Biometric Evaluation Common Framework

Wayne Salamon and Greg Fiumara

Contents

1	Introduction							
2								
3								
4	Erro	or Handling	7					
	4.1	Biometric Evaluation Exceptions	7					
	4.2	Signal Handling	7					
5	Inpu	nt/Output	11					
	5.1	Utility	11					
	5.2	Record Management	11					
	5.3	Logging	12					
6	Tim	e and Timing	15					
	6.1	Elapsed Time	15					
	6.2	Limiting Execution Time	16					
7	Ima	ge	17					
A	Tod	o List	21					
В	Nan	nespace Index	23					
	D 1	Manager Liet	22					

C	Clas	s Index			25
	C .1	Class I	Hierarchy		25
D	Clas	s Index			27
	D.1	Class I	List		27
E	Nan	nespace	Documen	tation	29
	E.1	Biome	tricEvalua	tion::Image Namespace Reference	29
		E.1.1	Detailed	Description	29
	E.2	Biome	tricEvalua	tion::Time Namespace Reference	29
		E.2.1	Detailed	Description	30
F	Clas	s Docui	nentation		31
	F.1	Biome	tricEvalua	tion::IO::ArchiveRecordStore Class Reference	31
		F.1.1	Detailed	Description	32
		F.1.2	Construc	tor & Destructor Documentation	33
			F.1.2.1	ArchiveRecordStore	33
			F.1.2.2	ArchiveRecordStore	33
			F.1.2.3	~ArchiveRecordStore	33
		F.1.3	Member	Function Documentation	34
			F.1.3.1	getSpaceUsed	34
			F.1.3.2	sync	34
			F.1.3.3	insert	34
			F.1.3.4	remove	35
			F.1.3.5	read	35
			F.1.3.6	replace	36
			F.1.3.7	length	36
			F.1.3.8	flush	36
			F.1.3.9	setCursor	37
			F.1.3.10	changeName	37
			F.1.3.11	vacuum	38
			F.1.3.12	getArchiveName	38

CONTENTS iii

		F.1.3.13 getManifestName	38
F.2	Biome	etricEvaluation::Utility::AutoArray< T > Class Template Refer-	39
F.3		etricEvaluation::Error::ConversionError Class Reference	39
1.3	F.3.1	Detailed Description	40
	F.3.2	Constructor & Destructor Documentation	40
	1.0.2	F.3.2.1 ConversionError	40
		F.3.2.2 ConversionError	40
F.4	Biome	etricEvaluation::IO::DBRecordStore Class Reference	41
	F.4.1	Detailed Description	42
	F.4.2	Constructor & Destructor Documentation	42
		F.4.2.1 DBRecordStore	42
		F.4.2.2 DBRecordStore	42
	F.4.3	Member Function Documentation	43
		F.4.3.1 getSpaceUsed	43
		F.4.3.2 sync	43
		F.4.3.3 insert	43
		F.4.3.4 remove	44
		F.4.3.5 read	44
		F.4.3.6 replace	45
		F.4.3.7 length	45
		F.4.3.8 flush	45
		F.4.3.9 setCursor	46
		F.4.3.10 changeName	46
F.5	Biome	etricEvaluation::Error::Exception Class Reference	47
	F.5.1	Detailed Description	48
	F.5.2	Constructor & Destructor Documentation	48
		F.5.2.1 Exception	48
		F.5.2.2 Exception	48
	F.5.3	Member Function Documentation	48
		F.5.3.1 getInfo	48

iv CONTENTS

F.6	Biome	etricEvaluation::IO::Factory Class Reference 49	9
	F.6.1	Detailed Description	9
	F.6.2	Member Function Documentation	9
		F.6.2.1 openRecordStore	9
F.7	Biome	etricEvaluation::Error::FileError Class Reference 50	0
	F.7.1	Detailed Description	0
	F.7.2	Constructor & Destructor Documentation	0
		F.7.2.1 FileError	0
		F.7.2.2 FileError	1
F.8	Biome	etricEvaluation::IO::FileRecordStore Class Reference 5	1
	F.8.1	Detailed Description	2
	F.8.2	Constructor & Destructor Documentation	2
		F.8.2.1 FileRecordStore	2
		F.8.2.2 FileRecordStore	3
	F.8.3	Member Function Documentation	3
		F.8.3.1 getSpaceUsed	3
		F.8.3.2 insert	4
		F.8.3.3 remove	4
		F.8.3.4 read	4
		F.8.3.5 replace	5
		F.8.3.6 length	5
		F.8.3.7 flush	6
		F.8.3.8 setCursor	6
		F.8.3.9 changeName	7
F.9	Biome	etricEvaluation::Image::Image Class Reference 5'	7
	F.9.1	Detailed Description	8
	F.9.2	Constructor & Destructor Documentation	8
		F.9.2.1 Image	8
	F.9.3	Member Function Documentation	9
		F.9.3.1 getXResolution	9
		F.9.3.2 getYResolution	9

CONTENTS

	F.9.3.3	getRawData	59
	F.9.3.4	getWidth	60
	F.9.3.5	getHeight	60
	F.9.3.6	getDepth	60
F.10 Biome	tricEvalua	tion::IO::LogCabinet Class Reference	60
F.10.1	Detailed	Description	61
F.10.2	Construc	tor & Destructor Documentation	62
	F.10.2.1	LogCabinet	62
	F.10.2.2	LogCabinet	62
F.10.3	Member	Function Documentation	63
	F.10.3.1	newLogSheet	63
	F.10.3.2	getName	63
	F.10.3.3	getDescription	63
	F.10.3.4	getCount	63
	F.10.3.5	remove	64
F.11 Biome	tricEvalua	tion::IO::LogSheet Class Reference	64
F.11.1	Detailed	Description	65
F.11.2	Construc	tor & Destructor Documentation	65
	F.11.2.1	LogSheet	65
F.11.3	Member	Function Documentation	66
	F.11.3.1	write	66
	F.11.3.2	newEntry	66
	F.11.3.3	getCurrentEntry	66
	F.11.3.4	resetCurrentEntry	66
	F.11.3.5	getCurrentEntryNumber	67
	F.11.3.6	sync	67
	F.11.3.7	setAutoSync	67
F.12 Biome	tricEvalua	tion::IO::ManifestEntry Struct Reference	67
F.13 Biome	tricEvalua	tion::Error::MemoryError Class Reference	68
F.13.1	Detailed	Description	68
F.13.2	Construc	tor & Destructor Documentation	68

vi CONTENTS

		F.13.2.1	MemoryError	68
		F.13.2.2	MemoryError	69
F.14 B	iomet	ricEvaluat	ion::Error::ObjectDoesNotExist Class Reference	69
F	.14.1	Detailed 1	Description	69
F	.14.2	Construct	tor & Destructor Documentation	70
		F.14.2.1	ObjectDoesNotExist	70
		F.14.2.2	ObjectDoesNotExist	70
F.15 B	iomet	ricEvaluat	ion::Error::ObjectExists Class Reference	70
F	.15.1	Detailed 1	Description	71
F	.15.2	Construct	tor & Destructor Documentation	71
		F.15.2.1	ObjectExists	71
		F.15.2.2	ObjectExists	71
F.16 B	iomet	ricEvaluat	ion::Error::ObjectIsClosed Class Reference	71
F	.16.1	Detailed 1	Description	72
F	.16.2	Construct	tor & Destructor Documentation	72
		F.16.2.1	ObjectIsClosed	72
		F.16.2.2	ObjectIsClosed	72
F.17 B	iomet	ricEvaluat	ion::Error::ObjectIsOpen Class Reference	72
F	.17.1	Detailed 1	Description	73
F	.17.2	Construct	tor & Destructor Documentation	73
		F.17.2.1	ObjectIsOpen	73
		F.17.2.2	ObjectIsOpen	73
F.18 B	iomet	ricEvaluat	ion::Error::ParameterError Class Reference	74
F	.18.1	Detailed 1	Description	74
F	.18.2	Construct	tor & Destructor Documentation	74
		F.18.2.1	ParameterError	74
		F.18.2.2	ParameterError	74
F.19 B	Siomet	ricEvaluat	ion::IO::Properties Class Reference	75
F	.19.1	Detailed 1	Description	75
F	.19.2	Construct	tor & Destructor Documentation	76
		F.19.2.1	Properties	76

CONTENTE	•
CONTENTS	VI
COMILIMO	V 11

	F.19.3	Member	Function Documentation	76
		F.19.3.1	setProperty	76
		F.19.3.2	setPropertyFromInteger	77
		F.19.3.3	removeProperty	77
		F.19.3.4	getProperty	78
		F.19.3.5	getPropertyAsInteger	78
		F.19.3.6	sync	78
		F.19.3.7	changeName	79
F.20	Biomet	tricEvalua	tion::Image::RawImage Class Reference	79
	F.20.1	Construc	tor & Destructor Documentation	80
		F.20.1.1	RawImage	80
	F.20.2	Member	Function Documentation	80
		F.20.2.1	getWidth	80
		F.20.2.2	getHeight	80
		F.20.2.3	getDepth	81
		F.20.2.4	getXResolution	81
		F.20.2.5	getYResolution	81
		F.20.2.6	getRawData	81
F.21	Biomet	tricEvalua	tion::IO::RecordStore Class Reference	82
	F.21.1	Detailed	Description	84
	F.21.2	Construc	tor & Destructor Documentation	84
		F.21.2.1	RecordStore	84
		F.21.2.2	RecordStore	85
	F.21.3	Member	Function Documentation	85
		F.21.3.1	getName	85
		F.21.3.2	getDescription	85
		F.21.3.3	getCount	85
		F.21.3.4	changeName	86
		F.21.3.5	changeDescription	86
		F.21.3.6	getSpaceUsed	86
		F.21.3.7	sync	87

viii CONTENTS

		F.21.3.8 insert	37
		F.21.3.9 remove	37
		F.21.3.10 read	88
		F.21.3.11 replace	88
		F.21.3.12 length	39
		F.21.3.13 flush	39
		F.21.3.14 setCursor	0
		F.21.3.15 removeRecordStore	90
F.2	21.4	Member Data Documentation	1
		F.21.4.1 CONTROLFILENAME	1
		F.21.4.2 NAMEPROPERTY)1
		F.21.4.3 BERKELEYDBTYPE	1
		F.21.4.4 BE_RECSTORE_SEQ_START	91
F.22 Bio	omet	ricEvaluation::Error::SignalManager Class Reference 9)2
F.2	22.1	Detailed Description	92
F.2	22.2	Constructor & Destructor Documentation	93
		F.22.2.1 SignalManager	93
F.2	22.3	Member Function Documentation	93
		F.22.3.1 setSignalSet	93
		F.22.3.2 clearSignalSet	94
		F.22.3.3 setDefaultSignalSet	94
		F.22.3.4 sigHandled	94
		F.22.3.5 start	94
		F.22.3.6 stop	95
		F.22.3.7 setSigHandled	95
		F.22.3.8 clearSigHandled	95
F.2	22.4	Member Data Documentation	95
		F.22.4.1 _canSigJump	95
		F.22.4.2 _sigJumpBuf	95
F.23 Bio	omet	ricEvaluation::Error::StrategyError Class Reference 9	96
F.2	23.1	Detailed Description	96

CONTRENIEC	•
CONTENTS	13
COMILIMO	1.7

F.23.2	Constructor & Destructor Documentation	96
	F.23.2.1 StrategyError	96
	F.23.2.2 StrategyError	96
F.24 Biome	etricEvaluation::Time::Timer Class Reference	97
F.24.1	Detailed Description	97
F.24.2	Constructor & Destructor Documentation	97
	F.24.2.1 Timer	97
F.24.3	Member Function Documentation	97
	F.24.3.1 start	97
	F.24.3.2 stop	98
	F.24.3.3 elapsed	98
F.25 Biome	etricEvaluation::Error::Utility Class Reference	98
F.25.1	Detailed Description	99
F.25.2	Member Function Documentation	99
	F.25.2.1 errorStr	99
F.26 Biome	etricEvaluation::IO::Utility Class Reference	99
F.26.1	Detailed Description	100
F.26.2	Member Function Documentation	100
	F.26.2.1 removeDirectory	100
	F.26.2.2 getFileSize	100
	F.26.2.3 fileExists	101
	F.26.2.4 validateRootName	101
	F.26.2.5 constructAndCheckPath	101
F.27 Biome	etricEvaluation::Time::Watchdog Class Reference	102
F.27.1	Detailed Description	103
F.27.2	Constructor & Destructor Documentation	103
	F.27.2.1 Watchdog	103
F.27.3	Member Function Documentation	104
	F.27.3.1 setInterval	104
	F.27.3.2 start	104
	F.27.3.3 stop	104

X CONTENTS

	F.27.3.4	expired	105
	F.27.3.5	setCanSigJump	105
	F.27.3.6	clearCanSigJump	105
	F.27.3.7	setExpired	105
	F.27.3.8	clearExpired	105
F.27.4	Member	Data Documentation	105
	F.27.4.1	PROCESSTIME	105
	F.27.4.2	REALTIME	105

Introduction

This document describes the framework and application programming interfaces (API) used to support the evaluation of biometric software within the Image Group at NIST. 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 that vendors implement in there software, which is delivered to NIST as a software library. A common test driver is used to call the vendor library to perform the biometric operation. In order to support the common functionality used across all evaluations, such as logging, file input/output, etc., a common framework is used.

2 Introduction

Overview

The Biometric Evaluation Framework (BECommon) is a set of C++[1] 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.

BECommon 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*. All classes within BECommon belong to the top-level *BiometricEvaluation* name space.

4 Overview

Utility Classes

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.

4.1 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 informational string stored within that object provides more information on the error.

Listing 5.1 shows an example of exception handling when using the logging classes described in Section 5.3.

4.2 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 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

8 Error Handling

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 watchdog block.

The example in Listing 4.2 shows application use of the *SignalManager* class.

