::AN2KView, [184](#)
- red
  - BiometricEvaluation::Image::BMP::ColorTable↔
    - Entry, [255](#)
- reference
  - BiometricEvaluation::IO::RecordStoreIterator, [521](#)
  - BiometricEvaluation::Memory::AutoArray, [235](#)
  - BiometricEvaluation::Memory::AutoArrayIterator, [245](#)
  - BiometricEvaluation::Memory::OrderedMapConst↔
    - Iterator, [467](#)
  - BiometricEvaluation::Memory::OrderedMapIterator, [470](#)
- remove
  - BiometricEvaluation::IO::ArchiveRecordStore, [230](#)
  - BiometricEvaluation::IO::CompressedRecordStore, [268](#)
  - BiometricEvaluation::IO::DBRecordStore, [301](#)
  - BiometricEvaluation::IO::FileRecordStore, [331](#)
  - BiometricEvaluation::IO::ListRecordStore, [418](#)
  - BiometricEvaluation::IO::RecordStore, [513](#)
  - BiometricEvaluation::IO::SQLiteRecordStore, [558](#)
- removeComponents
  - BiometricEvaluation::Image, [123](#)
- removeDirectory
  - BiometricEvaluation::IO::Utility, [135](#)
- removeOption
  - BiometricEvaluation::IO::Compressor, [280](#)
- removeProperty
  - BiometricEvaluation::IO::Properties, [491](#)
- removeRecordStore
  - BiometricEvaluation::IO::RecordStore, [513](#)
- removeView
  - BiometricEvaluation::DataInterchange::ANSI2004↔
    - Record, [210](#)
- replace
  - BiometricEvaluation::IO::FileRecordStore, [331](#)
  - BiometricEvaluation::IO::ListRecordStore, [418](#)
  - BiometricEvaluation::IO::RecordStore, [514](#)
- reserved
  - BiometricEvaluation::Image::BMP::ColorTable↔
    - Entry, [255](#)
- reset
  - BiometricEvaluation::Process::ForkWorkerController, [343](#)
- resetCurrentEntry
  - BiometricEvaluation::IO::Logsheet, [426](#)
- resize
  - BiometricEvaluation::Memory::AutoArray, [242](#)
- Resolution
  - BiometricEvaluation::Image::Resolution, [532](#)
- Resources
  - BiometricEvaluation::MPI::Resources, [535](#)
- response
  - BiometricEvaluation::Device::Smartcard::APD↔
    - UException, [220](#)
- responsibleFor
  - BiometricEvaluation::Process::ForkManager, [337](#)
- Result



- BiometricEvaluation::Framework::API::Result, 537
- roi
  - BiometricEvaluation::Feature::AN2K11EFS::Image↔ Info, 367
- rtrim
  - BiometricEvaluation::Text, 158
- rtrimWhitespace
  - BiometricEvaluation::Text, 158
- Runtime
  - BiometricEvaluation::MPI::Runtime, 540
- SYSLOGURLSCHEME
  - BiometricEvaluation::IO::Logsheet, 429
- scan
  - BiometricEvaluation::Memory::IndexedBuffer, 397
- scanBeU16Val
  - BiometricEvaluation::Memory::IndexedBuffer, 398
- scanBeU32Val
  - BiometricEvaluation::Memory::IndexedBuffer, 398
- scanU16Val
  - BiometricEvaluation::Memory::IndexedBuffer, 399
- scanU32Val
  - BiometricEvaluation::Memory::IndexedBuffer, 399
- scanU64Val
  - BiometricEvaluation::Memory::IndexedBuffer, 399
- scanU8Val
  - BiometricEvaluation::Memory::IndexedBuffer, 399
- segment
  - BiometricEvaluation::View::AN2KViewVariable↔ Resolution::PrintPositionCoordinate, 486
- Semaphore
  - BiometricEvaluation::Process::Semaphore, 542, 543
- sendAPDU
  - BiometricEvaluation::Device::Smartcard, 553
- sendMessageToManager
  - BiometricEvaluation::Process::Worker, 599
- sendMessageToWorker
  - BiometricEvaluation::Process::WorkerController, 604
- sendResponse
  - BiometricEvaluation::Process::CommandCenter, 259
  - BiometricEvaluation::Process::MessageCenter, 438
- sequence
  - BiometricEvaluation::IO::ArchiveRecordStore, 231
  - BiometricEvaluation::IO::CompressedRecordStore, 268