Listing 4.1: Using the SignalManger

```
#include <be_error_signal_manager.h>
    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.

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

Listing 4.2: Using the SignalManger

```
#include <be_error_signal_manager.h>
2
   using namespace BiometricEvaluation;
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);
            sigmgr->setSignalSet(sigset);
11
12
13
            BEGIN_SIGNAL_BLOCK(sigmgr, sigblock2);
            // code that may result in signal generation
14
            END_SIGNAL_BLOCK(asigmgr, sigblock2);
15
            if (sigmgr->sigHandled()) {
16
                    cout << "SIGUSR1_occurred." << endl;
17
18
19
   }
```

Input/Output

The *BiometricEvaluation::IO*) classes are 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.

5.1 Utility

The *IO::Utility* class provides static methods that are used to manipulate the file system and other low-level mechanisms. These methods can be used by applications in addition to being used by other classes within the Biometric Evaluation framework.

5.2 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 *Record-Store* 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 in not efficient, and leads to longer

12 Input/Output

run times as well as difficulties 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.

Record stores 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 pointer and data length, and is associated with a string, the key. Keys must be unique and are associated with a single record. Attempts to insert multiple records with the same key result in an exception being thrown.

5.3 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 output details. There are two classes, *IO::LogCabinet* and *IO::LogSheet* that are used to perform consistent logging of information by applications. A *LogCabinet* contains a set of *LogSheet*s.

A *LogSheet* is an output stream (subclass of *std::ostringstream*), and therefore can handle built-in types and any class that supports streaming. The example code in 5.1 shows how an application can use a *LogSheet*, contained within a *LogCabinet*, to record operational information.

Log sheets are simple text files, with each entry numbered by the *LogSheet* class when written to the file. The description of the sheet is placed at the top of the file during construction of the *LogSheet* object. A call to the *newEntry()* method commits the current entry to the log file, 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.

5.3 Logging 13

The example in Listing 5.1 shows application use of the logging facility.

Listing 5.1: Using a LogSheet within a LogCabinet

```
1
   #include <be_io_logcabinet.h>
   using namespace BiometricEvaluation;
3
   using namespace BiometricEvaluation::IO;
5
   LogCabinet *1c;
6
   try {
        lc = new LogCabinet(lcname, "A, Log, Cabinet", "");
7
   } catch (Error::ObjectExists &e) {
8
9
        cout << "The_Log_Cabinet_already_exists." << endl;</pre>
10
        return (-1);
11
   } catch (Error::StrategyError& e) {
12
        cout << "Caught_{\bot}" << e.getInfo() << endl;
        return (-1);
13
14
   }
   auto_ptr <LogCabinet> alc(lc);
15
16
   try {
        1s = alc ->newLogSheet(lsname, "Log_Sheet_in_Cabinet");
17
18
   } catch (Error:: ObjectExists &e) {
19
        cout << "The_Log_Sheet_already_exists." << endl;</pre>
20
        return (-1);
21
   } catch (Error::StrategyError& e) {
22
        cout << "Caught_" << e.getInfo() << endl;
23
        return (-1);
24
25
   1s->setAutoSync(true); // Force write of every entry when finished
26
   int i = ...
   *ls << "Adding_an_integer_value_" << i << "_to_the_log." << endl;
2.7
                            // Forces the write of the current entry
28
   ls -> newEntry();
29
30
   delete 1s;
31
   return;
                             // The LogCabinet is destructed by the auto_ptr
```

14 Input/Output

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.

6.1 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 6.1 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 6.1: Using the Timer

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

6.2 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.

Watch dog 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 4.2 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 6.2 shows how an application can use a *Watchdog* object to limit the about of process time for a block of code.

Listing 6.2: Using the Watchdog

```
#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
             BEGIN\_WATCHDOG\_BLOCK(theDog, watchdogblock1);\\
6
7
                      // Do something that may take more than 300 usecs
8
             END\_WATCHDOG\_BLOCK(\,the\,Dog\,,\ watchdog\,block\,1\,)\,;\\
9
             if (theDog->expired()) {
10
                      cout << "That_took_too_long." << endl;</pre>
11
                      // further processing
12
             }
13
14
```

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 watch dog 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.

Image

18 Image

Bibliography

[1] Bjarne Stroustrup. The C++ Programming Language. Addison Wesley, special edition, 2000. 3

Appendix A

Todo List

Namespace BiometricEvaluation::Image Add more detail.

Class BiometricEvaluation::Image::Image Add more info on the image data, and what coversions are possible.

22 Todo List

Appendix B

Namespace Index

B.1 Namespace List

Here is a list of all documented namespaces with brief descriptions:	
BiometricEvaluation::Image (A class representing a raw image)	2
Riometric Evaluation: Time	20

Appendix C

Class Index

C.1 Class Hierarchy

his inheritance list is sorted roughly, but not completely, alphabetically:	
BiometricEvaluation::Utility::AutoArray< T >	39
BiometricEvaluation::Error::Exception	47
BiometricEvaluation::Error::ConversionError	39
BiometricEvaluation::Error::FileError	50
BiometricEvaluation::Error::MemoryError	68
BiometricEvaluation::Error::ObjectDoesNotExist	69
BiometricEvaluation::Error::ObjectExists	70
BiometricEvaluation::Error::ObjectIsClosed	71
BiometricEvaluation::Error::ObjectIsOpen	72
BiometricEvaluation::Error::ParameterError	74
BiometricEvaluation::Error::StrategyError	96
BiometricEvaluation::IO::Factory	49
BiometricEvaluation::Image::Image	
BiometricEvaluation::Image::RawImage	79
BiometricEvaluation::IO::LogCabinet	60
BiometricEvaluation::IO::LogSheet	64
BiometricEvaluation::IO::ManifestEntry	67
BiometricEvaluation::IO::Properties	75
BiometricEvaluation::IO::RecordStore	82
BiometricEvaluation::IO::ArchiveRecordStore	31
BiometricEvaluation::IO::DBRecordStore	41
BiometricEvaluation::IO::FileRecordStore	51
BiometricEvaluation::Error::SignalManager	
BiometricEvaluation::Time::Timer	97

6	Class Index

BiometricEvaluation::Error::Utility											98
BiometricEvaluation::IO::Utility											99
BiometricEvaluation::Time::Watchdog											102

 $Generated \ on \ Fri \ Feb \ 4\ 2011\ 15:07:11 \ for \ Biometric \ Evaluation \ Common \ Framework \ by \ Doxygen$

Appendix D

Class Index

D.1 Class List

Here a	are the classes, structs, unions and interfaces	with	briei de	scriptions:	
Bi	ometricEvaluation::IO::ArchiveRecordStore	(Thi	is class	implements	the
	IO::RecordStore interface by storing	data	items in	single file,	with

10Recordstore interface by storing data items in single me, with	
an associated manifest file)	3
BiometricEvaluation::Utility::AutoArray< T >	39
BiometricEvaluation::Error::ConversionError (Error when converting one	
object into another, a property value from string to int, for example)	39
BiometricEvaluation::IO::DBRecordStore (A class that implements	
IO::RecordStore using a Berkeley DB database as the underlying	
record storage system)	4
BiometricEvaluation::Error::Exception (The parent class of all BiometricE-	
valuation exceptions)	4
BiometricEvaluation::IO::Factory	49
BiometricEvaluation::Error::FileError (File error when opening, reading,	
writing, etc)	5
BiometricEvaluation::IO::FileRecordStore	5
BiometricEvaluation::Image::Image (A abstract class to represent images	
and their attributes)	5'
BiometricEvaluation::IO::LogCabinet	
BiometricEvaluation::IO::LogSheet (A class to represent a single logging	
mechanism)	6
BiometricEvaluation::IO::ManifestEntry	6
BiometricEvaluation::Error::MemoryError (An error occurred when allocat-	
ing an object)	6
BiometricEvaluation::Error::ObjectDoesNotExist (The named object does	
mat arrive)	6

28 Class Index

BiometricEvaluation::Error::ObjectExists (The named object exists and will	
not be replaced)	70
BiometricEvaluation::Error::ObjectIsClosed (The object is closed)	71
BiometricEvaluation::Error::ObjectIsOpen (The object is already opened)	72
BiometricEvaluation::Error::ParameterError (An invalid parameter was	
passed to a constructor or method)	74
BiometricEvaluation::IO::Properties (A Properties class is used to maintain	
key/value pairs of strings, with each property matched to one value)	75
BiometricEvaluation::Image::RawImage	79
BiometricEvaluation::IO::RecordStore (A class to represent a data storage	
mechanism)	82
BiometricEvaluation::Error::SignalManager (A SignalManager object is	
used to handle signals that come from the operating system)	92
BiometricEvaluation::Error::StrategyError (A StrategyError object is thrown	
when the underlying implementation of this interface encounters an	
error)	96
BiometricEvaluation::Time::Timer (This class can be used by applications to	
report the amount of time a block of code takes to execute)	97
BiometricEvaluation::Error::Utility (This class contains methods that are	
useful utility functions, such as converting system values to strings)	98
BiometricEvaluation::IO::Utility	99
BiometricEvaluation::Time::Watchdog (A Watchdog object can be used by	
applications to limit the amount of processing time taken by a block	100
of code)	102

Appendix E

Namespace Documentation

E.1 BiometricEvaluation::Image Namespace Reference

A class representing a raw image.

Classes

- class Image
 - A abstract class to represent images and their attributes.
- class RawImage

E.1.1 Detailed Description

A class representing a raw image.

Todo

Add more detail.

E.2 BiometricEvaluation::Time Namespace Reference

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 object can be used by applications to limit the amount of processing time taken by a block of code.

Functions

• 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

E.2.1 Detailed Description

The Time name space gathers all timing relating matters, such as Timers, Watchdog timers, etc. Time values are in microsecond units.

Appendix F

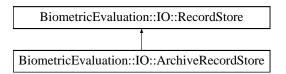
Class Documentation

F.1 BiometricEvaluation::IO::ArchiveRecordStore Class Reference

This class implements the IO::RecordStore 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 string &name, const string &description, const string &parentDir) throw (Error::ObjectExists, Error::StrategyError)
- ArchiveRecordStore (const string &name, const string &parentDir, uint8_t mode=IO::READWRITE) throw (Error::ObjectDoesNotExist, Error::StrategyError)
- ~ArchiveRecordStore ()
- uint64_t getSpaceUsed () throw (Error::StrategyError)
- void sync () throw (Error::StrategyError)
- void insert (const string &key, const void *const data, const uint64_t size) throw (Error::ObjectExists, Error::StrategyError)

- void remove (const string &key) throw (Error::ObjectDoesNotExist, Error::StrategyError)
- uint64_t read (const string &key, void *const data) throw (Error::ObjectDoesNotExist, Error::StrategyError)
- void replace (const string &key, const void *const data, const uint64_t size) throw (Error::ObjectDoesNotExist, Error::StrategyError)
- uint64_t length (const string &key) throw (Error::ObjectDoesNotExist)
- void flush (const string &key) throw (Error::ObjectDoesNotExist, Error::StrategyError)
- uint64_t sequence (string &key, void *const data, int cursor=BE_RECSTORE_-SEQ_NEXT) throw (Error::ObjectDoesNotExist, Error::StrategyError)
- void setCursor (string &key) throw (Error::ObjectDoesNotExist, Error::StrategyError)
- void changeName (const string &name) throw (Error::ObjectExists, Error::StrategyError)
- string getArchiveName ()
- string getManifestName ()

Static Public Member Functions

static void vacuum (const string &name, const string &parentDir) throw (Error::ObjectDoesNotExist, Error::StrategyError)

F.1.1 Detailed Description

This class implements the IO::RecordStore 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.

F.1.2 Constructor & Destructor Documentation

F.1.2.1 BiometricEvaluation::IO::ArchiveRecordStore::ArchiveRecordStore (const string & name, const string & description, const string & parentDir) throw (Error::ObjectExists, Error::StrategyError)

Create a new ArchiveRecordStore, read/write mode.

Parameters

```
name[in] The name of the store.description[in] The store's description.parentDir[in] The directory where the store is to be created.
```

Exceptions

```
Error::ObjectExists The store already exists.
```

Error::StrategyError An error occurred when accessing the underlying file system.

F.1.2.2 BiometricEvaluation::IO::ArchiveRecordStore::ArchiveRecordStore (const string & name, const string & parentDir, uint8_t mode = IO::READWRITE) throw (Error::ObjectDoesNotExist, Error::StrategyError)

Open an existing ArchiveRecordStore.

Parameters

```
name[in] The name of the store.parentDir[in] The directory where the store is to be created.mode[in] Open mode, read-only or read-write.
```

Exceptions

```
Error::ObjectDoesNotExist The store does not exist.
```

Error::StrategyError An error occurred when accessing the underlying file system.

F.1.2.3 BiometricEvaluation::IO::ArchiveRecordStore::~ArchiveRecordStore

Destructor.

F.1.3 Member Function Documentation

F.1.3.1 uint64_t BiometricEvaluation::IO::ArchiveRecordStore::getSpaceUsed () throw (Error::StrategyError) [virtual]

Obtain the amount of 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.

Exceptions

Error::StrategyError An error occurred when using the underlying storage system.

Reimplemented from BiometricEvaluation::IO::RecordStore.

F.1.3.2 void BiometricEvaluation::IO::ArchiveRecordStore::sync () throw (Error::StrategyError) [virtual]

Synchronize the entire record store to persistent storage.

Exceptions

Error::StrategyError An error occurred when using the underlying storage system.

Reimplemented from BiometricEvaluation::IO::RecordStore.

F.1.3.3 void BiometricEvaluation::IO::ArchiveRecordStore::insert (const string & key, const void *const data, const uint64_t size) throw (Error::ObjectExists, Error::StrategyError) [virtual]

Insert a record into the store.

Parameters

```
key[in] The key of the record to be flushed.data[in] The data for the record.size[in] The size, in bytes, of the record.
```

Exceptions

Error:: ObjectExists A record with the given key is already present.

Error::StrategyError An error occurred when using the underlying storage system.

Implements BiometricEvaluation::IO::RecordStore.

F.1.3.4 void BiometricEvaluation::IO::ArchiveRecordStore::remove (const string & key) throw (Error::ObjectDoesNotExist, Error::StrategyError) [virtual]

Remove a record from the store.

Parameters

key[in] The key of the record to be removed.

Exceptions

Error::ObjectDoesNotExist A record for the key does not exist.

Error::StrategyError An error occurred when using the underlying storage system.

Implements BiometricEvaluation::IO::RecordStore.

F.1.3.5 uint64_t BiometricEvaluation::IO::ArchiveRecordStore::read (const string & key, void *const data) throw (Error::ObjectDoesNotExist, Error::StrategyError) [virtual]

Read a complete record from a store. Applications are responsible for allocating storage for the record's data.

Parameters

key[in] The key of the record to be read. [in] Pointer to where the data is to be written.

Returns

The size of the record.

Exceptions

Error::ObjectDoesNotExist A record for the key does not exist.

Error::StrategyError An error occurred when using the underlying storage system.

Implements BiometricEvaluation::IO::RecordStore.

F.1.3.6 void BiometricEvaluation::IO::ArchiveRecordStore::replace (const string & key, const void *const data, const uint64_t size) throw (Error::ObjectDoesNotExist, Error::StrategyError) [virtual]

Replace a complete record in a store.

Parameters

key[in] The key of the record to be replaced.data[in] The data for the record.

Exceptions

Error::ObjectDoesNotExist A record for the key does not exist.

Error::StrategyError An error occurred when using the underlying storage system.

Implements BiometricEvaluation::IO::RecordStore.

F.1.3.7 uint64_t BiometricEvaluation::IO::ArchiveRecordStore::length (const string & key) throw (Error::ObjectDoesNotExist) [virtual]

Return the length of a record.

Parameters

key[in] The key of the record.

Returns

The record length.

Exceptions

Error::ObjectDoesNotExist A record for the key does not exist.

Error::StrategyError An error occurred when using the underlying storage system.

Implements BiometricEvaluation::IO::RecordStore.

F.1.3.8 void BiometricEvaluation::IO::ArchiveRecordStore::flush (const string & key) throw (Error::ObjectDoesNotExist, Error::StrategyError) [virtual]

Commit the record's data to storage.

Parameters

key[in] The key of the record to be flushed.

Exceptions

Error::ObjectDoesNotExist A record for the key does not exist.

Error::StrategyError An error occurred when using the underlying storage system.

Implements BiometricEvaluation::IO::RecordStore.

F.1.3.9 void BiometricEvaluation::IO::ArchiveRecordStore::setCursor(string & key) throw (Error::ObjectDoesNotExist, Error::StrategyError) [virtual]

Set the sequence cursor to an arbitrary position within the RecordStore, starting at key. Key will be the first record returned from the next call to sequence().

Parameters

key[in] The key of the record which will be returned by the first subsequent call to sequence().

Exceptions

Error::ObjectDoesNotExist A record for the key does not exist.

Error::StrategyError An error occurred when using the underlying storage system.

Implements BiometricEvaluation::IO::RecordStore.

F.1.3.10 void BiometricEvaluation::IO::ArchiveRecordStore::changeName (const string & name) throw (Error::ObjectExists, Error::StrategyError) [virtual]

Change the name of the RecordStore.

Parameters

name[in] The new name for the RecordStore.

Exceptions

Error::StrategyError An error occurred when using the underlying storage system, or the name is malformed.

Reimplemented from BiometricEvaluation::IO::RecordStore.

F.1.3.11 static void BiometricEvaluation::IO::ArchiveRecordStore::vacuum (const string & name, const string & parentDir) throw (Error::ObjectDoesNotExist, Error::StrategyError) [static]

Remove deleted entries from the manifest and archive files to save space on disk.

Parameters

```
name[in] The name of the existing RecordStore.
parentDir[in] Where, in the file system, the store is rooted.
```

Exceptions

```
Error::ObjectDoesNotExist A record with the given key does not exist.
Error::StrategyError An error occurred when using the underlying storage system.
```

Note

This is an expensive operation.

F.1.3.12 string BiometricEvaluation::IO::ArchiveRecordStore::getArchiveName (

Obtain the name of the file storing the data for this store.

Returns

Path to archive file.

F.1.3.13 string BiometricEvaluation::IO::ArchiveRecordStore::getManifestName (

Obtain the name of the file storing the manifest data data for this store.

Returns

Path to manifest file.

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

• be_io_archiverecstore.h

F.2 BiometricEvaluation::Utility::AutoArray< T > Class Template Reference

Public Types

- typedef T value_type
- typedef T * iterator
- typedef const T * const_iterator
- typedef T & reference
- typedef const T & const_reference

Public Member Functions

- operator T * ()
- reference **operator**[] (ptrdiff_t i)
- const_reference **operator**[] (ptrdiff_t i) const
- AutoArray & operator= (const AutoArray &other)
- iterator begin ()
- const_iterator begin () const
- iterator end ()
- const_iterator end () const
- size_t size () const
- AutoArray (size_t size)
- AutoArray (const AutoArray ©)

$template < class \ T > class \ Biometric Evaluation :: Utility :: AutoArray < T >$

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

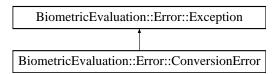
• be_utility_autoarray.h

F.3 BiometricEvaluation::Error::ConversionError Class Reference

Error 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 (string info)

F.3.1 Detailed Description

Error when converting one object into another, a property value from string to int, for example.

F.3.2 Constructor & Destructor Documentation

F.3.2.1 BiometricEvaluation::Error::ConversionError::ConversionError()

Construct a ConversionError object with the default information string.

Returns

The ConversionError object.

F.3.2.2 BiometricEvaluation::Error::ConversionError::ConversionError (string *info*)

Construct a ConversionError object with an information string appended to the default information string.

Returns

The ConversionError object.

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

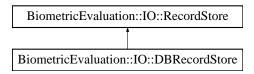
• be_error_exception.h

F.4 BiometricEvaluation::IO::DBRecordStore Class Reference

A class that implements IO::RecordStore 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 string &name, const string &description, const string &parentDir) throw (Error::ObjectExists, Error::StrategyError)
- DBRecordStore (const string &name, const string &parentDir, uint8_t mode=IO::READWRITE) throw (Error::ObjectDoesNotExist, Error::StrategyError)
- uint64_t getSpaceUsed () throw (Error::StrategyError)
- void sync () throw (Error::StrategyError)
- void insert (const string &key, const void *const data, const uint64_t size) throw (Error::ObjectExists, Error::StrategyError)
- void remove (const string &key) throw (Error::ObjectDoesNotExist, Error::StrategyError)
- uint64_t read (const string &key, void *const data) throw (Error::ObjectDoesNotExist, Error::StrategyError)
- virtual void replace (const string &key, const void *const data, const uint64_t size) throw (Error::ObjectDoesNotExist, Error::StrategyError)
- virtual uint64_t length (const string &key) throw (Error::ObjectDoesNotExist, Error::StrategyError)
- void flush (const string &key) throw (Error::ObjectDoesNotExist, Error::StrategyError)
- uint64_t sequence (string &key, void *const data, int cursor=BE_RECSTORE_-SEQ_NEXT) throw (Error::ObjectDoesNotExist, Error::StrategyError)
- void setCursor (string &key) throw (Error::ObjectDoesNotExist, Error::StrategyError)
- void changeName (const string &name) throw (Error::ObjectExists, Error::StrategyError)

F.4.1 Detailed Description

A class that implements IO::RecordStore using a Berkeley DB database as the underlying record storage system.

E.4.2 Constructor & Destructor Documentation

F.4.2.1 BiometricEvaluation::IO::DBRecordStore::DBRecordStore (const string & name, const string & description, const string & parentDir) throw (Error::ObjectExists, Error::StrategyError)

Create a new DBRecordStore, read/write mode.

Parameters

```
name[in] The name of the store.
description[in] The store's description.
parentDir[in] The directory where the store is to be created.
```

Exceptions

```
Error::ObjectExists The store already exists.
```

Error::StrategyError An error occurred when accessing the underlying file system.

F.4.2.2 BiometricEvaluation::IO::DBRecordStore::DBRecordStore (const string & name, const string & parentDir, uint8_t mode = IO::READWRITE) throw (Error::ObjectDoesNotExist, Error::StrategyError)

Open an existing DBRecordStore.

Parameters

```
name[in] The name of the store.parentDir[in] The directory where the store is to be created.mode[in] Open mode, read-only or read-write.
```

Exceptions

```
Error::ObjectDoesNotExist The store does not exist.
```

Error::StrategyError An error occurred when accessing the underlying file system.

F.4.3 Member Function Documentation

F.4.3.1 uint64_t BiometricEvaluation::IO::DBRecordStore::getSpaceUsed () throw (Error::StrategyError) [virtual]

Obtain the amount of 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.

Exceptions

Error::StrategyError An error occurred when using the underlying storage system.

Reimplemented from BiometricEvaluation::IO::RecordStore.

F.4.3.2 void BiometricEvaluation::IO::DBRecordStore::sync () throw (Error::StrategyError) [virtual]

Synchronize the entire record store to persistent storage.

Exceptions

Error::StrategyError An error occurred when using the underlying storage system.

Reimplemented from BiometricEvaluation::IO::RecordStore.

F.4.3.3 void BiometricEvaluation::IO::DBRecordStore::insert (const string & key, const void *const data, const uint64_t size) throw (Error::ObjectExists, Error::StrategyError) [virtual]

Insert a record into the store.

Parameters

```
key[in] The key of the record to be flushed.data[in] The data for the record.size[in] The size, in bytes, of the record.
```

Exceptions

Error::ObjectExists A record with the given key is already present.

Error::StrategyError An error occurred when using the underlying storage system.

Implements BiometricEvaluation::IO::RecordStore.

F.4.3.4 void BiometricEvaluation::IO::DBRecordStore::remove (const string & key) throw (Error::ObjectDoesNotExist, Error::StrategyError) [virtual]

Remove a record from the store.

Parameters

key[in] The key of the record to be removed.

Exceptions

Error::ObjectDoesNotExist A record for the key does not exist.

Error::StrategyError An error occurred when using the underlying storage system.

Implements BiometricEvaluation::IO::RecordStore.

F.4.3.5 uint64_t BiometricEvaluation::IO::DBRecordStore::read (const string & key, void *const data) throw (Error::ObjectDoesNotExist, Error::StrategyError) [virtual]

Read a complete record from a store. Applications are responsible for allocating storage for the record's data.

Parameters

key[in] The key of the record to be read. [in] Pointer to where the data is to be written.

Returns

The size of the record.

Exceptions

Error::ObjectDoesNotExist A record for the key does not exist.

Error::StrategyError An error occurred when using the underlying storage system.

Implements BiometricEvaluation::IO::RecordStore.

F.4.3.6 virtual void BiometricEvaluation::IO::DBRecordStore::replace (const string & key, const void *const data, const uint64_t size) throw (Error::ObjectDoesNotExist, Error::StrategyError) [virtual]

Replace a complete record in a store.

Parameters

key[in] The key of the record to be replaced.data[in] The data for the record.

Exceptions

Error::ObjectDoesNotExist A record for the key does not exist.

Error::StrategyError An error occurred when using the underlying storage system.

Implements BiometricEvaluation::IO::RecordStore.

F.4.3.7 virtual uint64_t BiometricEvaluation::IO::DBRecordStore::length (const string & key) throw (Error::ObjectDoesNotExist, Error::StrategyError) [virtual]

Return the length of a record.

Parameters

key[in] The key of the record.

Returns

The record length.

Exceptions

Error::ObjectDoesNotExist A record for the key does not exist.

Error::StrategyError An error occurred when using the underlying storage system.

Implements BiometricEvaluation::IO::RecordStore.

F.4.3.8 void BiometricEvaluation::IO::DBRecordStore::flush (const string & key) throw (Error::ObjectDoesNotExist, Error::StrategyError) [virtual]

Commit the record's data to storage.

Parameters

key[in] The key of the record to be flushed.

Exceptions

Error::ObjectDoesNotExist A record for the key does not exist.

Error::StrategyError An error occurred when using the underlying storage system.

Implements BiometricEvaluation::IO::RecordStore.

F.4.3.9 void BiometricEvaluation::IO::DBRecordStore::setCursor (string & key) throw (Error::ObjectDoesNotExist, Error::StrategyError) [virtual]

Set the sequence cursor to an arbitrary position within the RecordStore, starting at key. Key will be the first record returned from the next call to sequence().

Parameters

key[in] The key of the record which will be returned by the first subsequent call to sequence().

Exceptions

Error::ObjectDoesNotExist A record for the key does not exist.

Error::StrategyError An error occurred when using the underlying storage system.

Implements BiometricEvaluation::IO::RecordStore.

F.4.3.10 void BiometricEvaluation::IO::DBRecordStore::changeName (const string & name) throw (Error::ObjectExists, Error::StrategyError) [virtual]

Change the name of the RecordStore.

Parameters

name[in] The new name for the RecordStore.

Exceptions

Error::StrategyError An error occurred when using the underlying storage system, or the name is malformed.

 $Reimplemented\ from\ Biometric Evaluation:: IO:: Record Store.$

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

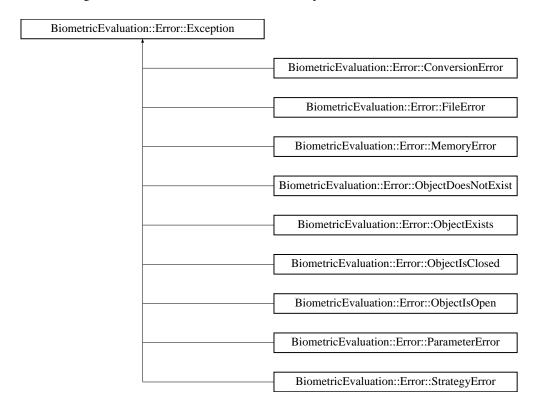
· be_io_dbrecstore.h

F.5 BiometricEvaluation::Error::Exception Class Reference

The parent class of all BiometricEvaluation exceptions.

#include <be_error_exception.h>

Inheritance diagram for BiometricEvaluation::Error::Exception:



Public Member Functions

- Exception ()
- Exception (string info)

• string getInfo ()

F.5.1 Detailed Description

The parent class of all BiometricEvaluation 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.

F.5.2 Constructor & Destructor Documentation

F.5.2.1 BiometricEvaluation::Error::Exception::Exception ()

Construct an Exception object without an information string.

Returns

The Exception object.

F.5.2.2 BiometricEvaluation::Error::Exception::Exception (string info)

Construct an Exception object with an information string.

Parameters

info[in] The information string associated with the exception.

Returns

The Exception object.

F.5.3 Member Function Documentation

F.5.3.1 string BiometricEvaluation::Error::Exception::getInfo()

Obtain the information string associated with the exception.

Returns

The information string.

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

• be_error_exception.h

F.6 BiometricEvaluation::IO::Factory Class Reference

#include <be_io_factory.h>

Static Public Member Functions

static tr1::shared_ptr< RecordStore > openRecordStore (const string &name, const string &parentDir, uint8_t mode=READWRITE) throw (Error::ObjectDoesNotExist, Error::StrategyError)

Open an existing RecordStore and return a managed pointer to the the object representing that store.

F.6.1 Detailed Description

A class to provide constructed objects of classes defined in the BiometricEvaluation::IO package, RecordStores, etc.

F.6.2 Member Function Documentation

F.6.2.1 static tr1::shared_ptr<RecordStore> BiometricEvaluation::IO::Factory::openRecordStore (const string & name, const string & parentDir, uint8_t mode = READWRITE) throw (Error::ObjectDoesNotExist, Error::StrategyError) [static]

Open an existing RecordStore 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 it is.

The allocated object will be automatically freed when the returned pointer goes out of scope. Applications should not delete the object.

Parameters

name[in] The name of the store to be opened.parentDir[in] Where, in the file system, the store is rooted.mode[in] The type of access a client of this RecordStore has.

Returns

An object representing the existing store.

Exceptions

Error::ObjectDoesNotExist The RecordStore does not exist.

Error::StrategyError An error occurred when using the underlying storage system, or the name is malformed.

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

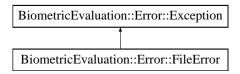
• be_io_factory.h

F.7 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 (string info)

F.7.1 Detailed Description

File error when opening, reading, writing, etc.

F.7.2 Constructor & Destructor Documentation

F.7.2.1 BiometricEvaluation::Error::FileError()

Construct a FileError object with the default information string.

Returns

The FileError object.

F.7.2.2 BiometricEvaluation::Error::FileError::FileError (string info)

Construct a FileError object with an information string appended to the default information string.

Returns

The FileError object.

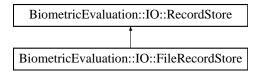
The documentation for this class was generated from the following file:

• be_error_exception.h

F.8 BiometricEvaluation::IO::FileRecordStore Class Reference

#include <be_io_filerecstore.h>

Inheritance diagram for BiometricEvaluation::IO::FileRecordStore:



Public Member Functions

- FileRecordStore (const string &name, const string &description, const string &parentDir) throw (Error::ObjectExists, Error::StrategyError)
- FileRecordStore (const string &name, const string &parentDir, uint8_t mode=IO::READWRITE) throw (Error::ObjectDoesNotExist, Error::StrategyError)
- uint64_t getSpaceUsed () throw (Error::StrategyError)
- void insert (const string &key, const void *const data, const uint64_t size) throw (Error::ObjectExists, Error::StrategyError)
- void remove (const string &key) throw (Error::ObjectDoesNotExist, Error::StrategyError)
- uint64_t read (const string &key, void *const data) throw (Error::ObjectDoesNotExist, Error::StrategyError)
- virtual void replace (const string &key, const void *const data, const uint64_t size) throw (Error::ObjectDoesNotExist, Error::StrategyError)

- virtual uint64_t length (const string &key) throw (Error::ObjectDoesNotExist, Error::StrategyError)
- void flush (const string &key) throw (Error::ObjectDoesNotExist, Error::StrategyError)
- uint64_t **sequence** (string &key, void *const data, int cursor=BE_RECSTORE_-SEQ_NEXT) throw (Error::ObjectDoesNotExist, Error::StrategyError)
- void setCursor (string &key) throw (Error::ObjectDoesNotExist, Error::StrategyError)
- void changeName (const string &name) throw (Error::ObjectExists, Error::StrategyError)

Protected Member Functions

• string canonicalName (const string &name)

F.8.1 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 will be thrown if the key string is not compliant. A FileRecordStore has the additional requirement that a key name may not contain path delimiter characters ('/' and '\'), or begin with whitespace.

F.8.2 Constructor & Destructor Documentation

F.8.2.1 BiometricEvaluation::IO::FileRecordStore::FileRecordStore (const string & name, const string & description, const string & parentDir) throw (Error::ObjectExists, Error::StrategyError)

Create a new FileRecordStore, read/write mode.

Parameters