  - BiometricEvaluation::IO::DBRecordStore, 301
  - BiometricEvaluation::IO::FileLogsheet, 322
  - BiometricEvaluation::IO::FileRecordStore, 331
  - BiometricEvaluation::IO::ListRecordStore, 418
  - BiometricEvaluation::IO::RecordStore, 515
  - BiometricEvaluation::IO::SQLiteRecordStore, 559
  - sequenceKey
    - BiometricEvaluation::IO::ArchiveRecordStore, 231
    - BiometricEvaluation::IO::CompressedRecordStore, 269
    - BiometricEvaluation::IO::DBRecordStore, 301
    - BiometricEvaluation::IO::FileRecordStore, 332
    - BiometricEvaluation::IO::ListRecordStore, 419
    - BiometricEvaluation::IO::RecordStore, 515
    - BiometricEvaluation::IO::SQLiteRecordStore, 559
  - setAppendixFCompliance
    - BiometricEvaluation::Finger::INCITSView, 393
  - setAsideName
    - BiometricEvaluation::IO::Utility, 135
  - setAutoSync
    - BiometricEvaluation::IO::Logsheet, 426
  - setBitDepth
    - BiometricEvaluation::Image::Image, 365
  - setCBEFFProductIDs
    - BiometricEvaluation::Finger::INCITSView, 393
  - setCanSigJump
    - BiometricEvaluation::Time::Watchdog, 593
  - setCaptureEquipmentID
    - BiometricEvaluation::Finger::INCITSView, 393
  - setColorDepth
    - BiometricEvaluation::Image::Image, 365
  - setCommentCommit
    - BiometricEvaluation::IO::Logsheet, 426
  - setCommit
    - BiometricEvaluation::IO::Logsheet, 426
  - setCorePointSet
    - BiometricEvaluation::Feature::INCITSMinutiae, 370
  - setCursorAtKey
    - BiometricEvaluation::IO::ArchiveRecordStore, 232
    - BiometricEvaluation::IO::CompressedRecordStore, 269
    - BiometricEvaluation::IO::DBRecordStore, 302
    - BiometricEvaluation::IO::FileRecordStore, 332
    - BiometricEvaluation::IO::ListRecordStore, 419
    - BiometricEvaluation::IO::RecordStore, 516
    - BiometricEvaluation::IO::SQLiteRecordStore, 560
  - setData
    - BiometricEvaluation::MPI::WorkPackage, 607
  - setDebugCommit
    - BiometricEvaluation::IO::Logsheet, 427
  - setDefaultSignalSet
    - BiometricEvaluation::Error::SignalManager, 547
  - setDeltaPointSet
    - BiometricEvaluation::Feature::INCITSMinutiae, 370
  - setDimensions
    - BiometricEvaluation::Image::Image, 366

- setDryrun
  - BiometricEvaluation::Device::Smartcard, 553
- setExitCallback
  - BiometricEvaluation::Process::ForkManager, 338
- setExitStatus
  - BiometricEvaluation::Process::ForkManager, 338
- setExpired
  - BiometricEvaluation::Time::Watchdog, 593
- setFramePixelFormat
  - BiometricEvaluation::Video::Stream, 569
- setFrameScale
  - BiometricEvaluation::Video::Stream, 569
- setHasAlphaChannel
  - BiometricEvaluation::Image::Image, 366
- setImageColorDepth
  - BiometricEvaluation::View::View, 590
- setImageData
  - BiometricEvaluation::View::View, 590
- setImageResolution
  - BiometricEvaluation::View::View, 590
- setImageSize
  - BiometricEvaluation::View::View, 590
- setImpl
  - BiometricEvaluation::IO::RecordStoreUnion, 531
- setImpressionType
  - BiometricEvaluation::Finger::AN2KView, 193
  - BiometricEvaluation::Finger::INCITSView, 393
- setIndex
  - BiometricEvaluation::Memory::IndexedBuffer, 400
- setInterval
  - BiometricEvaluation::Time::Watchdog, 593
- setLogsheet
  - BiometricEvaluation::MPI::WorkPackageProcessor, 610
- setMinutia
  - BiometricEvaluation::DataInterchange::ANSI2004Record, 210, 211
- setMinutiaPoints
  - BiometricEvaluation::Feature::INCITSMinutiae, 370