```
name[in] The name of the store.description[in] The store's description.parentDir[in] The directory where the store is to be created.
```

Exceptions

Error::ObjectExists The store already exists.

Error::StrategyError An error occurred when accessing the underlying file system.

Open an existing FileRecordStore.

Parameters

```
name[in] The name of the store.parentDir[in] The directory where the store is to be created.mode[in] Open mode, read-only or read-write.
```

Exceptions

Error::ObjectDoesNotExist The store does not exist.

Error::StrategyError An error occurred when accessing the underlying file system.

F.8.3 Member Function Documentation

F.8.3.1 uint64_t BiometricEvaluation::IO::FileRecordStore::getSpaceUsed () throw (Error::StrategyError) [virtual]

Obtain the amount of 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.

Exceptions

Error::StrategyError An error occurred when using the underlying storage system.

Reimplemented from BiometricEvaluation::IO::RecordStore.

F.8.3.2 void BiometricEvaluation::IO::FileRecordStore::insert (const string & key, const void *const data, const uint64_t size) throw (Error::ObjectExists, Error::StrategyError) [virtual]

Insert a record into the store.

Parameters

key[in] The key of the record to be flushed.

data[in] The data for the record.

size[in] The size, in bytes, of the record.

Exceptions

Error:: ObjectExists A record with the given key is already present.

Error::StrategyError An error occurred when using the underlying storage system.

Implements BiometricEvaluation::IO::RecordStore.

F.8.3.3 void BiometricEvaluation::IO::FileRecordStore::remove (const string & key) throw (Error::ObjectDoesNotExist, Error::StrategyError) [virtual]

Remove a record from the store.

Parameters

key[in] The key of the record to be removed.

Exceptions

Error::ObjectDoesNotExist A record for the key does not exist.

Error::StrategyError An error occurred when using the underlying storage system.

Implements BiometricEvaluation::IO::RecordStore.

F.8.3.4 uint64_t BiometricEvaluation::IO::FileRecordStore::read (const string & key, void *const data) throw (Error::ObjectDoesNotExist, Error::StrategyError) [virtual]

Read a complete record from a store. Applications are responsible for allocating storage for the record's data.

Parameters

key[in] The key of the record to be read. [in] Pointer to where the data is to be written.

Returns

The size of the record.

Exceptions

Error::ObjectDoesNotExist A record for the key does not exist.

Error::StrategyError An error occurred when using the underlying storage system.

Implements BiometricEvaluation::IO::RecordStore.

F.8.3.5 virtual void BiometricEvaluation::IO::FileRecordStore::replace (const string & key, const void *const data, const uint64_t size) throw (Error::ObjectDoesNotExist, Error::StrategyError) [virtual]

Replace a complete record in a store.

Parameters

```
key[in] The key of the record to be replaced.data[in] The data for the record.
```

Exceptions

Error::ObjectDoesNotExist A record for the key does not exist.

Error::StrategyError An error occurred when using the underlying storage system.

Implements BiometricEvaluation::IO::RecordStore.

F.8.3.6 virtual uint64_t BiometricEvaluation::IO::FileRecordStore::length (const string & key) throw (Error::ObjectDoesNotExist, Error::StrategyError) [virtual]

Return the length of a record.

Parameters

key[in] The key of the record.

Returns

The record length.

Exceptions

Error::ObjectDoesNotExist A record for the key does not exist.

Error::StrategyError An error occurred when using the underlying storage system.

Implements BiometricEvaluation::IO::RecordStore.

F.8.3.7 void BiometricEvaluation::IO::FileRecordStore::flush (const string & key) throw (Error::ObjectDoesNotExist, Error::StrategyError) [virtual]

Commit the record's data to storage.

Parameters

key[in] The key of the record to be flushed.

Exceptions

Error::ObjectDoesNotExist A record for the key does not exist.

Error::StrategyError An error occurred when using the underlying storage system.

Implements BiometricEvaluation::IO::RecordStore.

F.8.3.8 void BiometricEvaluation::IO::FileRecordStore::setCursor (string & key) throw (Error::ObjectDoesNotExist, Error::StrategyError) [virtual]

Set the sequence cursor to an arbitrary position within the RecordStore, starting at key. Key will be the first record returned from the next call to sequence().

Parameters

key[in] The key of the record which will be returned by the first subsequent call to sequence().

Exceptions

Error::ObjectDoesNotExist A record for the key does not exist.

Error::StrategyError An error occurred when using the underlying storage system.

Implements BiometricEvaluation::IO::RecordStore.

F.8.3.9 void BiometricEvaluation::IO::FileRecordStore::changeName (const string & name) throw (Error::ObjectExists, Error::StrategyError) [virtual]

Change the name of the RecordStore.

Parameters

name[in] The new name for the RecordStore.

Exceptions

Error::StrategyError An error occurred when using the underlying storage system, or the name is malformed.

Reimplemented from BiometricEvaluation::IO::RecordStore.

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

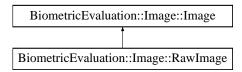
• be_io_filerecstore.h

F.9 BiometricEvaluation::Image::Image Class Reference

A abstract class to represent images and their attributes.

```
#include <be_image_image.h>
```

Inheritance diagram for BiometricEvaluation::Image::Image:



Public Member Functions

• Image (uint8_t *data, uint64_t size, uint64_t width, uint64_t height, unsigned int depth, unsigned int XResolution, unsigned int YResolution)

- virtual unsigned int getXResolution () const =0
- virtual unsigned int getYResolution () const =0
- virtual Utility::AutoArray< uint8_t > getRawData () const =0
- virtual uint64_t getWidth () const =0
- virtual uint64_t getHeight () const =0
- virtual unsigned int getDepth () const =0

Protected Attributes

- uint64_t _width
- uint64 t height
- unsigned int _depth
- unsigned int _XResolution
- unsigned int _YResolution
- Utility::AutoArray< uint8_t > _data

F.9.1 Detailed Description

A abstract class to represent images and their attributes. 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, etc. Implementations of this abstraction provide the getRawData() method to convert image data to 'raw' format.

Image resolution is in pixels per centimeter, while the coordinate system has the origin at the upper left of the image.

Todo

Add more info on the image data, and what coversions are possible.

F.9.2 Constructor & Destructor Documentation

F.9.2.1 BiometricEvaluation::Image::Image::Image (uint8_t * data, uint64_t size, uint64_t width, uint64_t height, unsigned int depth, unsigned int XResolution, unsigned int YResolution)

Parent constructor for all Image classes.

Parameters

```
data[in] The image data.size[in] The size of the image data, in bytes.width[in] The width of the image, in pixels.
```

height[in] The height of the image, in pixels.

depth[in] The image depth, in bits-per-pixel.

XResolution[in] The resolution of the image in the horizontal direction, in pixels-per-centimeter.

YResolution[in] The resolution of the image in the horizontal direction, in pixels-per-centimeter.

F.9.3 Member Function Documentation

F.9.3.1 virtual unsigned int BiometricEvaluation::Image::Image::getXResolution() const [pure virtual]

Accessor for the X-resolution of the image in terms of pixels per centimeter.

Returns

X-resolution (pixel/cm).

Implemented in BiometricEvaluation::Image::RawImage.

F.9.3.2 virtual unsigned int BiometricEvaluation::Image::Image::getYResolution() const [pure virtual]

Accessor for the Y-resolution of the image in terms of pixels per centimeter.

Returns

Y-resolution (pixel/cm).

Implemented in BiometricEvaluation::Image::RawImage.

F.9.3.3 virtual Utility::AutoArray<uint8_t> BiometricEvaluation::Image::Image::getRawData() const [pure virtual]

Accessor for the raw image data. The data returned should not be compressed or encoded.

Returns

Raw image data.

Implemented in BiometricEvaluation::Image::RawImage.

F.9.3.4 virtual uint64_t BiometricEvaluation::Image::Image::getWidth() const [pure virtual]

Accessor for the width of the image in pixels.

Returns

Width of image (pixel).

Implemented in BiometricEvaluation::Image::RawImage.

F.9.3.5 virtual uint64_t BiometricEvaluation::Image::Image::getHeight() const [pure virtual]

Accessor for the height of the image in pixels.

Returns

Height of image (pixel).

Implemented in BiometricEvaluation::Image::RawImage.

F.9.3.6 virtual unsigned int BiometricEvaluation::Image::Image::getDepth() const [pure virtual]

Accessor for the color depth of the image in bits.

Returns

The color depth of the image (bit).

Implemented in BiometricEvaluation::Image::RawImage.

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

• be_image_image.h

F.10 BiometricEvaluation::IO::LogCabinet Class Reference

#include <be_io_logcabinet.h>

Public Member Functions

- LogCabinet (const string &name, const string &description, const string &parentDir) throw (Error::ObjectExists, Error::StrategyError)
- LogCabinet (const string &name, const string &parentDir) throw (Error::ObjectDoesNotExist, Error::StrategyError)
- LogSheet * newLogSheet (const string &name, const string &description) throw (Error::ObjectExists, Error::StrategyError)
- string getName ()
- string getDescription ()
- unsigned int getCount ()

Static Public Member Functions

static void remove (const string &name, const string &parentDir) throw (Error::ObjectDoesNotExist, Error::StrategyError)

Protected Member Functions

- string canonicalName (const string &name)
- void **readControlFile** () throw (Error::StrategyError)
- void writeControlFile () throw (Error::StrategyError)

Protected Attributes

- string _name
- string _parentDir
- string _directory
- string _description
- unsigned int _count
- int _cursor

F.10.1 Detailed Description

A class to represent a collection of log sheets.

F.10.2 Constructor & Destructor Documentation

F.10.2.1 BiometricEvaluation::IO::LogCabinet::LogCabinet (const string & name, const string & description, const string & parentDir) throw (Error::ObjectExists, Error::StrategyError)

Create a new LogCabinet in the file system.

Parameters

name[in] The name of the LogCabinet to be created.description[in] The text used to describe the cabinet.parentDir[in] Where, in the file system, the cabinet is to be stored. This directory must exist.

Returns

An object representing the new log cabinet.

Exceptions

Error:: ObjectExists The cabinet was previously created.

Error::StrategyError

Error::StrategyError An error occurred when using the underlying file system, or name or parentDir is malformed.

F.10.2.2 BiometricEvaluation::IO::LogCabinet::LogCabinet (const string & name, const string & parentDir) throw (Error::ObjectDoesNotExist, Error::StrategyError)

Open an existing LogCabinet.

Parameters

```
name[in] The name of the LogCabinet to be created.description[in] The text used to describe the cabinet.parentDir[in] Where, in the file system, the cabinet is to be stored. This directory must exist.
```

Returns

An object representing the log cabinet.

Exceptions

Error::ObjectDoesNotExist The cabinet does not exist in the file system.Error::StrategyError An error occurred when using the underlying file system, or name or parentDir is malformed.

F.10.3 Member Function Documentation

F.10.3.1 LogSheet* BiometricEvaluation::IO::LogCabinet::newLogSheet (const string & name, const string & description) throw (Error::ObjectExists, Error::StrategyError)

Create a new LogSheet within the LogCabinet.

Parameters

name[in] The name of the LogSheet to be created.

description[in] The text used to describe the sheet. This text is written into the log file prior to any entries.

parentDir[in] Where, in the file system, the sheet is to be stored. This directory
must exist.

Returns

An object pointer to the new log sheet.

Exceptions

Error::ObjectExists The sheet was previously created.

Error::StrategyError An error occurred when using the underlying file system, or name or parentDir is malformed.

F.10.3.2 string BiometricEvaluation::IO::LogCabinet::getName ()

Obtain the name of the LogCabinet.

@ returns The name of the LogCabinet.

F.10.3.3 string BiometricEvaluation::IO::LogCabinet::getDescription ()

Obtain the description of the LogCabinet.

@ returns The description of the LogCabinet.

F.10.3.4 unsigned int BiometricEvaluation::IO::LogCabinet::getCount()

Obtain the number of items in the LogCabinet.

@ returns The number of LogSheets manages by the cabinet.

F.10.3.5 static void BiometricEvaluation::IO::LogCabinet::remove (const string & name, const string & parentDir) throw (Error::ObjectDoesNotExist, Error::StrategyError) [static]

Remove a LogCabinet.

Parameters

name[in] The name of the LogCabinet to be removed.

parentDir[in] Where, in the file system, the sheet is to be stored. This directory
must exist.

Exceptions

Error:: ObjectDoesNotExist The LogCabinet does not exist.

Error::StrategyError An error occurred when using the underlying file system, or name or parentDir is malformed.

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

• be_io_logcabinet.h

F.11 BiometricEvaluation::IO::LogSheet Class Reference

A class to represent a single logging mechanism.

```
#include <be_io_logcabinet.h>
```

Public Member Functions

- LogSheet (const string &name, const string &description, const string &parent-Dir) throw (Error::ObjectExists, Error::StrategyError)
- void write (const string &entry) throw (Error::StrategyError)
- void newEntry () throw (Error::StrategyError)
- string getCurrentEntry ()
- void resetCurrentEntry ()
- uint32_t getCurrentEntryNumber ()
- void sync () throw (Error::StrategyError)
- void setAutoSync (bool state)

F.11.1 Detailed Description

A class to represent a single logging mechanism. A LogSheet 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(). Entries in the log file are prefixed with an entry number, which is incremented when the entry is written (either by directly calling write(), or calling newEntry()).

A LogSheet object can be constructed and passed back to the client by the LogCabinet object. All sheets created in the manner are placed in a common area maintained by the cabinet.

Note

By default, the entries in the LogSheet may not be immediately written to the file system, depending on the buffering behavior of the operating system. Applications can force a write by invoking sync(), or force a write at every new log entry by invoking setAutoSync(true).

F.11.2 Constructor & Destructor Documentation

F.11.2.1 BiometricEvaluation::IO::LogSheet::LogSheet (const string & name, const string & description, const string & parentDir) throw (Error::ObjectExists, Error::StrategyError)

Create a new log sheet.

Parameters

name[in] The name of the LogSheet to be created.

description[in] The text used to describe the sheet. This text is written into the log file prior to any entries.

parentDir[in] Where, in the file system, the sheet is to be stored. This directory
must exist.

Returns

An object representing the new log sheet.

Exceptions

Error::ObjectExists The sheet was previously created.

Error::StrategyError An error occurred when using the underlying file system, or name or parentDir is malformed.

F.11.3 Member Function Documentation

F.11.3.1 void BiometricEvaluation::IO::LogSheet::write (const string & entry) throw (Error::StrategyError)

Write a string as an entry to the log file. This does not affect the current log entry buffer, but does increment the entry number.

Parameters

entry[in] The text of the log entry.

Exceptions

Error::StrategyError An error occurred when using the underlying file system.

F.11.3.2 void BiometricEvaluation::IO::LogSheet::newEntry () throw (Error::StrategyError)

Start a new entry, causing the existing entry to be closed. Applications do not have to call this method for the first entry, however, as the stream is ready for writing upon construction.

Exceptions

Error::StrategyError An error occurred when using the underlying file system.

F.11.3.3 string BiometricEvaluation::IO::LogSheet::getCurrentEntry()

Obtain the contents of the current entry currently under construction.

Returns

The text of the current entry.

F.11.3.4 void BiometricEvaluation::IO::LogSheet::resetCurrentEntry()

Reset the current entry buffer to the beginning.

F.11.3.5 uint32_t BiometricEvaluation::IO::LogSheet::getCurrentEntryNumber ()

Obtain the current entry number.

Returns

The current entry number.

F.11.3.6 void BiometricEvaluation::IO::LogSheet::sync () throw (Error::StrategyError)

Synchronize any buffered data to the underlying log file. This syncing is dependent on the behavior of the underlying filesystem and operating system.

Exceptions

Error::StrategyError An error occurred when using the underlying file system.

F.11.3.7 void BiometricEvaluation::IO::LogSheet::setAutoSync (bool state)

Turn on/off auto-sync of the data. Applications can gain loggin performance by turning off auto-sysnc, or gain reliability by turning it on.

Parameters

state When true, the data is sync'd whenever newEntry() is or write() is called. When false, sync() must be called to force a write.

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

· be io logcabinet.h

F.12 BiometricEvaluation::IO::ManifestEntry Struct Reference

Public Attributes

- · long offset
- uint64_t size

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

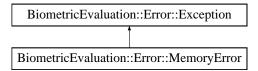
• be_io_archiverecstore.h

F.13 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 (string info)

F.13.1 Detailed Description

An error occurred when allocating an object.

F.13.2 Constructor & Destructor Documentation

F.13.2.1 BiometricEvaluation::Error::MemoryError::MemoryError ()

Construct a MemoryError object with the default information string.

Returns

The MemoryError object.

F.13.2.2 BiometricEvaluation::Error::MemoryError::MemoryError (string info)

Construct a MemoryError object with an information string appended to the default information string.

Returns

The MemoryError object.

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

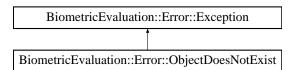
• be_error_exception.h

F.14 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 (string info)

F.14.1 Detailed Description

The named object does not exist.

F.14.2 Constructor & Destructor Documentation

F.14.2.1 BiometricEvaluation::Error::ObjectDoesNotExist::ObjectDoesNotExist

Construct a ObjectDoesNotExist object with the default information string.

Returns

The ObjectDoesNotExist object.

F.14.2.2 BiometricEvaluation::Error::ObjectDoesNotExist::ObjectDoesNotExist (string info)

Construct a ObjectDoesNotExist object with an information string appended to the default information string.

Returns

The ObjectDoesNotExist object.

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

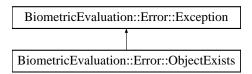
• be_error_exception.h

F.15 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 (string info)

F.15.1 Detailed Description

The named object exists and will not be replaced.

F.15.2 Constructor & Destructor Documentation

F.15.2.1 BiometricEvaluation::Error::ObjectExists::ObjectExists()

Construct a ObjectExists object with the default information string.

Returns

The ObjectExists object.

F.15.2.2 BiometricEvaluation::Error::ObjectExists::ObjectExists (string info)

Construct a ObjectExists object with an information string appended to the default information string.

Returns

The ObjectExists object.

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

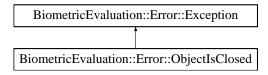
• be_error_exception.h

F.16 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 (string info)

F.16.1 Detailed Description

The object is closed.

F.16.2 Constructor & Destructor Documentation

F.16.2.1 BiometricEvaluation::Error::ObjectIsClosed::ObjectIsClosed()

Construct a ObjectIsClosed object with the default information string.

Returns

The ObjectIsClosed object.

F.16.2.2 BiometricEvaluation::Error::ObjectIsClosed::ObjectIsClosed (string info)

Construct a ObjectIsClosed object with an information string appended to the default information string.

Returns

The ObjectIsClosed object.

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

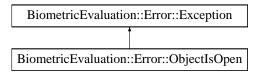
• be_error_exception.h

F.17 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 (string info)

F.17.1 Detailed Description

The object is already opened.

F.17.2 Constructor & Destructor Documentation

F.17.2.1 BiometricEvaluation::Error::ObjectIsOpen::ObjectIsOpen()

Construct a ObjectIsOpen object with the default information string.

Returns

The ObjectIsOpen object.

F.17.2.2 Biometric Evaluation::Error::ObjectIsOpen::ObjectIsOpen (string info)

Construct a ObjectIsOpen object with an information string appended to the default information string.

Returns

The ObjectIsOpen object.

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

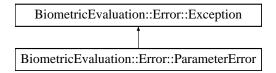
• be_error_exception.h

F.18 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 (string info)

F.18.1 Detailed Description

An invalid parameter was passed to a constructor or method.

F.18.2 Constructor & Destructor Documentation

F.18.2.1 BiometricEvaluation::Error::ParameterError::ParameterError ()

Construct a ParameterError object with the default information string.

Returns

The ParameterError object.

F.18.2.2 BiometricEvaluation::Error::ParameterError::ParameterError (string info)

Construct a ParameterError object with an information string appended to the default information string.

Returns

The ParameterError object.

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

• be_error_exception.h

F.19 BiometricEvaluation::IO::Properties Class Reference

A Properties class is used to maintain key/value pairs of strings, with each property matched to one value.

```
#include <be_io_properties.h>
```

Public Types

• typedef PropertiesMap::const_iterator Properties_iter

Public Member Functions

- Properties (const string &filename, uint8_t mode=IO::READWRITE) throw (Error::StrategyError, Error::FileError)
- void setProperty (const string &property, const string &value) throw (Error::StrategyError)
- void setPropertyFromInteger (const string &property, int64_t value) throw (Error::StrategyError)
- void removeProperty (const string &property) throw (Error::ObjectDoesNotExist, Error::StrategyError)
- string getProperty (const string &property) throw (Error::ObjectDoesNotExist)
- int64_t getPropertyAsInteger (const string &property) throw (Error::ObjectDoesNotExist, Error::ConversionError)
- void sync () throw (Error::FileError, Error::StrategyError)
- void changeName (const string &filename) throw (Error::StrategyError)

F.19.1 Detailed Description

A Properties class is used to maintain key/value pairs of strings, with each property matched to one value. The properties are read from a file that is specified in the constructor, and will be created if it does not exist.

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 a 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.

F.19.2 Constructor & Destructor Documentation

F.19.2.1 BiometricEvaluation::IO::Properties::Properties (const string & filename, uint8_t mode = IO::READWRITE) throw (Error::StrategyError, Error::FileError)

Construct a new Properties object from an existing or to be created properties file. The constructor will create the file when it does not exist.

Parameters

filename[in] The name of the file to store the properties. This can be the empty string, meaning the properties are to be stored in memory only.

mode[in] The read/write mode of the object.

Returns

An object representing the properties set.

Exceptions

Error::StrategyError A line in the properties file is malformed.

Error::FileError An error occurred when using the underlying storage system.

F.19.3 Member Function Documentation

F.19.3.1 void BiometricEvaluation::IO::Properties::setProperty (const string & property, const string & value) throw (Error::StrategyError)

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

```
property[in] The name of the property to set.
value[in] The value associated with the property.
```

Exceptions

Error::StrategyError The Properties object is read-only.

F.19.3.2 void BiometricEvaluation::IO::Properties::setPropertyFromInteger (const string & property, int64_t value) throw (Error::StrategyError)

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

```
property[in] The name of the property to set.
value[in] The value associated with the property.
```

Exceptions

Error::StrategyError The Properties object is read-only.

Remove a property.

Parameters

property[in] The name of the property to set.

Exceptions

```
Error::ObjectDoesNotExist The named property does not exist.Error::StrategyError The Properties object is read-only.
```

F.19.3.4 string BiometricEvaluation::IO::Properties::getProperty (const string & property) throw (Error::ObjectDoesNotExist)

Retrieve a property value as a string object.

Parameters

property[in] The name of the property to get.

Exceptions

Error::ObjectDoesNotExist The named property does not exist.

F.19.3.5 int64_t BiometricEvaluation::IO::Properties::getPropertyAsInteger (const string & property) throw (Error::ObjectDoesNotExist, Error::ConversionError)

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

property[in] The name of the property to get.

Exceptions

Error::ObjectDoesNotExist The named property does not exist.

Error::ConversionError The property value cannot be converted, usually due to non-numeric characters in the string.

F.19.3.6 void BiometricEvaluation::IO::Properties::sync () throw (Error::FileError, Error::StrategyError)

Write the properties to the underlying file, synchronizing the in-memory and on-disk versions.

Exceptions

Error::FileError An error occurred when using the underlying storage system.

Error::StrategyError The object was constructed with NULL as the file name, or is read-only.

F.19.3.7 void BiometricEvaluation::IO::Properties::changeName (const string & filename) throw (Error::StrategyError)

Change the name of the Properties, which means changing the name of the underlying file that stores the properties. The empty string ("") can be used to indicate no backing file.

Note

No check is made that the file is writeable at this time.

Parameters

filename[in] The name of the properties file.

Exceptions

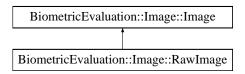
Error::StrategyError The object is read-only.

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

• be_io_properties.h

F.20 BiometricEvaluation::Image::RawImage Class Reference

Inheritance diagram for BiometricEvaluation::Image::RawImage:



Public Member Functions

- RawImage (uint8_t *_data, uint64_t size, uint64_t width, uint64_t height, unsigned int depth, unsigned int XResolution, unsigned int YResolution)
- uint64_t getWidth () const
- uint64_t getHeight () const
- unsigned int getDepth () const
- unsigned int getXResolution () const
- unsigned int getYResolution () const
- $Utility::AutoArray < uint8_t > getRawData$ () const

F.20.1 Constructor & Destructor Documentation

F.20.1.1 BiometricEvaluation::Image::RawImage::RawImage (uint8_t * __data, uint64_t size, uint64_t width, uint64_t height, unsigned int depth, unsigned int XResolution, unsigned int YResolution)

Construct a RawImage object.

Parameters

```
data[in] The image data.
```

size[in] The size of the image data, in bytes.

width[in] The width of the image, in pixels.

height[in] The height of the image, in pixels.

depth[in] The image depth, in bits-per-pixel.

XResolution[in] The resolution of the image in the horizontal direction, in pixels-per-centimeter.

YResolution[in] The resolution of the image in the horizontal direction, in pixelsper-centimeter.

F.20.2 Member Function Documentation

F.20.2.1 uint64_t BiometricEvaluation::Image::RawImage::getWidth() const [virtual]

Accessor for the width of the image in pixels.

Returns

Width of image (pixel).

Implements BiometricEvaluation::Image::Image.

F.20.2.2 uint64_t BiometricEvaluation::Image::RawImage::getHeight() const [virtual]

Accessor for the height of the image in pixels.

Returns

Height of image (pixel).

Implements BiometricEvaluation::Image::Image.

F.20.2.3	unsigned int BiometricEvaluation::Image::RawImage::getDepth ()
	const [virtual]	

Accessor for the color depth of the image in bits.

Returns

The color depth of the image (bit).

Implements BiometricEvaluation::Image::Image.

F.20.2.4 unsigned int BiometricEvaluation::Image::RawImage::getXResolution () const [virtual]

Accessor for the X-resolution of the image in terms of pixels per centimeter.

Returns

X-resolution (pixel/cm).

Implements BiometricEvaluation::Image::Image.

F.20.2.5 unsigned int BiometricEvaluation::Image::RawImage::getYResolution () const [virtual]

Accessor for the Y-resolution of the image in terms of pixels per centimeter.

Returns

Y-resolution (pixel/cm).

Implements BiometricEvaluation::Image::Image.

F.20.2.6 Utility::AutoArray<uint8_t> BiometricEvaluation::Image::RawImage::getRawData() const [virtual]

Accessor for the raw image data. The data returned should not be compressed or encoded.

Returns

Raw image data.

Implements BiometricEvaluation::Image::Image.

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

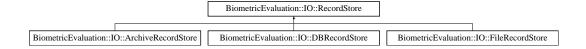
• be image rawimage.h

F.21 BiometricEvaluation::IO::RecordStore Class Reference

A class to represent a data storage mechanism.