- setMinutiaeData
  - BiometricEvaluation::Finger::INCITSView, 394
- setMinutiaeReservedData
  - BiometricEvaluation::Finger::INCITSView, 394
- setNotWorking
  - BiometricEvaluation::Process::ForkManager, 339
- setNumElements
  - BiometricEvaluation::MPI::WorkPackage, 607
- setOption
  - BiometricEvaluation::IO::Compressor, 280, 281
- setParameter
  - BiometricEvaluation::Process::Worker, 600
- BiometricEvaluation::Process::WorkerController, 604
- setParameterFromDouble
  - BiometricEvaluation::Process::WorkerController, 605
- setParameterFromInteger
  - BiometricEvaluation::Process::WorkerController, 605
- setParameterFromString
  - BiometricEvaluation::Process::WorkerController, 605
- setPosition
  - BiometricEvaluation::Finger::INCITSView, 394
- setPositions
  - BiometricEvaluation::Finger::AN2KView, 193
- setPrimitive
  - BiometricEvaluation::Device::TLV, 585
- setProperty
  - BiometricEvaluation::IO::Properties, 491
- setPropertyFromBoolean
  - BiometricEvaluation::IO::Properties, 492
- setPropertyFromDouble
  - BiometricEvaluation::IO::Properties, 492
- setPropertyFromInteger
  - BiometricEvaluation::IO::Properties, 493
- setQuality
  - BiometricEvaluation::Finger::INCITSView, 394
- setResolution
  - BiometricEvaluation::Image::Image, 366
- setRidgeCountItems
  - BiometricEvaluation::Feature::INCITSMinutiae, 371
- setScanResolution
  - BiometricEvaluation::View::View, 591
- setSigHandled
  - BiometricEvaluation::Error::SignalManager, 547
- setSignalSet
  - BiometricEvaluation::Error::SignalManager, 547
- setString
  - BiometricEvaluation::Memory::AutoArrayUtility, 144
- setTag
  - BiometricEvaluation::Device::TLV, 585
- setUsage
  - BiometricEvaluation::Process::CommandParser, 261
- setViewNumber
  - BiometricEvaluation::Finger::INCITSView, 395
- setup
  - BiometricEvaluation::IO::SysLogsheet, 572
- sgp
  - BiometricEvaluation::Feature::AN2K11EFS::F←PPPosition, 346



- shutdown
  - BiometricEvaluation::MPI::Runtime, [541](#)
- sigHandled
  - BiometricEvaluation::Error::SignalManager, [547](#)
- SignalManager
  - BiometricEvaluation::Error::SignalManager, [546](#)
- Size
  - BiometricEvaluation::Image::Size, [549](#)
- size
  - BiometricEvaluation::Memory::AutoArray, [242](#)
  - BiometricEvaluation::Memory::OrderedMap, [465](#)
- size\_type
  - BiometricEvaluation::Memory::AutoArray, [235](#)
- skipComment
  - BiometricEvaluation::Image::NetPBM, [455](#)
- skipLine
  - BiometricEvaluation::Image::NetPBM, [456](#)
- Smartcard
  - BiometricEvaluation::Device::Smartcard, [550](#), [551](#)
- sort
  - BiometricEvaluation::Feature::Sort, [114](#)
- split
  - BiometricEvaluation::Text, [159](#)
- stableSort
  - BiometricEvaluation::Feature::Sort, [115](#)
- standard
  - BiometricEvaluation::Feature::AN2K7Minutiae↵  
::PatternClassification::Entry, [310](#)
- start
  - BiometricEvaluation::Error::SignalManager, [548](#)
  - BiometricEvaluation::MPI::Distributor, [307](#)
  - BiometricEvaluation::MPI::Receiver, [499](#)
  - BiometricEvaluation::MPI::Runtime, [541](#)
  - BiometricEvaluation::Time::Timer, [580](#)
  - BiometricEvaluation::Time::Watchdog, [594](#)
- startAutoLogging
  - BiometricEvaluation::Process::Statistics, [564](#)
- startWorker
  - BiometricEvaluation::Process::ForkManager, [339](#)
  - BiometricEvaluation::Process::Manager, [433](#)
  - BiometricEvaluation::Process::POSIXThreadManager, [482](#)