```
#include <be io recordstore.h>
```

Inheritance diagram for BiometricEvaluation::IO::RecordStore:



Public Member Functions

- RecordStore (const string &name, const string &description, const string &type, const string &parentDir) throw (Error::ObjectExists, Error::StrategyError)
- RecordStore (const string &name, const string &parentDir, uint8_t mode=READWRITE) throw (Error::ObjectDoesNotExist, Error::StrategyError)
- string getName ()
- string getDescription ()
- unsigned int getCount ()
- virtual void changeName (const string &name) throw (Error::ObjectExists, Error::StrategyError)
- virtual void changeDescription (const string &description) throw (Error::StrategyError)
- virtual uint64_t getSpaceUsed () throw (Error::StrategyError)
- virtual void sync () throw (Error::StrategyError)
- virtual void insert (const string &key, const void *const data, const uint64_t size)=0 throw (Error::ObjectExists, Error::StrategyError)
- virtual void remove (const string &key)=0 throw (Error::ObjectDoesNotExist, Error::StrategyError)
- virtual uint64_t read (const string &key, void *const data)=0 throw (Error::ObjectDoesNotExist, Error::StrategyError)
- virtual void replace (const string &key, const void *const data, const uint64_t size)=0 throw (Error::ObjectDoesNotExist, Error::StrategyError)

- virtual uint64_t length (const string &key)=0 throw (Error::ObjectDoesNotExist, Error::StrategyError)
- virtual void flush (const string &key)=0 throw (Error::ObjectDoesNotExist, Error::StrategyError)
- virtual uint64_t sequence (string &key, void *const data=NULL, int cursor=BE_RECSTORE_SEQ_NEXT)=0 throw (Error::ObjectDoesNotExist, Error::StrategyError)
- virtual void setCursor (string &key)=0 throw (Error::ObjectDoesNotExist, Error::StrategyError)

Static Public Member Functions

 static void removeRecordStore (const string &name, const string &parentDir) throw (Error::ObjectDoesNotExist, Error::StrategyError)

Static Public Attributes

- static const string CONTROLFILENAME
- static const string NAMEPROPERTY
- static const string **DESCRIPTIONPROPERTY**
- static const string COUNTPROPERTY
- static const string TYPEPROPERTY
- static const string BERKELEYDBTYPE
- static const string ARCHIVETYPE
- static const string **FILETYPE**
- static const int BE_RECSTORE_SEQ_START = 1
- static const int **BE_RECSTORE_SEQ_NEXT** = 2

Protected Member Functions

- string **canonicalName** (const string &name)
- void **readControlFile** () throw (Error::StrategyError)
- void writeControlFile () throw (Error::StrategyError)

Protected Attributes

- string _name
- string _description
- string _type
- string _directory
- string _parentDir

- unsigned int _count
- int cursor
- uint8_t _mode

F.21.1 Detailed Description

A class to represent a data storage mechanism. A RecordStore is an abstraction that associates keys with a specific record. 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.

See also

IO::ArchiveRecordStore, IO::DBRecordStore, IO::FileRecordStore.

F.21.2 Constructor & Destructor Documentation

F.21.2.1 BiometricEvaluation::IO::RecordStore::RecordStore (const string & name, const string & description, const string & type, const string & parentDir) throw (Error::ObjectExists, Error::StrategyError)

Constructor to create a new RecordStore.

Parameters

```
name[in] The name of the RecordStore to be created.
```

description[in] The text used to describe the store.

type[in] The type of RecordStore.

parentDir[in] Where, in the file system, the store is to be rooted. This directory
must exist.

Returns

An object representing the new, empty store.

Exceptions

Error::ObjectExists The store was previously created, or the directory where it would be created exists.

Error::StrategyError An error occurred when using the underlying storage system, or the the name malformed.

F.21.2.2 BiometricEvaluation::IO::RecordStore::RecordStore (const string & name, const string & parentDir, uint8_t mode = READWRITE) throw (Error::ObjectDoesNotExist, Error::StrategyError)

Constructor to open an existing RecordStore.

Parameters

```
name[in] The name of the store to be opened.parentDir[in] Where, in the file system, the store is rooted.mode[in] The type of access a client of this RecordStore has.
```

Returns

An object representing the existing store.

Exceptions

Error::ObjectDoesNotExist The RecordStore does not exist.

Error::StrategyError An error occurred when using the underlying storage system, or the name is malformed.

F.21.3 Member Function Documentation

F.21.3.1 string BiometricEvaluation::IO::RecordStore::getName()

Return the name of the RecordStore.

Returns

The RecordStore's name.

F.21.3.2 string BiometricEvaluation::IO::RecordStore::getDescription ()

Obtain a textual description of the RecordStore.

Returns

The RecordStore's description.

F.21.3.3 unsigned int BiometricEvaluation::IO::RecordStore::getCount()

Obtain the number of items in the RecordStore.

Returns

The number of items in the RecordStore.

F.21.3.4 virtual void BiometricEvaluation::IO::RecordStore::changeName (const string & name) throw (Error::ObjectExists, Error::StrategyError) [virtual]

Change the name of the RecordStore.

Parameters

name[in] The new name for the RecordStore.

Exceptions

Error::StrategyError An error occurred when using the underlying storage system, or the name is malformed.

Reimplemented in BiometricEvaluation::IO::ArchiveRecordStore, BiometricEvaluation::IO::DBRecordStore, and BiometricEvaluation::IO::FileRecordStore.

F.21.3.5 virtual void BiometricEvalua-

```
tion::IO::RecordStore::changeDescription ( const
string & description ) throw (Error::StrategyError) [virtual]
```

Change the description of the RecordStore.

Parameters

description[in] The new description.

Exceptions

Error::StrategyError An error occurred when using the underlying storage system.

$\begin{tabular}{ll} F.21.3.6 & virtual uint 64_t \ Biometric Evaluation :: IO:: Record Store :: get Space Used \\ & () throw (Error :: Strategy Error) & [virtual] \end{tabular}$

Obtain the amount of 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.

Exceptions

Error::StrategyError An error occurred when using the underlying storage system.

Reimplemented in BiometricEvaluation::IO::ArchiveRecordStore, BiometricEvaluation::IO::DBRecordStore, and BiometricEvaluation::IO::FileRecordStore.

F.21.3.7 virtual void BiometricEvaluation::IO::RecordStore::sync () throw (Error::StrategyError) [virtual]

Synchronize the entire record store to persistent storage.

Exceptions

Error::StrategyError An error occurred when using the underlying storage system.

Reimplemented in BiometricEvaluation::IO::ArchiveRecordStore, and BiometricEvaluation::IO::DBRecordStore.

F.21.3.8 virtual void BiometricEvaluation::IO::RecordStore::insert (const string & key, const void *const data, const uint64_t size) throw (Error::ObjectExists, Error::StrategyError) [pure virtual]

Insert a record into the store.

Parameters

```
key[in] The key of the record to be flushed.data[in] The data for the record.size[in] The size, in bytes, of the record.
```

Exceptions

Error::ObjectExists A record with the given key is already present.

Error::StrategyError An error occurred when using the underlying storage system.

Implemented in BiometricEvaluation::IO::ArchiveRecordStore, BiometricEvaluation::IO::DBRecordStore, and BiometricEvaluation::IO::FileRecordStore.

F.21.3.9 virtual void BiometricEvaluation::IO::RecordStore::remove (const string & key) throw (Error::ObjectDoesNotExist, Error::StrategyError) [pure virtual]

Remove a record from the store.

Parameters

key[in] The key of the record to be removed.

Exceptions

Error::ObjectDoesNotExist A record for the key does not exist.

Error::StrategyError An error occurred when using the underlying storage system.

Implemented in BiometricEvaluation::IO::ArchiveRecordStore, BiometricEvaluation::IO::DBRecordStore, and BiometricEvaluation::IO::FileRecordStore.

F.21.3.10 virtual uint64_t BiometricEvaluation::IO::RecordStore::read (const string & key, void *const data) throw (Error::ObjectDoesNotExist, Error::StrategyError) [pure virtual]

Read a complete record from a store. Applications are responsible for allocating storage for the record's data.

Parameters

key[in] The key of the record to be read. [in] Pointer to where the data is to be written.

Returns

The size of the record.

Exceptions

Error::ObjectDoesNotExist A record for the key does not exist.

Error::StrategyError An error occurred when using the underlying storage system.

Implemented in BiometricEvaluation::IO::ArchiveRecordStore, BiometricEvaluation::IO::DBRecordStore, and BiometricEvaluation::IO::FileRecordStore.

F.21.3.11 virtual void BiometricEvaluation::IO::RecordStore::replace (const string & key, const void *const data, const uint64_t size) throw (Error::ObjectDoesNotExist, Error::StrategyError) [pure virtual]

Replace a complete record in a store.

Parameters

key[in] The key of the record to be replaced.data[in] The data for the record.

Exceptions

Error::ObjectDoesNotExist A record for the key does not exist.

Error::StrategyError An error occurred when using the underlying storage system.

Implemented in BiometricEvaluation::IO::ArchiveRecordStore, BiometricEvaluation::IO::DBRecordStore, and BiometricEvaluation::IO::FileRecordStore.

F.21.3.12 virtual uint64_t BiometricEvaluation::IO::RecordStore::length (const string & key) throw (Error::ObjectDoesNotExist, Error::StrategyError) [pure virtual]

Return the length of a record.

Parameters

key[in] The key of the record.

Returns

The record length.

Exceptions

Error::ObjectDoesNotExist A record for the key does not exist.

Error::StrategyError An error occurred when using the underlying storage system.

Implemented in BiometricEvaluation::IO::ArchiveRecordStore, BiometricEvaluation::IO::DBRecordStore, and BiometricEvaluation::IO::FileRecordStore.

F.21.3.13 virtual void BiometricEvaluation::IO::RecordStore::flush (const string & key) throw (Error::ObjectDoesNotExist, Error::StrategyError) [pure virtual]

Commit the record's data to storage.

Parameters

key[in] The key of the record to be flushed.

Exceptions

Error::ObjectDoesNotExist A record for the key does not exist.

Error::StrategyError An error occurred when using the underlying storage system.

Implemented in BiometricEvaluation::IO::ArchiveRecordStore, BiometricEvaluation::IO::DBRecordStore, and BiometricEvaluation::IO::FileRecordStore.

F.21.3.14 virtual void BiometricEvaluation::IO::RecordStore::setCursor (string & key) throw (Error::ObjectDoesNotExist, Error::StrategyError) [pure virtual]

Set the sequence cursor to an arbitrary position within the RecordStore, starting at key. Key will be the first record returned from the next call to sequence().

Parameters

key[in] The key of the record which will be returned by the first subsequent call to sequence().

Exceptions

Error::ObjectDoesNotExist A record for the key does not exist.

Error::StrategyError An error occurred when using the underlying storage system.

Implemented in BiometricEvaluation::IO::ArchiveRecordStore, BiometricEvaluation::IO::DBRecordStore, and BiometricEvaluation::IO::FileRecordStore.

F.21.3.15 static void BiometricEvalua-

```
tion::IO::RecordStore::removeRecordStore ( const string & name, const string & parentDir ) throw (Error::ObjectDoesNotExist, Error::StrategyError) [static]
```

Remove a RecordStore by deleting all persistant data associated with the store.

Parameters

```
name[in] The name of the existing RecordStore.parentDir[in] Where, in the file system, the store is rooted.
```

Exceptions

Error::ObjectDoesNotExist A record with the given key does not exist.
Error::StrategyError An error occurred when using the underlying storage system.

F.21.4 Member Data Documentation

F.21.4.1 const string BiometricEvaluation::IO::RecordStore::CONTROLFILENAME [static]

The name of the control file, a properties list.

F.21.4.2 const string BiometricEvaluation::IO::RecordStore::NAMEPROPERTY [static]

Keys used in the Properties list for the RecordStore.

"Name" - The name of the store "Description" - The description of the store "Count" - The number of items in the store "Type" - The type of RecordStore.

F.21.4.3 const string BiometricEvaluation::IO::RecordStore::BERKELEYDBTYPE [static]

The known RecordStore type strings: "BerkeleyDB" - Berkeley database "Archive" - Archive file "File" - One file per record

F.21.4.4 const int BiometricEvaluation::IO::RecordStore::BE_RECSTORE_-SEQ_START = 1 [static]

Sequence through a RecordStore, returning the key/data pairs. Sequencing means to start at some point in the store and return the record, then repeatedly calling the sequencor to return the next record. The starting point is typically the the first record, and is set to that when the RecordStore object is created. The starting point can be reset by calling this method with the cursor parameter set to BE_RECSTORE_SEQ_START.

Parameters

key[out] The key of the currently sequenced record.

data[in] Pointer to where the data is to be written. Applications can set data to NULL to indicate only the key is wanted.

cursor[in] The location within the sequence of the key/data pair to return.

Returns

The length of the record currently in sequence.

Exceptions

Error::ObjectDoesNotExist A record for the key does not exist.

Error::StrategyError An error occurred when using the underlying storage system.

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

• be_io_recordstore.h

F.22 BiometricEvaluation::Error::SignalManager Class Reference

A Signal Manager object is used to handle signals that come from the operating system.

```
#include <be_error_signal_manager.h>
```

Public Member Functions

- SignalManager () throw (Error::StrategyError)
- **SignalManager** (const sigset_t signalSet) throw (Error::ParameterError)
- void setSignalSet (const sigset_t signalSet) throw (Error::ParameterError)
- void clearSignalSet ()
- void setDefaultSignalSet ()
- bool sigHandled ()
- void start () throw (Error::StrategyError)
- void stop () throw (Error::StrategyError)
- void setSigHandled ()
- void clearSigHandled ()

Static Public Attributes

- static bool _canSigJump
- static sigjmp_buf _sigJumpBuf

F.22.1 Detailed Description

A SignalManager object is used to handle signals that come from the operating system. Applications typically do not invoke most methods of a SignalManager, except the setSignalSet(), setDefaultSignalSet(), and sigHandled(). An application wishing

to just catch memory errors can simply construct a SignalManager object, and invoke sigHandled() 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 Signal-Manager object to start handling signals. Applications can call either setSignalSet() or setDefaultSignalSet() 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.

A SignalManager is passive (i.e. no signal handlers are installed) until that start() method is called, and becomes passive when stop() is invoked. The signals that are to be handled by the object are maitained as state, and the set of signals can be changed at any time, but are not in effect until start() is called.

Attention

The start(), stop(), setSigHandled() and clearSigHandled() methods are not meant to be used directly by applications, which should use the BEGIN_SIGNAL_BLOCK()/END_SIGNAL_BLOCK() macro pair.

F.22.2 Constructor & Destructor Documentation

F.22.2.1 BiometricEvaluation::Error::SignalManager::SignalManager () throw (Error::StrategyError)

Construct a new SignalManager object with the default signal handling: SIGSEGV and SIGBUS.

Returns

The SignalManager.

Exceptions

Error::StrategyError Could not register the signal handler.

F.22.3 Member Function Documentation

F.22.3.1 void BiometricEvaluation::Error::SignalManager::setSignalSet (const sigset_t signalSet) throw (Error::ParameterError)

Set the signals this object will manage.

Parameters

signalSet (in) The signal set; see sigaction(2), sigemptyset(3) and sigaddset(3).

Exceptions

Error::ParameterError One of the signals in signalSet cannot be handled (SIGKILL, SIGSTOP.).

F.22.3.2 void BiometricEvaluation::Error::SignalManager::clearSignalSet ()

Clear all signal handling.

```
F.22.3.3 void BiometricEvaluation::Error::SignalManager::setDefaultSignalSet (
```

Set the default signals this object will manage: SIGSEGV and SIGBUS.

F.22.3.4 bool BiometricEvaluation::Error::SignalManager::sigHandled ()

Indicate whether a signal was handled.

Returns

true if a signal was handled, false otherwise.

F.22.3.5 void BiometricEvaluation::Error::SignalManager::start () throw (Error::StrategyError)

Start handling signals of the current signal set.

Exceptions

Error::StrategyError Could not register the signal handler.

Note

If an application invokes start() 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.

F.22.3.6 void BiometricEvaluation::Error::SignalManager::stop () throw (Error::StrategyError)

Stop handling signals of the current signal set.

Exceptions

Error::StrategyError Could not register the signal handler.

F.22.3.7 void BiometricEvaluation::Error::SignalManager::setSigHandled ()

Set a flag to indicate a signal was handled.

F.22.3.8 void BiometricEvaluation::Error::SignalManager::clearSigHandled (

Clear the indication that a signal was handled.

F.22.4 Member Data Documentation

F.22.4.1 bool BiometricEvaluation::Error::SignalManager::_canSigJump [static]

Flag indicating can jump after handling a signal.

Note

Should not be directly used by applications.

F.22.4.2 sigjmp_buf BiometricEvaluation::Error::SignalManager::_sigJumpBuf [static]

The jump buffer used by the signal handler.

Note

Should not be directly used by applications.

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

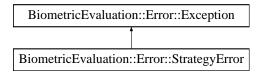
• be_error_signal_manager.h

F.23 BiometricEvaluation::Error::StrategyError Class Reference

A StrategyError 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 (string info)

F.23.1 Detailed Description

A StrategyError object is thrown when the underlying implementation of this interface encounters an error.

F.23.2 Constructor & Destructor Documentation

F.23.2.1 BiometricEvaluation::Error::StrategyError::StrategyError()

Construct a StrategyError object with the default information string.

Returns

The StrategyError object.

F.23.2.2 BiometricEvaluation::Error::StrategyError::StrategyError (string info)

Construct a StrategyError object with an information string appended to the default information string.

Returns

The StrategyError object.

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

• be_error_exception.h

F.24 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 Member Functions

- Timer ()
- void start () throw (Error::StrategyError)
- void stop () throw (Error::StrategyError)
- uint64 t elapsed () throw (Error::StrategyError)

F.24.1 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() and Timer::stop() calls, then use Timer::elapsed() to obtain the calculated time of the operation.

F.24.2 Constructor & Destructor Documentation

F.24.2.1 BiometricEvaluation::Time::Timer::Timer ()

Constructor for the Timer object.

F.24.3 Member Function Documentation

F.24.3.1 void BiometricEvaluation::Time::Timer::start () throw (Error::StrategyError)

Start tracking time.

Exceptions

Error::StrategyError This object is currently timing an operation or an error occurred when obtaining timing information.

F.24.3.2 void BiometricEvaluation::Time::Timer::stop () throw (Error::StrategyError)

Stop tracking time.

Exceptions

Error::StrategyError This object is not currently timing an operation or an error occurred when obtaining timing information.

F.24.3.3 uint64_t BiometricEvaluation::Time::Timer::elapsed () throw (Error::StrategyError)

Get the elapsed time in microseconds between calls to this object's start() and stop() methods.

Returns

The number of microseconds between calls to this object's start() and stop() methods.

Exceptions

Error::StrategyError This object is currently timing an operation or an error occurred when obtaining timing information.

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

• be_time_timer.h

F.25 BiometricEvaluation::Error::Utility Class Reference

This class contains methods that are useful utility functions, such as converting system values to strings.