- startWorkers
  - BiometricEvaluation::Process::ForkManager, [339](#)
  - BiometricEvaluation::Process::Manager, [434](#)
  - BiometricEvaluation::Process::POSIXThreadManager, [482](#)
- Statistics
  - BiometricEvaluation::Process::Statistics, [561](#), [562](#)
- Status
  - BiometricEvaluation::Framework::Status, [565](#)
- status
  - BiometricEvaluation::Framework::API::Result, [538](#)
- stop
  - BiometricEvaluation::Error::SignalManager, [548](#)
  - BiometricEvaluation::Time::Timer, [581](#)
  - BiometricEvaluation::Time::Watchdog, [594](#)
- stopAutoLogging
  - BiometricEvaluation::Process::Statistics, [565](#)
- stopRequested
  - BiometricEvaluation::Process::Worker, [600](#)
- stopWorker
  - BiometricEvaluation::Process::ForkManager, [340](#)
  - BiometricEvaluation::Process::Manager, [434](#)
  - BiometricEvaluation::Process::POSIXThreadManager, [483](#)
- StrategyError
  - BiometricEvaluation::Error::StrategyError, [567](#)
- stringFromTLV
  - BiometricEvaluation::Device::TLV, [585](#)
- sumDirectoryUsage
  - BiometricEvaluation::IO::Utility, [136](#)
- sw1
  - BiometricEvaluation::Device::Smartcard::APD↵  
UResponse, [221](#)
- sw2
  - BiometricEvaluation::Device::Smartcard::APD↵  
UResponse, [221](#)
- sync
  - BiometricEvaluation::IO::ArchiveRecordStore, [232](#)
  - BiometricEvaluation::IO::CompressedRecordStore, [270](#)
  - BiometricEvaluation::IO::DBRecordStore, [302](#)
  - BiometricEvaluation::IO::FileLogsheet, [323](#)
  - BiometricEvaluation::IO::FileRecordStore, [333](#)
  - BiometricEvaluation::IO::ListRecordStore, [420](#)
  - BiometricEvaluation::IO::Logsheet, [427](#)
  - BiometricEvaluation::IO::PropertiesFile, [496](#)
  - BiometricEvaluation::IO::RecordStore, [516](#)
  - BiometricEvaluation::IO::SQLiteRecordStore, [560](#)
  - BiometricEvaluation::IO::SysLogsheet, [573](#)
- SysLogsheet
  - BiometricEvaluation::IO::SysLogsheet, [571](#), [572](#)
- TLV
  - BiometricEvaluation::Device::TLV, [582](#), [583](#)
- TRIMPROPERTY
  - BiometricEvaluation::MPI::CSVResources, [294](#)
- TaskCommand
  - BiometricEvaluation::MPI, [147](#)
- TaskStatus
  - BiometricEvaluation::MPI, [147](#)
- taskcmd\_t

- BiometricEvaluation::MPI, [146](#)
- taskstat\_t
  - BiometricEvaluation::MPI, [146](#)
- TerminateJob
  - BiometricEvaluation::MPI::TerminateJob, [575](#)
- time
  - BiometricEvaluation::Time::Timer, [581](#)
- timedwait
  - BiometricEvaluation::Process::Semaphore, [543](#)
- Timer
  - BiometricEvaluation::Time::Timer, [579](#)
- to\_string
  - BiometricEvaluation::Framework, [120](#)
  - BiometricEvaluation::Image, [124](#), [125](#)
- to\_vector
  - BiometricEvaluation::Memory::AutoArray, [243](#)
- toLowerCase
  - BiometricEvaluation::Text, [159](#)
- toUnits
  - BiometricEvaluation::Image::Resolution, [533](#)
- toUpperCase
  - BiometricEvaluation::Text, [160](#)
- trim
  - BiometricEvaluation::IO::FileLogsheet, [323](#)
  - BiometricEvaluation::IO::Logsheet, [427](#)
  - BiometricEvaluation::Text, [160](#)
- trimWhitespace
  - BiometricEvaluation::Text, [160](#)
- trv
  - BiometricEvaluation::Feature::AN2K11EFS::ImageWaitInfo, [368](#)
- trywait
  - BiometricEvaluation::Process::Semaphore, [544](#)
- USEBUFFERPROPERTY
  - BiometricEvaluation::MPI::CSVResources, [294](#)
- unique\_array\_known\_bound
  - BiometricEvaluation::Memory::unique\_if< T[S]>, [587](#)