```
#include <be_error_utility.h>
```

Static Public Member Functions

• static string errorStr ()

F.25.1 Detailed Description

This class contains methods that are useful utility functions, such as converting system values to strings.

F.25.2 Member Function Documentation

F.25.2.1 static string BiometricEvaluation::Error::Utility::errorStr () [static]

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

Returns

The current error message specified by errno.

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

· be_error_utility.h

F.26 BiometricEvaluation::IO::Utility Class Reference

#include <be_io_utility.h>

Static Public Member Functions

- static void removeDirectory (const string &directory, const string &prefix) throw (Error::ObjectDoesNotExist, Error::StrategyError)
- static uint64_t getFileSize (const string &pathname) throw (Error::ObjectDoesNotExist, Error::StrategyError)
- static bool fileExists (const string &pathname) throw (Error::StrategyError)
- static bool validateRootName (const string &name)
- static bool constructAndCheckPath (const string &name, const string &parent-Dir, string &fullPath)

F.26.1 Detailed Description

A class containing utility functions used for IO operations. These functions are class methods.

F.26.2 Member Function Documentation

F.26.2.1 static void BiometricEvaluation::IO::Utility::removeDirectory (const string & directory, const string & prefix) throw (Error::ObjectDoesNotExist, Error::StrategyError) [static]

Remove a directory.

Parameters

directory[in] The name of the directory to be removed, without a preceding path.prefix[in] The path leading to the directory.

Exceptions

Error::ObjectDoesNotExist The named directory does not exist.

Error::StrategyError An error occurred when using the underlying storage system, or the directoy name or prefix is malformed.

Get the size of a file.

Parameters

pathname[in] The name of the file to be sized; can be a complete path.

Returns

The file size.

Exceptions

Error:: ObjectDoesNotExist The named directory does not exist.

Error::StrategyError An error occurred when using the underlying storage system, or pathname is malformed.

F.26.2.3 static bool BiometricEvaluation::IO::Utility::fileExists (const string & pathname) throw (Error::StrategyError) [static]

Indicate whether a file exists.

Parameters

pathname[in] The name of the file to be checked; can be a complete path.

Returns

true if the file exists, false otherwise.

Exceptions

Error::StrategyError An error occurred when using the underlying storage system, or pathname is malformed.

F.26.2.4 static bool BiometricEvaluation::IO::Utility::validateRootName (const string & name) [static]

Check whether or not a string is valid as a name for a rooted entity, such as a Record-Store or other type of container that is persistent within the file system. Notably, name cannot contain path name separators ('/' and '\') or begin with whitespace.

Parameters

name[in] The proposed name for the entity.

Returns

true if the name is acceptable, false otherwise.

F.26.2.5 static bool BiometricEvaluation::IO::Utility::constructAndCheckPath (const string & name, const string & parentDir, string & fullPath) [static]

Construct a full path for a rooted entity, and return true if that path exists; false otherwise.

Parameters

name[in] The proposed name for the entity; cannot be a pathname.parentDir[in] The name of the directory to contain the entity.

fullPath[out] The complete path to the new entity, when when true is returned; ambiguous when false is returned.

Returns

true if the named entiry is present in the file system, false otherwise.

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

• be_io_utility.h

F.27 BiometricEvaluation::Time::Watchdog Class Reference

A Watchdog 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) throw (Error::ParameterError)
- void setInterval (uint64_t interval)
- void start () throw (Error::StrategyError)
- void stop () throw (Error::StrategyError)
- 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

F.27.1 Detailed Description

A Watchdog object can be used by applications to limit the amount of processing time taken by a block of code. A Watchdog 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 builds on the POSIX setitimer(2) call. Timer 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 there own signal handlers, but use the Signal-Manager class instead.

The BEGIN_WATCHDOG_BLOCK macro sets up the jump block and tells the Watchdog object to start handling the alarm signal. Applications must call setInterval() before invoking the BEGIN_WATCHDOG_BLOCK() macro.

The END_WATCHDOG_BLOCK() macro disables the watchdog timer, but doesn't affect the current interval value. Applications can set the interval once and use the BE-GIN/END block macros repeatedly. Failure to call setInterval() results in an effectively disabled timer, as does setting the interval to 0.

Note

Process 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 class. Therefore, applications should not call sleep(3) inside the Watchdog block; behavior is undefined in that case, but usually results in cancellation of the Watchdog timer.

The setCanSigJump(), clearCanSigJump(), setExpired() and clearExpired() 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

E.27.2 Constructor & Destructor Documentation

F.27.2.1 BiometricEvaluation::Time::Watchdog::Watchdog (const uint8_t type) throw (Error::ParameterError)

Construct a new Watchdog object.

Parameters

type[in] The type of timer, ProcessTime or RealTime.

Returns

The Watchdog object.

Exceptions

Error::ParameterError The type is invalid.

F.27.3 Member Function Documentation

F.27.3.1 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 intervals are in microseconds, however actual intervals are dependent on the resolution of the system clock, and may not be at microsecond resolution.

Parameters

interval[in] The timer interval, in microseconds.

F.27.3.2 void BiometricEvaluation::Time::Watchdog::start () throw (Error::StrategyError)

Start a watchdog timer.

Exceptions

Error::StrategyError Could not register the signal handler, or could not create the timer.

F.27.3.3 void BiometricEvaluation::Time::Watchdog::stop () throw (Error::StrategyError)

Stop a watchdog timer.

Exceptions

Error::StrategyError Could not clear the timer.

F.27.3.4 bool BiometricEvaluation::Time::Watchdog::expired ()

Indicate whether the watchdog timer expired.

Returns

true if the timer expired, false otherwise.

F.27.3.5 void BiometricEvaluation::Time::Watchdog::setCanSigJump()

Indicate that the signal handler can jump into the application code after handling the signal.

F.27.3.6 void BiometricEvaluation::Time::Watchdog::clearCanSigJump ()

Clears the flag for the Watchdog object to indicate that the signal jump block is no longer valid.

F.27.3.7 void BiometricEvaluation::Time::Watchdog::setExpired ()

Set a flag to indicate the timer expired.

F.27.3.8 void BiometricEvaluation::Time::Watchdog::clearExpired ()

Clear the flag indicating the timer expired.

F.27.4 Member Data Documentation

F.27.4.1 const uint8_t BiometricEvaluation::Time::Watchdog::PROCESSTIME = 0 [static]

A Watchdog based on process time.

F.27.4.2 const uint8_t BiometricEvaluation::Time::Watchdog::REALTIME = 1 [static]

A Watchdog based on real (wall clock) time.

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

• be_time_watchdog.h

Index

~ArchiveRecordStore	BiometricEvaluation::Error::ObjectDoesNotExist,
BiometricEvalua-	69
tion::IO::ArchiveRecordStore,	ObjectDoesNotExist, 70
33	BiometricEvaluation::Error::ObjectExists,
_canSigJump	70
BiometricEvalua-	ObjectExists, 71
tion::Error::SignalManager,	BiometricEvaluation::Error::ObjectIsClosed,
95	71
_sigJumpBuf	ObjectIsClosed, 72
BiometricEvalua-	BiometricEvaluation::Error::ObjectIsOpen,
tion::Error::SignalManager,	72
95	ObjectIsOpen, 73
	BiometricEvaluation::Error::ParameterError,
ArchiveRecordStore	74
BiometricEvalua-	ParameterError, 74
tion::IO::ArchiveRecordStore,	BiometricEvaluation::Error::SignalManager,
33	92
33	_canSigJump, 95
DE DECCEDE CEO CEADE	_sigJumpBuf, 95
BE_RECSTORE_SEQ_START	clearSigHandled, 95
BiometricEvalua-	clearSignalSet, 94
tion::IO::RecordStore, 91	setDefaultSignalSet, 94
BERKELEYDBTYPE	setSigHandled, 95
BiometricEvalua-	setSignalSet, 93
tion::IO::RecordStore, 91	sigHandled, 94
BiometricEvaluation::Error::ConversionErr	218111111111111111111111111111111111111
39	start, 94
ConversionError, 40	stop, 94
BiometricEvaluation::Error::Exception,	BiometricEvaluation::Error::StrategyError,
47	96
Exception, 48	StrategyError, 96
getInfo, 48	BiometricEvaluation::Error::Utility, 98
BiometricEvaluation::Error::FileError, 50	errorStr, 99
FileError, 50, 51	BiometricEvaluation::Image, 29
Biometric Evaluation :: Error :: Memory Error,	BiometricEvaluation::Image::Image, 57
68	getDepth, 60
MemoryError, 68	getHeight, 60

D D	
getRawData, 59	BiometricEvaluation::IO::FileRecordStore,
getWidth, 59	51
getXResolution, 59	changeName, 57
getYResolution, 59	FileRecordStore, 52, 53
Image, 58	flush, 56
BiometricEvaluation::Image::RawImage,	getSpaceUsed, 53
79	insert, 53
getDepth, 80	length, 55
getHeight, 80	read, 54
getRawData, 81	remove, 54
getWidth, 80	replace, 55
getXResolution, 81	setCursor, 56
getYResolution, 81	BiometricEvaluation::IO::LogCabinet, 60
RawImage, 80	getCount, 63
BiometricEvaluation::IO::ArchiveRecordSt	
31	getName, 63
~ArchiveRecordStore, 33	LogCabinet, 62
ArchiveRecordStore, 33	newLogSheet, 63
changeName, 37	remove, 63
flush, 36	BiometricEvaluation::IO::LogSheet, 64
getArchiveName, 38	getCurrentEntry, 66
getManifestName, 38	getCurrentEntryNumber, 66
getSpaceUsed, 34	LogSheet, 65
insert, 34	newEntry, 66
length, 36	resetCurrentEntry, 66
read, 35	setAutoSync, 67
remove, 35	sync, 67
replace, 35	write, 66
setCursor, 37	BiometricEvaluation::IO::ManifestEntry,
sync, 34	67
vacuum, 37	BiometricEvaluation::IO::Properties, 75
BiometricEvaluation::IO::DBRecordStore,	changeName, 78
41	getProperty, 77
changeName, 46	getPropertyAsInteger, 78
DBRecordStore, 42	Properties, 76
flush, 45	removeProperty, 77
getSpaceUsed, 43	setProperty, 76
insert, 43	setPropertyFromInteger, 77
length, 45	sync, 78
read, 44	BiometricEvaluation::IO::RecordStore,
remove, 44	82
replace, 44	BE_RECSTORE_SEQ_START, 91
setCursor, 46	BERKELEYDBTYPE, 91
sync, 43	changeDescription, 86
BiometricEvaluation::IO::Factory, 49	changeName, 85
openRecordStore, 49	CONTROLFILENAME, 91
· r,	

flush, 89	changeName
getCount, 85	BiometricEvalua-
getDescription, 85	tion::IO::ArchiveRecordStore,
getName, 85	37
getSpaceUsed, 86	BiometricEvalua-
insert, 87	tion::IO::DBRecordStore,
length, 89	46
NAMEPROPERTY, 91	BiometricEvalua-
read, 88	tion::IO::FileRecordStore,
RecordStore, 84	57
remove, 87	BiometricEvalua-
removeRecordStore, 90	tion::IO::Properties, 78
replace, 88	BiometricEvalua-
setCursor, 90	tion::IO::RecordStore, 85
sync, 87	clearCanSigJump
BiometricEvaluation::IO::Utility, 99	BiometricEvalua-
constructAndCheckPath, 101	tion::Time::Watchdog, 105
fileExists, 100	_
getFileSize, 100	clearExpired
removeDirectory, 100	BiometricEvalua-
validateRootName, 101	tion::Time::Watchdog, 105
BiometricEvaluation::Time, 29	clearSigHandled
BiometricEvaluation::Time; 29 BiometricEvaluation::Time::Timer, 97	BiometricEvalua-
elapsed, 98	tion::Error::SignalManager,
start, 97	95
	clearSignalSet
stop, 98 Timer, 97	BiometricEvalua-
	tion::Error::SignalManager,
BiometricEvaluation::Time::Watchdog, 102	94
	constructAndCheckPath
clearCanSigJump, 105	BiometricEvaluation::IO::Utility,
clearExpired, 105	101
expired, 104	CONTROLFILENAME
PROCESSTIME, 105	BiometricEvalua-
REALTIME, 105	tion::IO::RecordStore, 91
setCanSigJump, 105	ConversionError
setExpired, 105	BiometricEvalua-
setInterval, 104	tion::Error::ConversionError,
start, 104	40
stop, 104	
Watchdog, 103	DBRecordStore
BiometricEvaluation::Utility::AutoArray,	BiometricEvalua-
39	tion::IO::DBRecordStore,
ahan aa Dagawinti an	42
changeDescription	· -
BiometricEvalua-	-lamas d
tion::IO::RecordStore, 86	elapsed

BiometricEvaluation::Time::Timer,	BiometricEvaluation::IO::LogSheet,
98	66
errorStr	getCurrentEntryNumber
BiometricEvaluation::Error::Utility, 99	BiometricEvaluation::IO::LogSheet, 66
Exception	getDepth
BiometricEvalua-	BiometricEvaluation::Image::Image,
tion::Error::Exception, 48	60
expired	BiometricEvalua-
BiometricEvalua-	tion::Image::RawImage, 80
tion::Time::Watchdog, 104	getDescription
	BiometricEvalua-
FileError	tion::IO::LogCabinet, 63
BiometricEvalua-	BiometricEvalua-
tion::Error::FileError, 50,	tion::IO::RecordStore, 85
51	getFileSize
fileExists	BiometricEvaluation::IO::Utility,
BiometricEvaluation::IO::Utility,	100
100	getHeight
FileRecordStore	BiometricEvaluation::Image::Image,
BiometricEvalua-	60
tion::IO::FileRecordStore,	BiometricEvalua-
52, 53	tion::Image::RawImage, 80
flush	getInfo
BiometricEvalua-	BiometricEvalua-
tion::IO::ArchiveRecordStore,	tion::Error::Exception, 48
36	getManifestName
B:	
BiometricEvalua-	BiometricEvalua-
BiometricEvalua- tion::IO::DBRecordStore,	BiometricEvalua- tion::IO::ArchiveRecordStore,
tion::IO::DBRecordStore,	tion::IO::ArchiveRecordStore,
tion::IO::DBRecordStore,	tion::IO::ArchiveRecordStore, 38
tion::IO::DBRecordStore, 45 BiometricEvalua-	tion::IO::ArchiveRecordStore, 38 getName
tion::IO::DBRecordStore, 45 BiometricEvalua- tion::IO::FileRecordStore,	tion::IO::ArchiveRecordStore, 38 getName BiometricEvalua-
tion::IO::DBRecordStore, 45 BiometricEvaluation::IO::FileRecordStore, 56	tion::IO::ArchiveRecordStore, 38 getName BiometricEvalua- tion::IO::LogCabinet, 63
tion::IO::DBRecordStore, 45 BiometricEvalua- tion::IO::FileRecordStore, 56 BiometricEvalua-	tion::IO::ArchiveRecordStore, 38 getName BiometricEvalua- tion::IO::LogCabinet, 63 BiometricEvalua-
tion::IO::DBRecordStore, 45 BiometricEvalua- tion::IO::FileRecordStore, 56 BiometricEvalua-	tion::IO::ArchiveRecordStore, 38 getName BiometricEvalua- tion::IO::LogCabinet, 63 BiometricEvalua- tion::IO::RecordStore, 85
tion::IO::DBRecordStore, 45 BiometricEvalua- tion::IO::FileRecordStore, 56 BiometricEvalua- tion::IO::RecordStore, 89	tion::IO::ArchiveRecordStore, 38 getName BiometricEvalua- tion::IO::LogCabinet, 63 BiometricEvalua- tion::IO::RecordStore, 85 getProperty
tion::IO::DBRecordStore, 45 BiometricEvalua- tion::IO::FileRecordStore, 56 BiometricEvalua- tion::IO::RecordStore, 89 getArchiveName	tion::IO::ArchiveRecordStore, 38 getName BiometricEvalua- tion::IO::LogCabinet, 63 BiometricEvalua- tion::IO::RecordStore, 85 getProperty BiometricEvalua-
tion::IO::DBRecordStore, 45 BiometricEvalua- tion::IO::FileRecordStore, 56 BiometricEvalua- tion::IO::RecordStore, 89 getArchiveName BiometricEvalua-	tion::IO::ArchiveRecordStore, 38 getName BiometricEvalua- tion::IO::LogCabinet, 63 BiometricEvalua- tion::IO::RecordStore, 85 getProperty BiometricEvalua- tion::IO::Properties, 77
tion::IO::DBRecordStore, 45 BiometricEvalua- tion::IO::FileRecordStore, 56 BiometricEvalua- tion::IO::RecordStore, 89 getArchiveName BiometricEvalua- tion::IO::ArchiveRecordStore,	tion::IO::ArchiveRecordStore, 38 getName BiometricEvalua- tion::IO::LogCabinet, 63 BiometricEvalua- tion::IO::RecordStore, 85 getProperty BiometricEvalua- tion::IO::Properties, 77 getPropertyAsInteger
tion::IO::DBRecordStore, 45 BiometricEvalua- tion::IO::FileRecordStore, 56 BiometricEvalua- tion::IO::RecordStore, 89 getArchiveName BiometricEvalua- tion::IO::ArchiveRecordStore, 38	tion::IO::ArchiveRecordStore, 38 getName BiometricEvalua- tion::IO::LogCabinet, 63 BiometricEvalua- tion::IO::RecordStore, 85 getProperty BiometricEvalua- tion::IO::Properties, 77 getPropertyAsInteger BiometricEvalua-
tion::IO::DBRecordStore, 45 BiometricEvalua- tion::IO::FileRecordStore, 56 BiometricEvalua- tion::IO::RecordStore, 89 getArchiveName BiometricEvalua- tion::IO::ArchiveRecordStore, 38 getCount	tion::IO::ArchiveRecordStore, 38 getName BiometricEvalua- tion::IO::LogCabinet, 63 BiometricEvalua- tion::IO::RecordStore, 85 getProperty BiometricEvalua- tion::IO::Properties, 77 getPropertyAsInteger BiometricEvalua- tion::IO::Properties, 78
tion::IO::DBRecordStore, 45 BiometricEvalua- tion::IO::FileRecordStore, 56 BiometricEvalua- tion::IO::RecordStore, 89 getArchiveName BiometricEvalua- tion::IO::ArchiveRecordStore, 38 getCount BiometricEvalua- tion::IO::LogCabinet, 63 BiometricEvalua-	tion::IO::ArchiveRecordStore, 38 getName BiometricEvalua- tion::IO::LogCabinet, 63 BiometricEvalua- tion::IO::RecordStore, 85 getProperty BiometricEvalua- tion::IO::Properties, 77 getPropertyAsInteger BiometricEvalua- tion::IO::Properties, 78 getRawData BiometricEvaluation::Image::Image, 59
tion::IO::DBRecordStore, 45 BiometricEvalua- tion::IO::FileRecordStore, 56 BiometricEvalua- tion::IO::RecordStore, 89 getArchiveName BiometricEvalua- tion::IO::ArchiveRecordStore, 38 getCount BiometricEvalua- tion::IO::LogCabinet, 63	tion::IO::ArchiveRecordStore, 38 getName BiometricEvalua- tion::IO::LogCabinet, 63 BiometricEvalua- tion::IO::RecordStore, 85 getProperty BiometricEvalua- tion::IO::Properties, 77 getPropertyAsInteger BiometricEvalua- tion::IO::Properties, 78 getRawData BiometricEvaluation::Image::Image,

getSpaceUsed	BiometricEvalua-
BiometricEvalua-	tion::IO::ArchiveRecordStore,
tion::IO::ArchiveRecordStore,	36
34	BiometricEvalua-
BiometricEvalua-	tion::IO::DBRecordStore,
tion::IO::DBRecordStore,	45
43	BiometricEvalua-
BiometricEvalua-	tion::IO::FileRecordStore,
tion::IO::FileRecordStore,	55
53	BiometricEvalua-
BiometricEvalua-	tion::IO::RecordStore, 89
tion::IO::RecordStore, 86	LogCabinet
getWidth	BiometricEvalua-
BiometricEvaluation::Image::Image,	tion::IO::LogCabinet, 62
59	LogSheet
BiometricEvalua-	BiometricEvaluation::IO::LogSheet,
tion::Image::RawImage, 80	65
getXResolution	
BiometricEvaluation::Image::Image,	MemoryError
59	BiometricEvalua-
BiometricEvalua-	tion::Error::MemoryError,
tion::Image::RawImage, 81	68
getYResolution	NA MEDD ODEDWY
BiometricEvaluation::Image::Image,	NAMEPROPERTY
59	BiometricEvalua-
BiometricEvalua-	tion::IO::RecordStore, 91
tion::Image::RawImage, 81	newEntry RiometricEvaluation::IO::LogSheet
	BiometricEvaluation::IO::LogSheet,
Image	newLogSheet
BiometricEvaluation::Image::Image,	BiometricEvalua-
58	tion::IO::LogCabinet, 63
insert	tionioDogettomet, 05
BiometricEvalua-	ObjectDoesNotExist
tion::IO::ArchiveRecordStore,	BiometricEvalua-
	DidiicuicEvalua-
34	
BiometricEvalua-	tion::Error::ObjectDoesNotExist,
BiometricEvalua- tion::IO::DBRecordStore,	tion::Error::ObjectDoesNotExist,
BiometricEvaluation::IO::DBRecordStore, 43	tion::Error::ObjectDoesNotExist, 70
BiometricEvalua- tion::IO::DBRecordStore, 43 BiometricEvalua-	tion::Error::ObjectDoesNotExist, 70 ObjectExists
BiometricEvalua- tion::IO::DBRecordStore, 43 BiometricEvalua- tion::IO::FileRecordStore,	tion::Error::ObjectDoesNotExist, 70 ObjectExists BiometricEvalua-
BiometricEvalua- tion::IO::DBRecordStore, 43 BiometricEvalua- tion::IO::FileRecordStore, 53	tion::Error::ObjectDoesNotExist, 70 ObjectExists BiometricEvaluation::Error::ObjectExists, 71 ObjectIsClosed
BiometricEvalua- tion::IO::DBRecordStore, 43 BiometricEvalua- tion::IO::FileRecordStore, 53 BiometricEvalua-	tion::Error::ObjectDoesNotExist, 70 ObjectExists BiometricEvalua- tion::Error::ObjectExists, 71 ObjectIsClosed BiometricEvalua-
BiometricEvalua- tion::IO::DBRecordStore, 43 BiometricEvalua- tion::IO::FileRecordStore, 53	tion::Error::ObjectDoesNotExist, 70 ObjectExists BiometricEvalua- tion::Error::ObjectExists, 71 ObjectIsClosed BiometricEvalua- tion::Error::ObjectIsClosed,
BiometricEvaluation::IO::DBRecordStore, 43 BiometricEvaluation::IO::FileRecordStore, 53 BiometricEvaluation::IO::RecordStore, 87	tion::Error::ObjectDoesNotExist, 70 ObjectExists BiometricEvalua- tion::Error::ObjectExists, 71 ObjectIsClosed BiometricEvalua- tion::Error::ObjectIsClosed, 72
BiometricEvalua- tion::IO::DBRecordStore, 43 BiometricEvalua- tion::IO::FileRecordStore, 53 BiometricEvalua-	tion::Error::ObjectDoesNotExist, 70 ObjectExists BiometricEvalua- tion::Error::ObjectExists, 71 ObjectIsClosed BiometricEvalua- tion::Error::ObjectIsClosed,

BiometricEvalua-	BiometricEvalua-
tion::Error::ObjectIsOpen,	tion::IO::FileRecordStore,
73	54
	BiometricEvalua-
openRecordStore	
BiometricEvaluation::IO::Factory,	tion::IO::LogCabinet, 63 BiometricEvalua-
49	Brometre variation
Domonoston Emon	tion::IO::RecordStore, 87
ParameterError	removeDirectory
BiometricEvalua-	BiometricEvaluation::IO::Utility,
tion::Error::ParameterError,	100
74	removeProperty
PROCESSTIME	BiometricEvalua-
BiometricEvalua-	tion::IO::Properties, 77
tion::Time::Watchdog, 105	removeRecordStore
Properties	BiometricEvalua-
BiometricEvalua-	tion::IO::RecordStore, 90
tion::IO::Properties, 76	replace
D. I	BiometricEvalua-
RawImage	tion::IO::ArchiveRecordStore,
BiometricEvalua-	35
tion::Image::RawImage, 80	BiometricEvalua-
read	tion::IO::DBRecordStore,
BiometricEvalua-	44
tion::IO::ArchiveRecordStore,	BiometricEvalua-
35	tion::IO::FileRecordStore,
BiometricEvalua-	55
tion::IO::DBRecordStore,	BiometricEvalua-
44	tion::IO::RecordStore, 88
BiometricEvalua-	resetCurrentEntry
tion::IO::FileRecordStore,	BiometricEvaluation::IO::LogSheet,
54	66
BiometricEvalua-	
tion::IO::RecordStore, 88	
REALTIME	setAutoSync
BiometricEvalua-	BiometricEvaluation::IO::LogSheet,
tion::Time::Watchdog, 105	67
RecordStore	setCanSigJump
BiometricEvalua-	BiometricEvalua-
tion::IO::RecordStore, 84	tion::Time::Watchdog, 105
remove	setCursor
BiometricEvalua-	BiometricEvalua-
tion::IO::ArchiveRecordStore,	tion::IO::ArchiveRecordStore,
35	37
BiometricEvalua-	BiometricEvalua-
tion::IO::DBRecordStore,	tion::IO::DBRecordStore,
44	46

BiometricEvalua-	BiometricEvalua-
tion::IO::FileRecordStore,	tion::Error::SignalManager,
56	94
BiometricEvalua-	BiometricEvaluation::Time::Timer,
tion::IO::RecordStore, 90	98
setDefaultSignalSet	BiometricEvalua-
BiometricEvalua-	tion::Time::Watchdog, 104
tion::Error::SignalManager,	StrategyError
94	BiometricEvalua-
setExpired	tion::Error::StrategyError,
BiometricEvalua-	96
tion::Time::Watchdog, 105	sync
setInterval	BiometricEvalua-
BiometricEvalua-	tion::IO::ArchiveRecordStore,
tion::Time::Watchdog, 104	34
setProperty	BiometricEvalua-
BiometricEvalua-	tion::IO::DBRecordStore,
tion::IO::Properties, 76	43
setPropertyFromInteger	BiometricEvaluation::IO::LogSheet,
BiometricEvalua-	67
tion::IO::Properties, 77	BiometricEvalua-
setSigHandled	tion::IO::Properties, 78
BiometricEvalua-	BiometricEvalua-
tion::Error::SignalManager,	tion::IO::RecordStore, 87
95	
setSignalSet	Timer
BiometricEvalua-	BiometricEvaluation::Time::Timer,
tion::Error::SignalManager,	97
93	
sigHandled	vacuum
BiometricEvalua-	BiometricEvalua-
tion::Error::SignalManager,	tion::IO::ArchiveRecordStore,
94	37
SignalManager	validateRootName
BiometricEvalua-	BiometricEvaluation::IO::Utility,
tion::Error::SignalManager,	101
93	Watahda
start	Watchdog BiometricEvalua-
BiometricEvalua-	tion::Time::Watchdog, 103
tion::Error::SignalManager,	write
94	
BiometricEvaluation::Time::Timer,	BiometricEvaluation::IO::LogSheet, 66
97	00
BiometricEvalua-	
tion::Time::Watchdog, 104	
stop	