- unique\_array\_unknown\_bound
  - BiometricEvaluation::Memory::unique\_if< T[]>, [587](#)
- unique\_single
  - BiometricEvaluation::Memory::unique\_if, [586](#)
- Units
  - BiometricEvaluation::Image::Resolution, [532](#)
- units
  - BiometricEvaluation::Image::Resolution, [533](#)
- updateCursor
  - BiometricEvaluation::IO::FileLogsheet, [323](#)
- updateView
  - BiometricEvaluation::DataInterchange::ANSI2004↔Record, [211](#)
- useBuffer
  - BiometricEvaluation::MPI::CSVResources, [294](#)
- vacuum
  - BiometricEvaluation::IO::ArchiveRecordStore, [232](#)
- value\_type
  - BiometricEvaluation::IO::RecordStoreIterator, [521](#)
  - BiometricEvaluation::Memory::AutoArray, [235](#)
  - BiometricEvaluation::Memory::AutoArrayIterator, [245](#)
  - BiometricEvaluation::Memory::AutoBuffer, [251](#)
  - BiometricEvaluation::Memory::OrderedMapConst↔Iterator, [467](#)
  - BiometricEvaluation::Memory::OrderedMapIterator, [470](#)
- valueInColorspace
  - BiometricEvaluation::Image::Image, [366](#)
- version
  - BiometricEvaluation::DataInterchange::AN2K↔Record::CharacterSet, [255](#)
  - BiometricEvaluation::DataInterchange::AN2K↔Record::DomainName, [309](#)
- WINDOW\_BITS
  - BiometricEvaluation::IO::GZip, [355](#)
- WORKERSPERNODEPROPERTY
  - BiometricEvaluation::MPI::Resources, [536](#)
- wait
  - BiometricEvaluation::Process::Semaphore, [544](#)
- waitForMessage
  - BiometricEvaluation::Process::Manager, [435](#)
  - BiometricEvaluation::Process::Worker, [600](#)
- waitForWorkerExit
  - BiometricEvaluation::Process::ForkManager, [340](#)
  - BiometricEvaluation::Process::Manager, [435](#)
  - BiometricEvaluation::Process::POSIXThreadManager, [483](#)
- Watchdog
  - BiometricEvaluation::Time::Watchdog, [592](#)
- what
  - BiometricEvaluation::Error::Exception, [312](#)
- whatString
  - BiometricEvaluation::Error::Exception, [312](#)
- WorkPackage
  - BiometricEvaluation::MPI::WorkPackage, [607](#)
- WorkerController
  - BiometricEvaluation::Process::WorkerController, [602](#)
- workerMain

- BiometricEvaluation::Process::MessageCenterListener,  
[439](#)
- BiometricEvaluation::Process::MessageCenterReceiver,  
[441](#)
- BiometricEvaluation::Process::Worker, [601](#)
- write
  - BiometricEvaluation::IO::FileLogsheet, [323](#)
  - BiometricEvaluation::IO::Logsheet, [427](#)
  - BiometricEvaluation::IO::SysLogsheet, [573](#)
- writeComment
  - BiometricEvaluation::IO::FileLogsheet, [324](#)
  - BiometricEvaluation::IO::Logsheet, [428](#)
  - BiometricEvaluation::IO::SysLogsheet, [573](#)
- writeDebug
  - BiometricEvaluation::IO::FileLogsheet, [324](#)
  - BiometricEvaluation::IO::Logsheet, [428](#)
  - BiometricEvaluation::IO::SysLogsheet, [574](#)
- writeFile
  - BiometricEvaluation::IO::Utility, [136](#), [137](#)
- writePipe
  - BiometricEvaluation::IO::Utility, [137](#), [138](#)
- writeToLogger
  - BiometricEvaluation::IO::SysLogsheet, [574](#)
- x
  - BiometricEvaluation::Image::Coordinate, [285](#)
- xDistance
  - BiometricEvaluation::Image::Coordinate, [285](#)
- xRes
  - BiometricEvaluation::Image::Resolution, [533](#)
- xSize
  - BiometricEvaluation::Image::Size, [549](#)
- y
  - BiometricEvaluation::Image::Coordinate, [285](#)
- yDistance
  - BiometricEvaluation::Image::Coordinate, [285](#)
- yRes
  - BiometricEvaluation::Image::Resolution, [533](#)
- ySize
  - BiometricEvaluation::Image::Size, [549](#)